重要代码
未来道具研究所

```
import numpy

import cv2



# 差异度（欧氏距离）阈值——越小，识别越准|越难

FACE_MATCH_THRESHOLD = 0.95

matching = False

v = 0

text_color = (0, 255, 0)

rectangle_color = (0, 255, 0)

score_list=[]


# !!! 图像特征提取（128 维特征值—列表）——在 NCS 中调用模型-facenet.graph !!!

def run_inference(image_to_classify, facenet_graph):



    # 图像预处理———大小、通道顺序

    resized_image = preprocess_image(image_to_classify)

    # 图像传至 NCS

    facenet_graph.LoadTensor(resized_image.astype(numpy.float16), None)

    # !!! 特征提取 !!!


    output, userobj = facenet_graph.GetResult()
```

```
        # 返回特征值

        return output


    # 图像格式标准化调整

    def whiten_image(source_image):



        source_mean = numpy.mean(source_image)

        source_standard_deviation = numpy.std(source_image)

        std_adjusted = numpy.maximum(source_standard_deviation, 1.0 /
numpy.sqrt(source_image.size))

        whitened_image = numpy.multiply(numpy.subtract(source_image,
source_mean), 1 / std_adjusted)

        return whitened_image



    # 图像预处理————大小、通道顺序、格式调整

    def preprocess_image(src):



        # 大小调整

        NETWORK_WIDTH = 160

        NETWORK_HEIGHT = 160

        # resize
        #print(src)
        preprocessed_image = cv2.resize(src, (NETWORK_WIDTH,
NETWORK_HEIGHT))

        # BGR——>RGB
```

```
        preprocessed_image = cv2.cvtColor(preprocessed_image,
cv2.COLOR_BGR2RGB)
        # 格式调整
        preprocessed_image = whiten_image(preprocessed_image)
        # 返回预处理后的图像
        return preprocessed_image


    # 人脸识别比对（计算欧氏距离，越小越匹配）
    def face_match(face1_output, face2_output):
        if (len(face1_output) != len(face2_output)):
            print('length mismatch in face_match')
            return False
        total_diff = 0
        # 欧氏距离
        for output_index in range(0, len(face1_output)):
            this_diff = numpy.square(face1_output[output_index] -
face2_output[output_index])
            total_diff += this_diff
        print('Total Difference is: ' + str(total_diff))
        # !!! 识别———判别是否是同一人脸 !!!
        if (total_diff < FACE_MATCH_THRESHOLD):
            return True
        # differences between faces was over the threshold above so
```

```
        # they didn't match.
        else:
            return False

    # !（批量）特征提取 !
    def feature(targets_list, temp_list, graph):

        i = 0
        for target in targets_list :
            # 读取一个目标
            target =
cv2.imread('/home/xilinx/jupyter_notebooks/ncs_facenet/targets/' + target)
            # 目标特征集合（字典）
            #print(target)
            temp_list[i] = run_inference(target, graph)
            i = i + 1
        # 返回特征集合
        return temp_list

    # !!! 人脸识别（顶层模块）!!!
    def run_images(targets_feature, targets_list, graph, input_frame1, input_frame2):

        global matching,v,text_color,rectangle_color
```

```
        global score_list
        score_list=[]

        rect_width = 10
        offset = int(rect_width/2)

        l = len(targets_feature)
        f = 0
        i = 0

        # ！待检测图像特征提取 ！
        input_feature = run_inference(input_frame2, graph)

        text_to_return=""
        # !!! 人脸识别 —— 遍历目标特征集合（字典），循环比对 !!!
    #       for target_feature in targets_feature :
        for i in range(l):
            # 匹配
            total_diff = 0
            # 欧氏距离
            for output_index in range(0, len(targets_feature[i])):
                this_diff = numpy.square(targets_feature[i][output_index] -
input_feature[output_index])
```

```
                total_diff += this_diff

            print('Total Difference is: ' + str(total_diff))
            score_list.append(total_diff)

        # !!! 识别———判别是否是同一人脸 !!!


        #遍历得分表，找最小的索引
        least_index=0
        index=0
        least_score=100
        for score in score_list:
            if least_score>score:
                least_index=index
                least_score=score
            index=index+1

        matching=(least_score<0.3)


        if (matching):

            n = len(targets_list[least_index])

            # match, green rectangle

            text_to_return=targets_list[least_index][:n - 4]


        else:


            text_to_return="stranger"


        return text_to_return
import bnn
# 导入 NCS—SK—API
from mvnc import mvncapi as mvnc
# 导入 facenet 模块包
#import facenet_ncs
import cv2
import sys
```

```
import os
import numpy as np
from pynq.overlays.base import BaseOverlay
from pynq.lib.video import *
from time import sleep
import realtime_input
import serial
ser=serial.Serial('/dev/ttyUSB0',115200,timeout=0.5)
state=1
print(bnn.available_params(bnn.NETWORK_CNVW1A1))
classifier =
bnn.CnvClassifier(bnn.NETWORK_CNVW1A1,"streetview",bnn.RUNTIME_HW)
current_class="streetview"
print(classifier.classes)


cap = cv2.VideoCapture(0)
cap.set(3,320)#设置摄像头输出宽
cap.set(4,240)#设置摄像头输出高
print("start reading video...")
print("Capture device is open: " + str(cap.isOpened()))
if True:
    # !!! facenet 模型文件路径 ——— facenet.graph !!!
    GRAPH_FILENAME = "./facenet_celeb_ncs.graph"
    # 人脸录入路径
    targets_list_dir = './targets'
    mvnc.SetGlobalOption(mvnc.GlobalOption.LOG_LEVEL, 2)
    devices = mvnc.EnumerateDevices()
    if len(devices) == 0:
    # No NCS devices found
        print('No NCS devices found')
    else:
        # Try connecting to the first NCS device found
        device = mvnc.Device(devices[0])
        device.OpenDevice()
    # 准备模型文件(加载至片上内存)——facenet.graph
    graph_file_name = GRAPH_FILENAME
    with open(graph_file_name, mode='rb') as f:
        graph_in_memory = f.read()
    graph = device.AllocateGraph(graph_in_memory)
    # 获取人脸目标路径
    temp_list = {}
    targets_list = os.listdir(targets_list_dir)
```

```
    targets_list = [i for i in targets_list if i.endswith('.jpg')]

    targets_feature = feature(targets_list, temp_list, graph)



#从摄像头取图
# -*- coding:utf-8 -*-
import numpy as np
import imutils
import socket
from PIL import Image
import threading
import time
import math
res=''
state=1
name_from_server=''
watch_cascade = cv2.CascadeClassifier('cascade.xml')
def udp_receive(skt):
    global state
    global current_class
    global classifier
    global res
    while True:
        response, addr = skt.recvfrom(1024)
        res=response.decode()
        #print(res)

arg1=0.1

def deal_license(licenseimg):

    ret, thresh = cv2.threshold(licenseimg, 100, 255, cv2.THRESH_BINARY)

    return thresh


def find_end(start, arg, black, white, width, black_max, white_max):
    end = start + 1
    for m in range(start + 1, width - 1):
        if (black[m] ) > ((1-arg1)* black_max ):
            end = m
            break
```

```
        return end


    # 检测图像中的凸点(手指)个数
    def _get_contours(array):
        # 利用 findContours 检测图像中的轮廓，其中返回值 contours 包含了图像中所有轮廓的坐标点
        _, contours, _ = cv2.findContours(array, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
        return contours


    # 根据图像中凹凸点中的 (开始点，结束点，远点)的坐标，利用余弦定理计算两根手指之间的夹角，其必为锐角，根据锐角的个数判别手势.
    def _get_defects_count(array, contour, defects, verbose=False):
        ndefects = 0

        for i in range(defects.shape[0]):
            s, e, f, _ = defects[i, 0]
            beg = tuple(contour[s][0])
            end = tuple(contour[e][0])
            far = tuple(contour[f][0])
            a = _get_eucledian_distance(beg, end)
            b = _get_eucledian_distance(beg, far)
            c = _get_eucledian_distance(end, far)
            angle = math.acos((b ** 2 + c ** 2 - a ** 2) / (2 * b * c))    # * 57

            if angle <= math.pi / 2:    # 90:
                ndefects = ndefects + 1

                if verbose:
                    cv2.circle(array, far, 3, _COLOR_RED, -1)

            if verbose:
                cv2.line(array, beg, end, _COLOR_RED, 1)

        return array, ndefects


    def grdetect(array, verbose=False):
        copy = array.copy()
        array = _remove_background(array)    # 移除背景, add by wnavy
        thresh = _bodyskin_detetc(array)
```

```
            contours = _get_contours(thresh.copy())  # 计算图像的轮廓
            largecont = max(contours, key=lambda contour: cv2.contourArea(contour))

            hull = cv2.convexHull(largecont, returnPoints=False)  # 计算轮廓的凸点
            defects = cv2.convexityDefects(largecont, hull)  # 计算轮廓的凹点

            if defects is not None:
                # 利用凹陷点坐标，根据余弦定理计算图像中锐角个数
                copy, ndefects = _get_defects_count(copy, largecont, defects,
verbose=verbose)

                # 根据锐角个数判断手势，会有一定的误差



                return 1

        def _get_eucledian_distance(beg, end):#计算两点之间的坐标
            i=str(beg).split(',')
            j=i[0].split('(')
            x1=int(j[1])
            k=i[1].split(')')
            y1=int(k[0])
            i=str(end).split(',')
            j=i[0].split('(')
            x2=int(j[1])
            k=i[1].split(')')
            y2=int(k[0])
            d=math.sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2))
            return d




        Lower = np.array([130, 75, 0])
        Upper = np.array([175, 130, 18])


        circle_x=0
        circle_y=0
        circle_r=0
```

```
number_send_buffer=[]
number_cnt_buffer=[]
number_six_cnt=0
number_seven_cnt=0

number_four_cnt=0
number_five_cnt=0
number_eight_cnt=0
number_nine_cnt=0
number_cnt=0

# Black
Lowerb = np.array([0, 0, 0])
# 55
Upperb = np.array([180, 255, 130])



kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (9, 1))

kernel1 = cv2.getStructuringElement(cv2.MORPH_RECT, (1, 9))



#0 模式识别灯牌
#1 模式识别数字
#2 模式识别人脸
color = (255, 250, 87)
ss = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
num = 0
last_number='0'
font = cv2.FONT_HERSHEY_SIMPLEX
face_cascade = cv2.CascadeClassifier(
    '/home/xilinx/jupyter_notebooks/base/video/data/'
    'haarcascade_frontalface_default.xml')
t=threading.Thread(target=udp_receive,args=(ss,))
t.start()
while(True):
    images=[]
    _ , cv2_im = cap.read()
    #cv2_im = cv2.cvtColor(cv2_im,cv2.COLOR_BGR2RGB)
    #保存原始大小的图像
    img1 = cv2_im.copy()
```

```
            startt =time.clock()
            cv2_im=imutils.resize(cv2_im, width=160)
            if state==1:


                cv2_im=cv2.blur(cv2_im,(3,3))
                gray = cv2.cvtColor(cv2_im, cv2.COLOR_BGR2GRAY)

                circles =
cv2.HoughCircles(gray,cv2.HOUGH_GRADIENT,1,50,param1=100,param2=45,min
Radius=20,maxRadius=50)
                if circles is not None:
                    for i in circles[0,:]:
                    # draw the outer circle


                        region = gray[int(i[1] -    i[2]):int(i[1] + i[2]), int(i[0] -
i[2]):int(i[0] +i[2])]
                        if region.shape[0]>20 and region.shape[1]>20:
                        #region= imutils.resize(region, width=28)
                            img_name = 'test' + '%d.jpg ' % (num)
                            num += 1
                            x=int(i[0]-i[2])
                            y=int(i[1]-i[2])
                            w=2*int(i[2])
                            h=int(i[2])
                            image1=Image.fromarray(region)
                            result_class_idx = classifier.classify_image(image1)
                            #print("Inferred number:
{0}".format(classifier.class_name(result_class_idx)))
                            last_number=classifier.class_name(result_class_idx)
                            circle_x=int(i[0])
                            circle_y=int(i[1])
                            circle_r=int(i[2])
                            cv2.circle(img1, ((int)(2*i[0]),(int)(2* i[1])),
(int)(2*i[2]), (255, 0, 0), 2)
                            cv2.circle(img1, ((int)(2*i[0]),(int)(2*
i[1])),2,(255,0,0),3)
                            cv2.putText(img1, last_number, ((int)(2*i[0]),(int)(2*
i[1]-(int)(2*i[2]))), font, 1, (255, 0, 255), 4)


#cv2.rectangle(img1,(2*x,2*y),(2*(x+w),2*(y+h)),color,4)
                            #cv2.rectangle(img1,(2*x-2,2*y-
```

```
20),(2*(x+w+1),2*(y+0)),color,-1)
                sum=circle_x+circle_y+circle_r
                number_cnt=number_cnt+1
                if last_number=='4':
                    number_four_cnt=number_four_cnt+1
                if last_number=='5':
                    number_five_cnt=number_five_cnt+1
                if last_number=='8':
                    number_eight_cnt=number_eight_cnt+1
                if last_number=='9':
                    number_nine_cnt=number_nine_cnt+1

                if last_number=='6':
                    number_six_cnt=number_six_cnt+1
                if last_number=='7':
                    number_seven_cnt=number_seven_cnt+1

                if number_cnt==10:
                    number_cnt=0
                    if number_six_cnt>6:
                            Uart_buf = bytearray([0x55,0x06, circle_x,
circle_y,circle_r,sum & 0x00ff, 0x00,0xAA])
                            ser.write(Uart_buf)
                    if number_seven_cnt>6:
                            Uart_buf = bytearray([0x55,0x07, circle_x,
circle_y,circle_r,sum & 0x00ff, 0x00,0xAA])
                            ser.write(Uart_buf)
                    if number_four_cnt>6:
                            Uart_buf = bytearray([0x55,0x04, circle_x,
circle_y,circle_r,sum & 0x00ff, 0x00,0xAA])
                            ser.write(Uart_buf)
                    if number_five_cnt>6:
                            Uart_buf = bytearray([0x55,0x05, circle_x,
circle_y,circle_r,sum & 0x00ff, 0x00,0xAA])
                            ser.write(Uart_buf)
                    if number_eight_cnt>6:
                            Uart_buf = bytearray([0x55,0x08, circle_x,
circle_y,circle_r,sum & 0x00ff, 0x00,0xAA])
                            ser.write(Uart_buf)
                    if number_nine_cnt>6:
                            Uart_buf = bytearray([0x55,0x09, circle_x,
circle_y,circle_r,sum & 0x00ff, 0x00,0xAA])
                            ser.write(Uart_buf)
                    number_six_cnt=0
```

```
                    number_seven_cnt=0
                    number_four_cnt=0
                    number_five_cnt=0
                    number_eight_cnt=0
                    number_nine_cnt=0



                '''
                Uart_buf = bytearray(
                    [0x55,0x02, circle_x, circle_y,circle_r,sum & 0x00ff,
0x00,0xAA])
                if last_number=='6':
                                Uart_buf = bytearray(
                    [0x55,0x06, circle_x, circle_y,circle_r,sum & 0x00ff,
0x00,0xAA])
                if last_number=='7':
                                Uart_buf = bytearray(
                    [0x55,0x07, circle_x, circle_y,circle_r,sum & 0x00ff,
0x00,0xAA])


                ser.write(Uart_buf)
                '''

            if state==2:
                gray = cv2.cvtColor(cv2_im, cv2.COLOR_BGR2GRAY)
                faces = face_cascade.detectMultiScale(gray, 1.3, 5)
                for (x,y,w,h) in faces:
                    cv2.rectangle(img1,(2*x,2*y),(2*(x+w),2*(y+h)),color,4)
                    cv2.rectangle(img1,(2*x-2,2*y-20),(2*(x+w+1),2*(y+0)),color,-
1)
                    roi_color = cv2_im[y:y+h, x:x+w]
                    #cv2.rectangle(frame1, (200, 80),(440, 400),(0,255,0), 3)
                    #frame2 = frame1
                    text_returned = run_images(targets_feature, targets_list, graph,
cv2_im, roi_color)
        # HDMI 输出
                #frame_res = cv2.resize(frame_res, (640, 480)) #  输出结果大小调整
                #cv2.imwrite('zxd.jpg',frame1)
                #roi_color=cv2.cvtColor(roi_color, cv2.COLOR_BGR2RGB)
                    cv2.putText(img1, text_returned,(2*x,2*y), font, 0.4, (0, 0, 255),
1)
```

```
        if state==3:

            gray = cv2.cvtColor(cv2_im, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray, 1.3, 5)
            for (x,y,w,h) in faces:
                cv2.rectangle(img1,(2*x,2*y),(2*(x+w),2*(y+h)),(255,0,0),2)
                roi_color = cv2_im[y:y+h, x:x+w]
                cv2.imwrite('/home/xilinx/jupyter_notebooks/ncs_facenet/targets/'
+ name_from_server + '.jpg', roi_color)
                print('save')
            # 获取人脸目标路径
            temp_list = {}
            targets_list = os.listdir(targets_list_dir)
            targets_list = [i for i in targets_list if i.endswith('.jpg')]
            # !!! 人脸录入 !!!（获取目标人脸特征集）
            targets_feature = feature(targets_list, temp_list, graph)
            #cv2.putText(img1, 'Input...OK OK OK', (66, 88),
cv2.FONT_HERSHEY_SIMPLEX, 1, text_color, 4)
            state=2

        if state==6:

            frame=cv2_im
            fuck=cv2.blur(cv2_im,(3,3))
            elapsed = (time.clock() - startt)
            print_str='resize and blur cost: '+str(1000*elapsed)
            print(print_str)
            startt =time.clock()


            ycrcb = cv2.cvtColor(frame, cv2.COLOR_BGR2YCrCb) # 分解为
YUV 图像,得到 CR 分量
            hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)  # 分解为
YUV 图像,得到 CR 分量
            elapsed = (time.clock() - startt)
            print_str='cvt color cost: '+str(1000*elapsed)
            print(print_str)
            startt =time.clock()


            (_, cr1, cb1) = cv2.split(ycrcb)
            (h1,_,_)=cv2.split(hsv)
            crcbh=cv2.merge([cr1,cb1,h1])
```

```
                elapsed = (time.clock() - startt)
                print_str='split cost: '+str(1000*elapsed)
                print(print_str)
                startt =time.clock()
                skin = cv2.inRange(crcbh, Lower, Upper)
                elapsed = (time.clock() - startt)
                print_str='go over cost: '+str(1000*elapsed)
                print(print_str)
                startt =time.clock()
                kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))    #
椭圆结构

                skin = cv2.dilate(skin, kernel, iterations=1)
                fuck = cv2.erode(skin, kernel, iterations=1)




                elapsed = (time.clock() - startt)
                print_str='dilate and erode cost: '+str(1000*elapsed)
                print(print_str)
                startt =time.clock()
                contours = _get_contours(fuck)    # 计算图像的轮廓
                elapsed = (time.clock() - startt)
                print_str='findContours cost: '+str(1000*elapsed)
                print(print_str)
                startt =time.clock()
                if len(contours)>1:
                        contour= max(contours, key = lambda contour:
cv2.contourArea(contour))
                        slist=sorted(contours, key=lambda contour:
cv2.contourArea(contour),reverse=True)
                        contour1=slist[1]
                        x1, y1, w1, h1 = cv2.boundingRect(contour1)
                        # center = (int(x), int(y))
                        cv2.rectangle(img1, (2*x1, 2*y1), (2*(x1 + w1), 2*(y1 + h1)),
(100, 0, 110), 3)
                        x, y, w, h = cv2.boundingRect(contour)
                        # center = (int(x), int(y))
                        cv2.rectangle(img1, (2*x, 2*y), (2*(x + w),2*(y + h)), (100, 100,
0), 3)
                        if x1<x:
                                contour=contour1
                        hull = cv2.convexHull(contour, True, returnPoints=False)    # 获
```

```
得凸包点  x, y 坐标
                    defects = cv2.convexityDefects(contour, hull)    # 计算轮廓的凹
点
                    if defects is not None:    # 重要!
                        ndefects = 0
                        for i in range(defects.shape[0]):
                            s, e, f, d = defects[i, 0]
                            start = tuple(contour[s][0])    # 起点
                            end = tuple(contour[e][0])    # 终点
                            far = tuple(contour[f][0])    # 最远点
                            a = _get_eucledian_distance(start, end)
                            b = _get_eucledian_distance(start, far)
                            c = _get_eucledian_distance(end, far)
                            angle = math.acos((b ** 2 + c ** 2 - a ** 2) / (2 * b *
c))
                            cv2.line(img1, start, far, [255, 255, 0], 2)
                            cv2.line(img1, end, far, [255, 255, 0], 2)
                            #cv2.line(frame, start, end, [0, 255, 0], 2)
                            #cv2.circle(frame, far, 5, [0, 0, 255], -1)
                            angle = math.acos((b ** 2 + c ** 2 - a ** 2) / (2 * b *
c))    # * 57
                            if angle <= math.pi / 2:    # 90:
                                ndefects = ndefects + 1
                    cv2.putText(img1, str(ndefects+1), (50,50),
cv2.FONT_HERSHEY_COMPLEX, 2, (0, 255, 0), 2)

            if state==5:
                image_gray = cv2.cvtColor(cv2_im, cv2.COLOR_RGB2GRAY)
                watches = watch_cascade.detectMultiScale(image_gray, 1.1, 2,
minSize=(36, 9), maxSize=(36 * 40, 9 * 40))
                text=''
                #print("检测到车牌数", len(watches))

                for (x, y, w, h) in watches:
                    cv2.rectangle(img1, (2*x, 2*y), (2*(x + w), 2*(y + h)), (0, 0,
255), 1)

                    thresh = image_gray[y:y+h,x:x+w]
                    thresh_cpy = image_gray[y:y+h,x:x+w]
                    heightt=thresh.shape[0]    # 263

                    thresh = imutils.resize(thresh, width=int(0.5*thresh.shape[1]))

                    thresh = deal_license(thresh)
```

```
    # 记录黑白像素总和
        white = []
        black = []
        height = thresh.shape[0]    # 263
        width = thresh.shape[1]    # 400
        white_max = 0
        black_max = 0
        # 计算每一列的黑白像素总和
        for i in range(width):
            line_white = 0
            line_black = 0
            for j in range(height):
                if thresh[j][i] == 255:
                    line_white += 1
                if thresh[j][i] == 0:
                    line_black += 1
            white_max = max(white_max, line_white)
            black_max = max(black_max, line_black)
            white.append(line_white)
            black.append(line_black)
        arg = True
        if black_max < white_max:
            arg = False

        n = 1
        start = 1
        end = 2


        while n < width - 2:
            n += 1
            # 判断是白底黑字还是黑底白字    0.05 参数对应上面的
0.95 可作调整
            if (white[n]) > (arg1 * white_max):

                start = n
                end = find_end(start, arg, black, white, width,
black_max, white_max)
```

```
                                if end - start > 2:

                                    cj = thresh_cpy[1:heightt, 2*start:2*end]

                                    diff = heightt - 2*(end - start + 1)
                                    if diff>0:
                                        cj = cv2.copyMakeBorder(cj, 0, 0, int(0.5 *
diff), int(0.5 * diff), cv2.BORDER_CONSTANT, (0, 0, 0))
                                    #cv2.imwrite('./car_signs/'+str(n)+'.jpg',cj)

                                    image1=Image.fromarray(cj)
                                    images.append(image1)
                                #result_class_idx = classifier.classify_image(image1)
                                #print("Inferred number:
{0}".format(classifier.class_name(result_class_idx)))
                                #last_number=classifier.class_name(result_class_idx)


                                #print(last_number)
                                #cv2.putText(img1, str(last_number),
((int)(2*i[0]),(int)(2* i[1]-(int)(2*i[2]))), font, 1, (255, 0, 255), 4)
                                #text+=last_number


                                n = end
            if len(images)>0:
                result_class_array = classifier.classify_images(images)
                for num in result_class_array:
                    text+=classifier.class_name(num)


            if '9802' in text:
                cv2.putText(img1, '98022',(20,40), font, 1, (0, 0, 255), 4)
            if '3277' in text:
                cv2.putText(img1, '32770',(20,40), font, 1, (0, 0, 255), 4)
            if '7528' in text:
                cv2.putText(img1, '75282',(20,40), font, 1, (0, 0, 255), 4)
            if '0792' in text:
                cv2.putText(img1, '07921',(20,40), font, 1, (0, 0, 255), 4)
            if '2547' in text:
                cv2.putText(img1, '25477',(20,40), font, 1, (0, 0, 255), 4)


        if state==9:
```

```
                cv2_im=img1.copy()
                cv2_im=cv2.blur(cv2_im,(3,3))
                gray = cv2.cvtColor(cv2_im, cv2.COLOR_BGR2GRAY)

                images=[]
                circles =
cv2.HoughCircles(gray,cv2.HOUGH_GRADIENT,1,50,param1=100,param2=45,min
Radius=20,maxRadius=50)
                if circles is not None:
                    for i in circles[0,:]:
                    # draw the outer circle


                        region = gray[int(i[1] -    i[2]):int(i[1] + i[2]), int(i[0] -
i[2]):int(i[0] +i[2])]
                        if region.shape[0]>5 and region.shape[1]>5:
                        #region= imutils.resize(region, width=28)
                            img_name = 'test' + '%d.jpg ' % (num)
                            num += 1
                            x=int(i[0]-i[2])
                            y=int(i[1]-i[2])
                            w=2*int(i[2])
                            h=int(i[2])
                            #image1=Image.fromarray(region)
                            #images.append(image1)
                            #result_class_idx = classifier.classify_image(image1)
                            #print("Inferred number:
{0}".format(classifier.class_name(result_class_idx)))
                            #last_number=classifier.class_name(result_class_idx)
                            circle_x=int(i[0])
                            circle_y=int(i[1])
                            circle_r=int(i[2])
                            cv2.circle(img1, ((int)(1*i[0]),(int)(1* i[1])),
(int)(1*i[2]), (255, 0, 0), 2)
                            cv2.circle(img1, ((int)(1*i[0]),(int)(1*
i[1])),2,(255,0,0),3)
                            #cv2.putText(img1, last_number, ((int)(2*i[0]),(int)(2*
i[1]-(int)(2*i[2]))), font, 1, (255, 0, 255), 4)


#cv2.rectangle(img1,(2*x,2*y),(2*(x+w),2*(y+h)),color,4)
                            #cv2.rectangle(img1,(2*x-2,2*y-
20),(2*(x+w+1),2*(y+0)),color,-1)
                        #textt="
```

```
                    #result_class_array = classifier.classify_images(images)
                    #for num in result_class_array:
                        #text+=classifier.class_name(num)
                    #print(text)



            if len(res)>0:
                if res[0]=='1':

                    state=1
                    print(res[1:])
                    if current_class!="streetview":

                        classifier =
bnn.CnvClassifier(bnn.NETWORK_CNVW1A1,"streetview",bnn.RUNTIME_HW)

                        current_class="streetview"
                if res=='2':

                    state=2
                if res=='5':

                    state=5
                    if current_class!="streetview":

                        classifier =
bnn.CnvClassifier(bnn.NETWORK_CNVW1A1,"streetview",bnn.RUNTIME_HW)

                        current_class="streetview"


                if res=='4':

                    state=1
                    if current_class!="road-signs":

                        classifier =
bnn.CnvClassifier(bnn.NETWORK_CNVW1A1,"road-signs",bnn.RUNTIME_HW)

                        current_class="road-signs"
                if res=='9':
```

```
                state=9
                if current_class!="road-signs":

                    classifier =
bnn.CnvClassifier(bnn.NETWORK_CNVW1A1,"road-signs",bnn.RUNTIME_HW)

                    current_class="road-signs"
            if res=='6':

                state=6
                if current_class!="road-signs":

                    classifier =
bnn.CnvClassifier(bnn.NETWORK_CNVW1A1,"road-signs",bnn.RUNTIME_HW)

                    current_class="road-signs"
            if res[0]=='3'and state==2:

                state=3
                name_from_server=res[1:]
                #print(name_from_server)









        cv2.putText(img1, str(state),(20,200), font, 1, (0, 0, 255), 1)
        elapsed = (time.clock() - startt)
        cv2.putText(img1, str(1.0/elapsed),(200,200), font, 0.5, (0, 0, 255), 1)
        img_encode = cv2.imencode('.jpg', img1)[1]
        data_encode = np.array(img_encode)
        data = data_encode.tostring()
        ss.sendto(data, ('192.168.43.141', 6053))

        cv2.waitKey(1)
```

verilog 代码

```
`timescale 1ns/1ps
module led_control_AXILiteS_s_axi
#(parameter
    C_S_AXI_ADDR_WIDTH = 5,
    C_S_AXI_DATA_WIDTH = 32
)(
    // axi4 lite slave signals
    input   wire                                ACLK,
    input   wire                                ARESET,
    input   wire                                ACLK_EN,
    input   wire [C_S_AXI_ADDR_WIDTH-1:0] AWADDR,
    input   wire                                AWVALID,
    output wire                                AWREADY,
    input   wire [C_S_AXI_DATA_WIDTH-1:0] WDATA,
    input   wire [C_S_AXI_DATA_WIDTH/8-1:0] WSTRB,
    input   wire                                WVALID,
    output wire                                WREADY,
    output wire [1:0]                          BRESP,
    output wire                                BVALID,
    input   wire                                BREADY,
    input   wire [C_S_AXI_ADDR_WIDTH-1:0] ARADDR,
    input   wire                                ARVALID,
    output wire                                ARREADY,
    output wire [C_S_AXI_DATA_WIDTH-1:0] RDATA,
    output wire [1:0]                          RRESP,
    output wire                                RVALID,
    input   wire                                RREADY,
    // user signals
    output wire [31:0]                         total_cnt,
    output wire [31:0]                         high_cnt
);
//------------------------Address Info-------------------
// 0x00 : reserved
// 0x04 : reserved
// 0x08 : reserved
// 0x0c : reserved
// 0x10 : Data signal of total_cnt
//            bit 31~0 - total_cnt[31:0] (Read/Write)
// 0x14 : reserved
// 0x18 : Data signal of high_cnt
//            bit 31~0 - high_cnt[31:0] (Read/Write)
// 0x1c : reserved
// (SC = Self Clear, COR = Clear on Read, TOW = Toggle on Write, COH = Clear on
```

Handshake)

```
//------------------------Parameter----------------------
localparam
        ADDR_TOTAL_CNT_DATA_0 = 5'h10,
        ADDR_TOTAL_CNT_CTRL     = 5'h14,
        ADDR_HIGH_CNT_DATA_0    = 5'h18,
        ADDR_HIGH_CNT_CTRL      = 5'h1c,
        WRIDLE                          = 2'd0,
        WRDATA                          = 2'd1,
        WRRESP                          = 2'd2,
        WRRESET                         = 2'd3,
        RDIDLE                          = 2'd0,
        RDDATA                          = 2'd1,
        RDRESET                         = 2'd2,
        ADDR_BITS               = 5;

//------------------------Local signal-------------------
        reg    [1:0]                    wstate = WRRESET;
        reg    [1:0]                    wnext;
        reg    [ADDR_BITS-1:0]          waddr;
        wire [31:0]                     wmask;
        wire                            aw_hs;
        wire                            w_hs;
        reg    [1:0]                    rstate = RDRESET;
        reg    [1:0]                    rnext;
        reg    [31:0]                   rdata;
        wire                            ar_hs;
        wire [ADDR_BITS-1:0]            raddr;
        // internal registers
        reg    [31:0]                   int_total_cnt = 'b0;
        reg    [31:0]                   int_high_cnt = 'b0;

//------------------------Instantiation------------------

//------------------------AXI write fsm------------------
assign AWREADY = (wstate == WRIDLE);
assign WREADY    = (wstate == WRDATA);
assign BRESP     = 2'b00;    // OKAY
assign BVALID    = (wstate == WRRESP);
assign wmask     = { {8{WSTRB[3]}}, {8{WSTRB[2]}}, {8{WSTRB[1]}},
{8{WSTRB[0]}} };
assign aw_hs     = AWVALID & AWREADY;
assign w_hs      = WVALID & WREADY;
```

```
// wstate
always @(posedge ACLK) begin
    if (ARESET)
        wstate <= WRRESET;
    else if (ACLK_EN)
        wstate <= wnext;
end

// wnext
always @(*) begin
    case (wstate)
        WRIDLE:
            if (AWVALID)
                wnext = WRDATA;
            else
                wnext = WRIDLE;
        WRDATA:
            if (WVALID)
                wnext = WRRESP;
            else
                wnext = WRDATA;
        WRRESP:
            if (BREADY)
                wnext = WRIDLE;
            else
                wnext = WRRESP;
        default:
            wnext = WRIDLE;
    endcase
end

// waddr
always @(posedge ACLK) begin
    if (ACLK_EN) begin
        if (aw_hs)
            waddr <= AWADDR[ADDR_BITS-1:0];
    end
end

//------------------------AXI read fsm-------------------
assign ARREADY = (rstate == RDIDLE);
assign RDATA   = rdata;
assign RRESP   = 2'b00;  // OKAY
```

```
assign RVALID    = (rstate == RDDATA);
assign ar_hs     = ARVALID & ARREADY;
assign raddr     = ARADDR[ADDR_BITS-1:0];

// rstate
always @(posedge ACLK) begin
    if (ARESET)
        rstate <= RDRESET;
    else if (ACLK_EN)
        rstate <= rnext;
end

// rnext
always @(*) begin
    case (rstate)
        RDIDLE:
            if (ARVALID)
                rnext = RDDATA;
            else
                rnext = RDIDLE;
        RDDATA:
            if (RREADY & RVALID)
                rnext = RDIDLE;
            else
                rnext = RDDATA;
        default:
            rnext = RDIDLE;
    endcase
end

// rdata
always @(posedge ACLK) begin
    if (ACLK_EN) begin
        if (ar_hs) begin
            rdata <= 1'b0;
            case (raddr)
                ADDR_TOTAL_CNT_DATA_0: begin
                    rdata <= int_total_cnt[31:0];
                end
                ADDR_HIGH_CNT_DATA_0: begin
                    rdata <= int_high_cnt[31:0];
                end
            endcase
        end
```

```
        end
end


//------------------------Register logic-----------------
assign total_cnt = int_total_cnt;
assign high_cnt    = int_high_cnt;
// int_total_cnt[31:0]
always @(posedge ACLK) begin
        if (ARESET)
                int_total_cnt[31:0] <= 0;
        else if (ACLK_EN) begin
                if (w_hs && waddr == ADDR_TOTAL_CNT_DATA_0)
                        int_total_cnt[31:0] <= (WDATA[31:0] & wmask) | (int_total_cnt[31:0]
& ~wmask);
        end
end

// int_high_cnt[31:0]
always @(posedge ACLK) begin
        if (ARESET)
                int_high_cnt[31:0] <= 0;
        else if (ACLK_EN) begin
                if (w_hs && waddr == ADDR_HIGH_CNT_DATA_0)
                        int_high_cnt[31:0] <= (WDATA[31:0] & wmask) | (int_high_cnt[31:0]
& ~wmask);
        end
end


//------------------------Memory logic-------------------

endmodule
```