第 5 章

实践项目：机械手掌

智能视觉机械手掌全机身有 6 个自由度，手指关节有 5 个自由度，采用微型伺服电动机，可水平转动 180°。机身搭载高清晰度摄像头，基于 TensorFlow 和 OpenCV 可以轻松实现项视觉识别功能。其结构如图 5.1 所示。

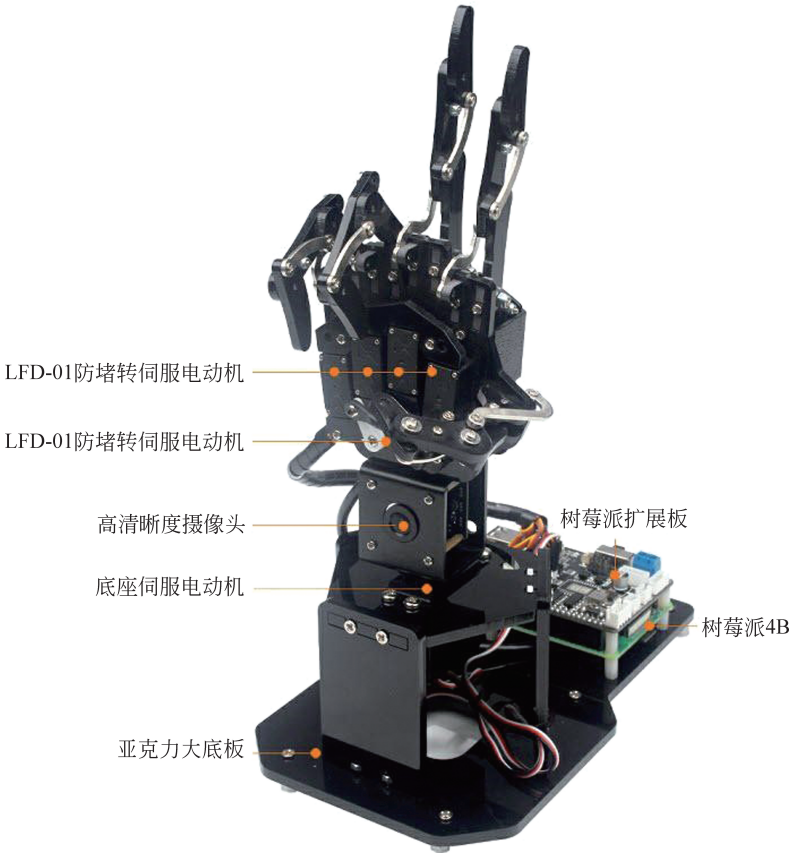

图 5.1　机械手掌

5.1 连接与控制

机械手掌本质上是一个基于智能视觉的机器人，其控制中枢是手掌上搭载的树莓派模块，在机械手掌正常运行前需要对其进行一些功能上的设定。本节主要讲述如何通过计算机连接机械手掌的树莓派控制板，树莓派程序架构以及如何进行机械手掌相应的实践功能。

5.1.1 远程连接机械手掌

在开始实践项目之前，首先连接智能视觉机械手掌，具体步骤如下。

（1）将 DC 电源对接线的红色和黑色分别连接至树莓派扩展板的正（+）、负极（−），将电源适配器的外接口与 DC 电源对接线的内接口连接。

（2）将树莓派扩展板的开关由 OFF 推动到 ON，此时树莓派的 LED1、LED2 常亮，稍等片刻后 LED1 由常亮变为每 2s 闪烁一次，同时手掌会做出抓取姿态，设备开机成功。

（3）机械手掌开机后，会产生一个以 HW 开头的热点，打开计算机 WiFi 可以搜索到该热点。

（4）在 VNC Viewer 中输入树莓派默认 IP 地址 192.168.149.1，按回车键，输入默认账号和密码，单击 OK 按钮，可以看到远程打开的树莓派桌面。

5.1.2 程序架构

程序结构如图 5.2 所示，包含主程序界面设计和各个实践项目的子程序设计。主程序文件名为 Transfer_Play.py，各实践项目子程序在文件 CvHand.py 中。

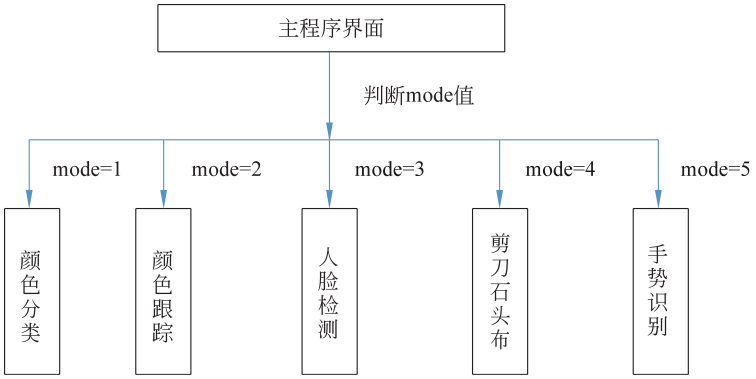

图 5.2 程序架构

在 Transfer_Play.py 中，项目跳转的程序代码如图 5.3 所示。

用户输入的跳转序列号保存在全局变量 mode 中，根据 mode 中的数值启动对应的项目子程序。例如，在 LX 终端执行命令 python3 Transfer_Play.py -1，跳转到 CvHand.py 的第 108 行（def cv_color_identify），如图 5.4 所示（对应图 5.3 第 77 代码中的 cv_color_identify），启动颜色分类项目。

```
71
72      yolo = YOLO()
73      while True:
74          try:
75              if frame is not None:
76                  if mode == 1:
77                      Frame, color = cv_color_identify(frame, 320, 240)
78                      image = cv2ImgAddText(frame, "颜色分类", text_x, text_y, textColor=textcolor, textSize=textsize)
79                      if color is not None:
80                          cv2.putText(image, "Color: " + color, (10, image.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.65, range_rgb[color], 2)
81                      else:
82                          cv2.putText(image, "Color: None", (10, image.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.65, (0, 0, 0), 2)
83                  elif mode == 2:
84                      Frame = cv_color_track(frame, 320, 240)
85                      image = cv2ImgAddText(Frame, "颜色跟踪", text_x, text_y, textColor=textcolor, textSize=textsize)
86                  elif mode == 3:
87                      Frame = cv_face_detection(frame, 240, 180)
88                      image = cv2ImgAddText(Frame, "人脸检测", text_x, text_y, textColor=textcolor, textSize=textsize)
89                  elif mode == 4:
90                      Frame = cv_rockpaperscissor(frame, 160, 160, yolo)
91                      image = cv2ImgAddText(Frame, "剪刀石头布", text_x, text_y, textColor=textcolor, textSize=textsize)
92                  elif mode == 5:
93                      Frame, result = cv_hand_gesture(frame, 160, 120)
94                      if result != '':
95                          cv2.putText(frame, result, (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)
96                      image = cv2ImgAddText(frame, "手势识别", text_x, text_y, textColor=textcolor, textSize=textsize)
97                  else:
98                      image = frame
99                      time.sleep(0.01)
100                 frame = None
101             else:
102                 time.sleep(0.01)
103         except BaseException as e:
104             print('主程序出错', e)
```

图 5.3　项目跳转的程序代码

```
108 def cv_color_identify(frame, rw, rh):
109     global color_list, last_Color, Color
110     global get_Color, action_finish_color_identify
111
112     frame_resize = cv2.resize(frame, (rw, rh), interpolation = cv2.INTER_CUBIC)
113     frame_GaussianBlur = cv2.GaussianBlur(frame_resize, (3,3), 0)#高斯模糊
114     frame_lab = cv2.cvtColor(frame_GaussianBlur, cv2.COLOR_BGR2LAB)#将图片转换到LAB空间
```

图 5.4　颜色空间转换代码

5.2　颜 色 分 类

给机械手掌加上摄像头视觉模块，通过视觉识别可以让它识别不同的颜色。

本实践项目使用 Lab 颜色空间进行颜色识别。首先将 RGB 颜色空间转换为 Lab 空间，然后进行二值化处理，再经过膨胀、腐蚀等操作，可获得只包含目标颜色的轮廓，最后将该颜色轮廓用圆圈框起。

1. 颜色空间转换

调用 OpenCV 颜色转换函数 cv2.cvtColor(Img，Transform_model) 将 RGB 颜色空间转换为 Lab 空间，其中参数 Img 为图片对象，参数 Transform_model 指定色彩转换模式。代码如图 5.4 所示。

2. 轮廓检测

调用 OpenCV 的 findContours(Img，Model，Method) 函数获取不同颜色小球的轮廓，代码如图 5.5 所示。其中，Img 表示要检测轮廓的图片对象，该图片需要灰度图片；Model 表示轮廓的检测模式，v2.RETR_EXTERNAL 表示只检测外轮廓；Method 表示轮廓的逼近方法(检测策略)，cv2.CHAIN_APPROX_NONE 存储所有的轮廓点，相邻的两个点的像素位置差不超过 1，即 max(abs(x1－x2)，abs(y2－y1))＝＝1。

```
    areaMaxContour = 0
    max_area = 0
    area_max = 0

    centerX = 0
    centerY = 0
    radius = 0
    for i in color_range:
        if i != 'black' and i != 'white' and i != 'green':
            frame_mask = cv2.inRange(frame_lab, color_range[i][0], color_range[i][1])#对原图像和掩模进行位运算
            opened = cv2.morphologyEx(frame_mask, cv2.MORPH_OPEN, np.ones((3,3),np.uint8))#开运算
            closed = cv2.morphologyEx(opened, cv2.MORPH_CLOSE, np.ones((3,3),np.uint8))#闭运算
            contours = cv2.findContours(closed, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)[-2]#找出轮廓
            areaMaxContour, area_max = getAreaMaxContour(contours)#找出最大轮廓
            if areaMaxContour is not None:
                if area_max > max_area:#找最大面积
                    max_area = area_max
                    color_max = i
                    areaMaxContour_max = areaMaxContour
```

图 5.5　轮廓检测代码

3. 圈出识别的颜色小球

在获取小球轮廓的基础上，使用 cv2.circle() 函数圈出指定颜色的小球，代码如图 5.6 所示。

```
    if max_area != 0:
        ((centerX, centerY), radius) = cv2.minEnclosingCircle(areaMaxContour_max)  # 获取最小外接圆
        centerX = int(leMap(centerX, 0, rw, 0, 640))
        centerY = int(leMap(centerY, 0, rh, 0, 480))
        radius = int(leMap(radius, 0, rh, 0, 480))

        centerX, centerY, radius = int(centerX), int(centerY), int(radius)#获取圆心，半径
        if color_max == 'red':  #红色最大
            Color_BGR = range_rgb["red"]
            color = 1
        elif color_max == 'green':  #绿色最大
            Color_BGR = range_rgb["green"]
            color = 2
        elif color_max == 'blue':  #蓝色最大
            Color_BGR = range_rgb["blue"]
            color = 3
        else:
            color = 0
            Color_BGR = range_rgb["black"]

        cv2.circle(frame, (centerX, centerY), radius, Color_BGR, 2)#画圆
        color_list.append(color)
        if len(color_list) >= 5:
            color = int(round(np.mean(color_list)))
            color_list = []
            #print(get_Color, actionfinish, Color)
            if color == 1:
                if action_finish_color_identify:
                    Color = 'red'
                    get_Color = True
            elif color == 2:
                if action_finish_color_identify:
                    Color = 'green'
                    get_Color = True
            elif color == 3:
                if action_finish_color_identify:
                    Color = 'blue'
                    get_Color = True
            else:
                Color = None
```

图 5.6　圈出指定颜色的小球代码

4. 实践流程

（1）在 LX 终端执行命令 cd uHand_Pi，定位到程序目录。

（2）执行命令 python3 Transfer_Play.py -1，将小球正对着摄像头，待识别到小球颜色且未闭合前，将小球放置在手掌心的位置，手掌会短暂抓取小球并转动（红色时云台向左

转，蓝色时向右转）。

（3）当摄像头识别后，会将红色小球框起来，如图 5.7 所示。

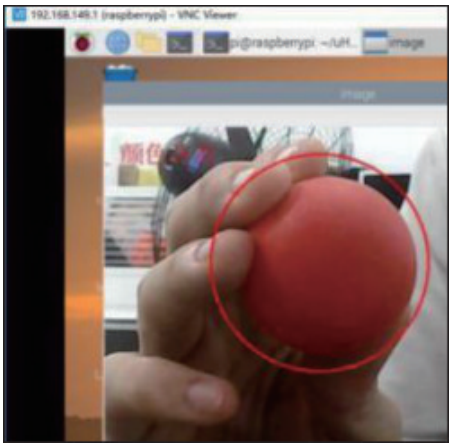

图 5.7　红色识别

5.3　颜 色 跟 踪

通过增加随着识别的颜色小球运动而运动的程序代码，达到颜色跟踪的目的。其原理是应用 PID 控制，驱动手掌云台转动，代码如图 5.8 所示。

```
            servo6_pid.SetPoint = img_center_x#设定
            center_x = 2*img_center_x - centerX
            if abs(img_center_x - center_x) < 10:
                center_x = img_center_x
            servo6_pid.update(center_x)#当前
            servo6_pwm = servo6_pid.output#输出
            servo6_color_track += servo6_pwm
            servo6_color_track = int(servo6_color_track)
            if servo6_color_track < 500:
                servo6_color_track = 500
            elif servo6_color_track > 2500:
                servo6_color_track = 2500
            centerX = int(leMap(centerX, 0, rw, 0, 640))
            centerY = int(leMap(centerY, 0, rh, 0, 480))
            radius = int(leMap(radius, 0, rw, 0, 640))
            cv2.circle(frame, (int(centerX), int(centerY)), int(radius), range_rgb[target_color], 2)
            if action_finish:
                dis_ok = True
```

图 5.8　颜色跟踪代码

执行命令 python3 Transfer_Play.py -2，通过摄像头模块识别指定颜色的物体，并实现手掌云台跟随目标颜色物体的功能。

注意，实践过程中手握小球移动时动作不可过快，并且要保证小球一直出现在摄像头的画面内。这样，手掌会跟随小球移动的方向进行对应的转动。

5.4　人 脸 检 测

首先调用人脸分类器正脸样本图像，然后将图像转换为灰度图像，最后使用分类器检测人脸，如果有返回则将人脸框出来。

人脸检测的关键是提取人脸的特征。对于一幅图像，提取出图像的细节对产生稳定分类结果和追踪结果很有用，这些提取的结果被称为特征。对于给定的图像，特征可能会因为区域的大小而有所不同，区域大小也被称为窗口大小。即使窗口大小不一样，仅在尺度上不同的两幅图像也应该有相似的特征。因此，生成能适应于不同大小窗口的特征是图像处理中非常关键的一点。这些大小不同的窗口中的同一种特征的集合被称为级联。

OpenCV 提供了尺度不变的 Haar 级联的分类器和跟踪器，在进行程序编写之前，将 OpenCV 源代码的 haarcascades 文件夹中有关人脸检测的 haarcascade_fullbody.xml 文件复制到保存人脸检测代码所在的文件夹 VOCData 中。

人脸检测的代码如图 5.9 所示。

```
face_x = 0
detector = dlib.get_frontal_face_detector() #获取人脸分类器
get_face = False
def cv_face_detection(frame, rw, rh):
    global get_face, face_x
    face_x = 0
    frame_resize = cv2.resize(frame, (rw, rh), interpolation = cv2.INTER_CUBIC)
    gray = cv2.cvtColor(frame_resize, cv2.COLOR_BGR2GRAY)
    dets = detector(gray) #使用detector进行人脸检测 dets为返回的结果
    if len(dets) > 0:
        for face in dets:
            x1, y1, x2, y2 = coordinate_map(face.left(), face.top(), face.right(), face.bottom(), rw, rh)
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
            face_x = (x1 + x2)/2
        if not get_face:
            get_face = True
    return frame
```

图 5.9　人脸检测代码

执行命令 python3 Transfer_Play.py -3，通过手掌的摄像头模块检测人脸。没检测到人脸时，手掌在 0°～180°范围左右转动；检测到人脸时，手掌停止转动，并招手打招呼，直到人脸离开检测范围，手掌恢复在 0°～180°左右转动。

5.5　石头剪刀布

本节的原理是机器学习。首先进行图像采集，对图像中手势部分进行标注；然后训练标注的信息，获得模型；最后调用模型检测对应的手势，做出相应的动作。手势标注代码如图 5.10 所示。

```
hand_gesture = None
def cv_rockpaperscissor(frame, rw, rh, yolo):
    global hand_gesture

    #将图像转换成可处理的格式
    Frame_ = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
    #获取检测到目标的两个矩形对角线顶点，物体类别标签
    frame_, (left, top), (right, bottom), label= yolo.detect_image(Frame_)
    frame_rgb = cv2.cvtColor(np.asarray(frame_), cv2.COLOR_RGB2BGR)
    if label is not None:
        hand_gesture = label.split(' ')[0]
    else:
        hand_gesture = None
    return frame_rgb
```

图 5.10　手势标注代码

执行命令 python3 Transfer_Play.py -4。通过摄像头模块检测人手的石头剪刀布手

势，待手掌自主判断后，它会做出对应的手势与人进行猜拳游戏。比如人伸出“剪刀”手势，它识别后会出“拳头”。

5.6 手势识别

按 2.3.7 节中创建人脸识别的方法，在百度智能云的平台的“产品”|“人工智能”|“人体分析”中创建手势识别应用。创建完毕后，记录申请到的 AppID、API Key 和 Secret Key。

1. 程序设计

（1）将获取的 APP_ID、API_KEY 和 SECRET_KEY 填入程序代码中，如图 5.11 所示。

```
#APP_ID = ''
#API_KEY = ''
#SECRET_KEY = ''

def get_token():
    host = 'https://aip.baidubce.com/oauth/2.0/token?grant_type=client_credentials&client_id=%s&client_secret=%s'%(API_KEY, SECRET_KEY)
    response = requests.get(host, timeout=2)
    if response:
        try:
            content = response.json()['access_token']
            return content
        except:
            return None
    else:
        return None
```

图 5.11 填写手势识别关键信息

（2）通过手势控制的在线识别功能获取识别结果，程序代码如图 5.12 所示。

```
def cv_hand_gesture(frame, rw, rh):
    global request_url, gesture

    frame_resize = cv2.resize(frame, (rw, rh), interpolation = cv2.INTER_CUBIC)
    try:
        if request_url is None:
            request_url = get_url()
            if request_url is None:
                Frame = cv2ImgAddText(frame, "", 200, 100, textColor=(255, 0, 0), textSize=30)
                return Frame, ''
        else:
            params = {"image":''+ image_to_base64(cv2.cvtColor(frame_resize, cv2.COLOR_BGR2RGB)) + ''}
            headers = {'content-type': 'application/x-www-form-urlencoded'}
            response = requests.post(request_url, data=params, headers=headers)

            if response:
                res = response.json()
                try:
                    result = res['result'][0]
                    #x, y, w, h = result['left'], result['top'], result['width'], result['height']
                    gesture = result['classname']
                    #x, y, w, h = coordinate_map(x, y, w, h, rw, rh)
                    #cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
                    #cv2.putText(frame, gesture, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
                except:
                    Frame = cv2ImgAddText(frame, res, 100, 100, textColor=(255, 0, 0), textSize=30)
                    return Frame, ''
            else:
                result = []
            return frame, gesture
```

图 5.12 获取识别结果代码

（3）根据手势识别的结果，驱动机械手掌做出相应的反馈动作，程序代码如图 5.13 所示。

2. 实践流程

（1）在 LX 终端下进入 uHand_Pi 文件夹，执行命令 sudo vim cv_Hand.py，将 AppID、

```
415                 if gesture != '':
416 #                   print(gesture)
417                     if gesture == 'One':
418                         LeArm.runActionGroup('2_1_2', 1)
419                     elif gesture == 'Two':
420                         LeArm.runActionGroup('6_2_23', 1)
421                     elif gesture == 'Three':
422                         LeArm.runActionGroup('11_3_234', 1)
423                     elif gesture == 'Four':
424                         LeArm.runActionGroup('14_4_2345', 1)
425                     elif gesture == 'Five':
426                         LeArm.runActionGroup('15_5_12345', 1)
427                     elif gesture == 'Fist':
428                         LeArm.runActionGroup('0_0_0', 1)
429                     elif gesture == 'Ok':
430                         LeArm.runActionGroup('13_3_345', 1)
431                     elif gesture == 'Thumb_up':
432                         LeArm.runActionGroup('1_1_1', 1)
433                     elif gesture == 'ILY':
434                         LeArm.runActionGroup('10_3_125', 1)
435                     elif gesture == 'Eight':
436                         LeArm.runActionGroup('4_2_12', 1)
437                     elif gesture == 'Rock':
438                         LeArm.runActionGroup('7_2_25', 1)
439                     elif gesture == 'Six':
440                         LeArm.runActionGroup('5_2_15', 1)
441                 else:
442                     time.sleep(0.01)
```

图 5.13 反馈动作代码

API Key、Secret Key 添加到如图 5.14 所示的位置，保存并退出。

```
pi@raspberrypi: ~/uHand_Pi
文件(F)  编辑(E)  标签(T)  帮助(H)
234                 get_face = True
235         return frame
236 ##########################################################################################
    ################
237 """ 人体分析 APPID AK SK """
238 APP_ID = ''
239 API_KEY = ''
240 SECRET_KEY = ''
241 def get_token():
242     host = 'https://aip.baidubce.com/oauth/2.0/token?grant_type=client_credentials&client_
    id=%s&client_secret=%s'%(API_KEY, SECRET_KEY)
243     response = requests.get(host, timeout=2)
244     if response:
245         try:
246             content = response.json()['access_token']
247             return content
248         except:
249             return None
250     else:
251         return None
252
253 def image_to_base64(image_np):
254  image = cv2.imencode('.jpg',image_np)[1]
255  image_code = str(base64.b64encode(image), 'utf-8')
256
                                                                    251,19        56%
```

图 5.14 修改参数

（2）执行命令 sudo reboot，重启树莓派。

（3）执行命令 python3 Transfer_play.py -5。利用机器人摄像头模块检测人手的手势，自动判断后，模拟人手做出相应的手势动作。

第 6 章 实践项目：视觉人形机器人

视觉人形机器人搭载 2 自由度云台和高清晰度摄像头，支持 AI 图像识别，可以实现颜色识别、自动踢球、视觉巡线等多种功能。

6.1 项目启动

在 LX 终端定位到 human_code 目录，执行命令 ls -l，显示该目录下的所有文件，如图 6.1 所示。

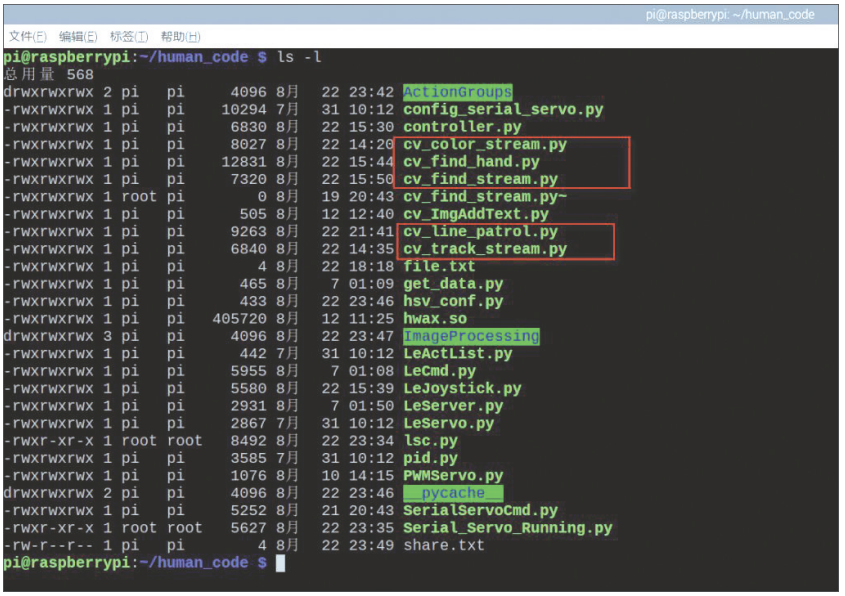

图 6.1 项目对应的文件

每个 cv 开头的文件都对应一个实践项目，如表 6.1 所示。

表 6.1 项目对应指令

序号	项 目 名	对 应 指 令
1	自主巡线	python3 cv_line_patrol.py
2	点球射门	python3 cv_find_stream.py

续表

序号	项 目 名	对 应 指 令
3	云台跟踪	python3 cv_track_stream.py
4	物品识别	python3 cv_color_stream.py
5	手势交互	python3cv_find_hand.py

执行命令 python3 cv_line_patrol.py，就可以运行实践项目“自主巡线”，以此类推。

6.2 自 主 巡 线

很多机器人比赛里都要求机器人能够沿着黑线行进，接下来通过编写程序实现机器人识别黑色。

1. 黑线识别

黑线识别的原理是先将图像缩小，然后转为灰度图，接着进行二值化、腐蚀、膨胀等操作去除噪点。识别到的颜色在画面中会呈现白色，没识别的颜色则为黑色。最后将图像分割成三段进行处理，每一段都用矩形将黑线框起来，并且标出中心点，用这 3 个点来近似的表示这条线。

机器人通过摄像头识别出黑线的过程如图 6.2 所示。

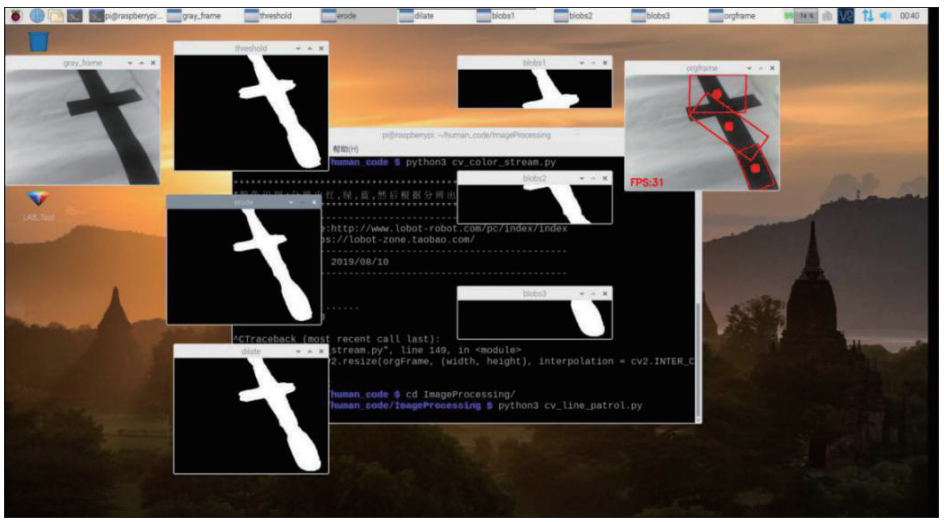

图 6.2 黑线识别过程

2. 自主巡线

将图像处理部分与机器人动作相结合，就可以让机器人在识别黑线的基础上实现自主巡线，程序代码如图 6.3 所示。

执行命令 python3 cv_line_patrol.py，启动自主巡线程序，机器人将沿着黑色线路进行巡线前行，如图 6.4 所示。