嵌入式系统分析与设计大作业

智能跟脸摄像头

大连理工大学

Dalian University of Technology

目　　录

引　　言

该系统基于搭载 Openeular Embedded 操作系统的树莓派 4B，该嵌入式 Linux 编译时加入了 python3、opencv、pip 等子包。使用两个 SG90 舵机进行摄像头的两轴移动，使用 opencv 进行人脸识别和定位，通过开环 PID 控制可以实现摄像头的实时跟脸操作，并使用 flask 推流。

需要使用的相关软硬件设备包括：

（1） 硬件：树莓派 4B 开发板、SG90 舵机×2，摄像头支架，CSI 摄像头，杜邦线

（2） 软件：WSL，MobaXterm，OpenEular Embedded

1 使用开源项目

1.1 openCV

OpenCV 是一个开源计算机视觉库，用于进行图像处理、计算机视觉和机器学习等任务。在本项目中，OpenCV 用于实现人脸检测和图像处理功能。OpenCV 通过提供丰富的函数和算法来实现图像处理任务。在人脸检测中，使用了预训练的分类器（lbpcascade_frontalface_improved.xml）来检测图像中的人脸。无需电路介绍，OpenCV 主要作为软件模块使用。OpenCV 的相关代码已在第 5.1 节中给出，主要包括人脸检测、图像处理等功能。

需注意交叉编译形成镜像时需要使用下面指令打包 opencv 软件包：

bitbake opencv

1.2 Flask

Flask 是一个轻量级的 Web 应用框架，用于构建 Web 应用程序。在本项目中，Flask 用于创建一个 Web 服务器，以实现图像流的传输和展示。Flask 通过定义路由和视图函数来处理 HTTP 请求，并通过模板引擎渲染动态内容。在本项目中，Flask 用于创建一个简单的 Web 页面，并通过视频流展示经过处理的图像。无需电路介绍，Flask 主要用于 Web 网页的搭建。Flask 的相关代码已在第 5.1 节中给出，主要包括 Web 应用的路由、页面渲染等功能。

需注意交叉编译形成镜像时需要使用下面指令打包 flask 和 python 软件包：

bitbake python-pip

bitbake python3

pip flask

1.3 RPi.GPIO

RPi.GPIO 是树莓派上用于控制 GPIO 引脚的 Python 库。在本项目中，RPi.GPIO 用于控制舵机的角度。RPi.GPIO 通过设置 GPIO 引脚的状态和使用 PWM 控制舵机的转动。在本项目中，RPi.GPIO 用于控制底部和顶部舵机的角度。RPi.GPIO 需要连接到树莓派的 GPIO.2 和 GPIO.3 引脚，分别用于控制左右转的舵机和上下转的舵机。

需注意使用 pip 安装 RPi.GPIO 包

pip RPi.GPIO

2 最终设计方法、实现效果

2.1 设计方法

本项目是一个基于树莓派（Raspberry Pi）的人脸追踪系统，通过摄像头捕捉图像中的人脸，并通过舵机控制系统调整云台的角度，使得摄像头能够持续跟随人脸运动。

2.1.1 代码结构

1.引用库和模块导入：代码一开始导入了所需的库和模块，包括 OpenCV（cv2）、时间相关库、NumPy、RPi.GPIO 用于树莓派的 GPIO 控制、Flask 用于 Web 应用开发等。

2.舵机初始化设置：通过设置 GPIO 引脚和 PWM（脉冲宽度调制）方式，对两个舵机进行初始化，其中 p1 和 p2 分别代表两个舵机的 PWM 对象。同时，定义了一些与舵机控制相关的参数，如初始角度、比例系数等。

3.Flask 应用设置：使用 Flask 搭建 Web 应用，定义了两个路由（index 和 video_feed），分别用于渲染主页和生成视频流。通过 generate_frames 函数实时处理图像并返回视频流。

4.图像处理函数：process 函数通过 OpenCV 捕获摄像头图像，检测人脸，并进行一系列的舵机角度控制。其中，btm_servo_control 和 top_servo_control 函数通过开环控制计算舵机要转动的角度。

5.舵机控制函数：set_btm_servo_angle 和 set_top_servo_angle 分别用于设置云台底部和顶部舵机的角度，限制在设定的最小和最大旋转角度范围内。

6.Flask 路由和页面渲染：定义了两个路由，其中 index 路由渲染主页，video_feed 路由用于生成视频流。通过 Flask 的 render_template 函数加载 HTML 页面。

2.1.2 舵机控制策略

通过计算人脸在画面中的水平偏移量，利用开环控制策略计算底部舵机应该旋转的角度，以实现跟随人脸运动。

2.1.3 视频流生成

通过 Flask 应用的 generate_frames 函数，不断处理图像并将其转换为 JPEG 格式的视频流，实现实时展示摄像头捕捉到的图像。

2.2 代码呈现

```
'''
Author          陈佳辉 1946847867@qq.com
Date            2023-11-22 13:08:46
LastEditTime    2023-11-23 17:58:48
```

Description

'''

```
import cv2
import time
import numpy as np
import RPi.GPIO as GPIO
from time import sleep
from flask import Flask, render_template, Response

# 舵机的 GPIO 引脚配置
servopin1 = 2   # 左右转
servopin2 = 3   # 上下转

# 设置 GPIO 模式
GPIO.setmode(GPIO.BCM)
GPIO.setup(servopin1, GPIO.OUT, initial=False)
GPIO.setup(servopin2, GPIO.OUT, initial=False)

# 为舵机设置 PWM
p1 = GPIO.PWM(servopin1, 50)
p2 = GPIO.PWM(servopin2, 50)

# 舵机控制的初始设置
last_btm_degree = 60
last_top_degree = 50
btm_kp = 2.5
top_kp = -1
offset_dead_block = 0.2
FaceCascade = cv2.CascadeClassifier('lbpcascade_frontalface_improved.xml')

# Flask 应用设置
app = Flask(__name__)
cap = cv2.VideoCapture(0)
```

```
cap.set(320, 240)
def tonum(num):
    '''
    用于处理角度转换的函数
    '''
    fm = 10.0 / 180.0
    num = num * fm + 2.5
    num = int(num * 10) / 10.0
    return num

def set_btm_degree(degrees):
    p1.start(tonum(degrees)) #
    time.sleep(0.1)
    p1.ChangeDutyCycle(0) #清除当前占空比，使舵机停止抖动
    time.sleep(0.01)


def set_top_degree(degrees):
    p2.start(tonum(degrees)) #
    time.sleep(0.1)
    p2.ChangeDutyCycle(0) #清除当前占空比，使舵机停止抖动
    time.sleep(0.01)

def btm_servo_control(offset_x):
    '''
    底部舵机的比例控制
    这里舵机使用开环控制
    '''
    global offset_dead_block #  偏移量死区大小
    global btm_kp #  控制舵机旋转的比例系数
    global last_btm_degree #  上一次底部舵机的角度

    #  设置最小阈值
    if abs(offset_x) < offset_dead_block:
        offset_x = 0
```

```
    # offset 范围在-50 到 50 左右
    delta_degree = offset_x * btm_kp
    # 计算得到新的底部舵机角度
    next_btm_degree = last_btm_degree + delta_degree
    # 添加边界检测
    if next_btm_degree < 0:
        next_btm_degree = 0
    elif next_btm_degree > 180:
        next_btm_degree = 180

    return int(next_btm_degree)


def top_servo_control(offset_y):
    '''
    顶部舵机的比例控制
    这里舵机使用开环控制
    '''
    global offset_dead_block
    global top_kp # 控制舵机旋转的比例系数
    global last_top_degree # 上一次顶部舵机的角度

    # 如果偏移量小于阈值就不相应
    if abs(offset_y) < offset_dead_block:
        offset_y = 0

    # offset_y *= -1
    # offset 范围在-50 到 50 左右
    delta_degree = offset_y * top_kp
    # 新的顶部舵机角度
    next_top_degree = last_top_degree + delta_degree
    # 添加边界检测
    if next_top_degree < 0:
        next_top_degree = 0
```

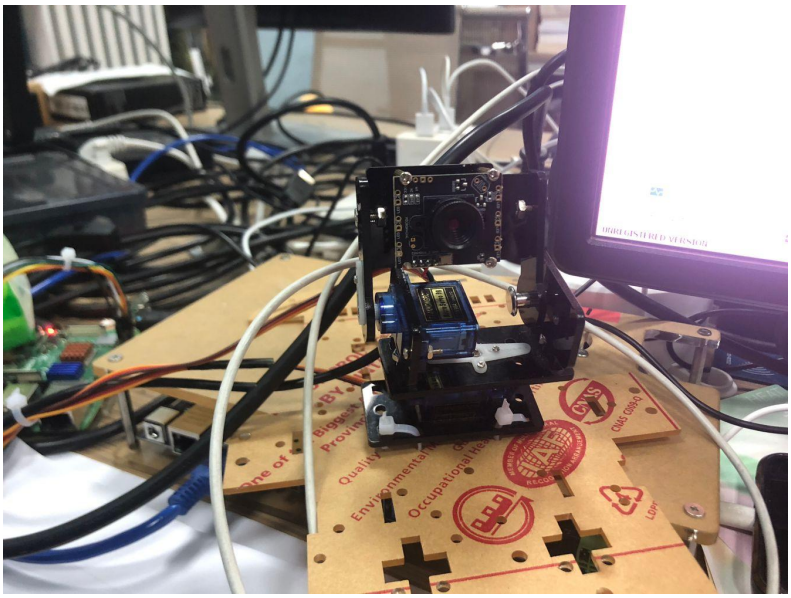
```
    elif next_top_degree > 180:
        next_top_degree = 180

    return int(next_top_degree)

def generate_frames():
    while True:
        processed_frame = process()
        ret, buffer = cv2.imencode('.jpg', processed_frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

def process():
    global last_top_degree
    global last_btm_degree

    ret, img = cap.read()

    # 手机画面水平翻转
    img = cv2.flip(img, 0)
    # 将彩色图片转换为灰度图
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # 检测画面中的人脸
    faces = FaceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5
    )
    # 人脸过滤
    face = face_filter(faces)
    if face is not None:
        # 当前画面有人脸
        (x, y, w, h) = face
        # 在原彩图上绘制矩形
```

```
            cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 4)

            img_height, img_width,_ = img.shape
            print("img h:{} w:{}".format(img_height, img_width))
            # 计算 x 轴与 y 轴的偏移量
            (offset_x, offset_y) = calculate_offset(img_width, img_height, face)
            # 计算下一步舵机要转的角度
            next_btm_degree = btm_servo_control(offset_x)
            next_top_degree = top_servo_control(offset_y)
            # 舵机转动
            #set_cloud_platform_degree(next_btm_degree, next_top_degree)
            set_btm_servo_angle(next_btm_degree)
            set_top_servo_angle(next_top_degree)
            # 更新角度值
            last_btm_degree = next_btm_degree
            last_top_degree = next_top_degree
            print("X 轴偏移量:{} Y 轴偏移量:{}".format(offset_x, offset_y))
            print('底部角度: {}  顶部角度:{}'.format(next_btm_degree, next_top_degree))

    return img

def face_filter(faces):
    '''
    对人脸进行一个过滤
    '''
    if len(faces) == 0:
        return None

    # 目前找的是画面中面积最大的人脸
    max_face =   max(faces, key=lambda face: face[2]*face[3])
    (x, y, w, h) = max_face
    if w < 10 or h < 10:
        return None
    return max_face
```

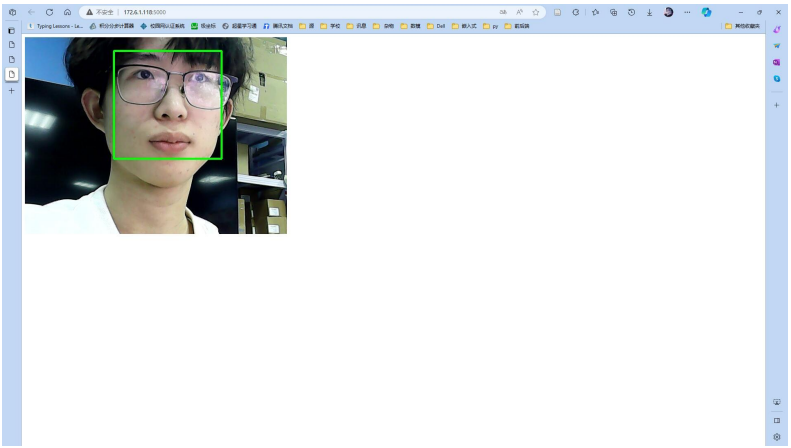
```
def calculate_offset(img_width, img_height, face):
    '''
    计算人脸在画面中的偏移量
    偏移量的取值范围: [-1, 1]
    '''
    (x, y, w, h) = face
    face_x = float(x + w/2.0)
    face_y = float(y + h/2.0)
    # 人脸在画面中心 X 轴上的偏移量
    offset_x = float(face_x / img_width - 0.5) * 2
    # 人脸在画面中心 Y 轴上的偏移量
    offset_y = float(face_y / img_height - 0.5) * 2

    return (offset_x, offset_y)


def set_btm_servo_angle(degree):

        '''
        设定云台底部舵机的角度
        '''

        btm_min_angle = 0 # 底部舵机最小旋转角度
        btm_max_angle = 180 # 底部舵机最大旋转角度
        btm_init_angle = 100 # 底部舵机的初始角度

        if degree < btm_min_angle:
            degree = btm_min_angle
        elif degree > btm_max_angle:
            degree = btm_max_angle

        #self.servos.position(self.btm_servo_idx, degrees=degree)

        set_btm_degree(degrees=degree)
```

```
def set_top_servo_angle(degree):

        '''
        设定云台底部舵机的角度
        '''

        top_min_angle = 0 # 底部舵机最小旋转角度
        top_max_angle = 180 # 底部舵机最大旋转角度
        top_init_angle = 100 # 底部舵机的初始角度

        if degree < top_min_angle:
            degree = top_min_angle
        elif degree > top_max_angle:
            degree = top_max_angle

        #self.servos.position(self.btm_servo_idx, degrees=degree)

        set_top_degree(degrees=degree)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/video_feed')
def video_feed():
            return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == "__main__":
    app.run(host='172.6.1.118', port=5000, debug=False)
    GPIO.cleanup()
```

2.3 实现效果



图一 具体结构



图二 效果

2.4 测试文件

将代码使用 MaboXterm 传至树莓派后使用下面代码运行

python myself_videocapture.py