实验报告 7

OpenCV 与摄像头的使用

范皓年 1900012739　信息科学技术学院

2020 年 11 月 27 日

目录

1 实验目的

1. 了解树莓派 CSI 接口摄像头的使用方法
2. 初步了解 opencv 中对图像的处理方法
3. 学习使用 opencv 对图像中物体进行跟踪

2 实验原理

2.1 OpenCV 总说

机器视觉一直是人工智能中研究的热点，本实验将使用 OpenCV 对视频信号进行采集，并对采集到的数据进行初步的处理。

OpenCV（Open Source Computer Vision Library）是一个开源的机器视觉程序库，包含多种常见的机器视觉算法，目前仍然处于积极开发中，不断有新的功能和算法被整合进来。

OpenCV 具有模块化结构，包含不同层次的功能为用户使用。下面是常用的模块列表：

- 核心功能（Core functionality）：定义了基本的数据结构，包括多维矩阵数组和被其他模块使用的基本功能。
- 图像处理（Image processing）：一个图像处理模块，它包括线性和非线性图像滤波，几何图形转化（重置大小，放射和透视变形，通用基本表格重置映射），色彩空间转换，直方图等。
- 视频处理（video）：一个视频处理模块，它包括动作判断，背景弱化和目标跟踪算法。
- 3D 校准（calib3d）：基于多视图的几何算法，平面和立体摄像机校准，对象姿势判断，立体匹配算法，和 3D 元素的重建。
- 平面特征（features2d）：突出的特征判断，特征描述和对特征描述的对比。
- 对象检测（objdetect）：目标和预定义类别实例化的检测（例如：脸、眼睛、杯子、人、汽车等等）。
- 图形接口（highgui）：一个容易使用的用户功能界面。
- 视频输入输出（videoio）：一个容易使用的视频采集和视频解码器。

OpenCV 采用 C++ 语言开发，它还包含一个 Python 语言的绑定，在树莓派上可以直接使用 Python 代码使用 OpenCV 所提供的功能。

例如下面的代码就用来展示图像处理的功能，打开一副保存在电脑上的图像，窗口名称为 Lena，图像名称为 test_set/lena_color_512.tif

```
1 import cv2 # 加载 OpenCV 库
2 img = cv2.imread('test_set/lena_color_512.tif',1) # 打开图像文件
3 cv2.imshow('Lena',img) # 显示图像
4 cv2.waitKey(0) # 等待用户按键
5 cv2.destroyWindow('Lena') # 关闭显示窗口
```

OpenCV 包含的功能模块非常多，这里介绍两个最常用的功能[1]

[1]其它的模块与函数的用法请参考其它书籍或者在线文档。

2.2 颜色空间分解

颜色空间是采用数学的方式表示颜色的方法，常见的有 RGB，HSV 等。RGB 就是采用颜色的红色（R）、绿色（G）和蓝色（B）分量来表示颜色，在 OpenCV 中，缺省的颜色空间是 BGR，也就是 RGB 的另外一种排序方式。HSV 是根据颜色的直观感受来进行分解的表示法，包括色调（H）、饱和度（S）和明度（V）三个分量。由于 HSV 颜色空间更符合人的主观感受，因此在图像处理软件中有很多应用，在机器视觉领域也经常用来对色彩进行分析。在 OpenCV 中，可以用 cvtColor 函数对颜色空间进行转换，然后使用 inRange 对图像中的色彩进行过滤。参考代码如下：

```
import numpy as np
import cv2
cam = cv2.VideoCapture(0) # 初始化摄像头
while ( True ):
    ret, frame = cam.read() # 读取摄像头数据
    hsv=cv2.cvtColor(frame,cv2.COLOR_BGR2HSV) # 转换颜色空间
    # 通过颜色设计模板
    image_mask=cv2.inRange(hsv,np.array([40,50,50]),
         np.array([80,255,255]))
    # 计算输出图像
    output=cv2.bitwise_and(frame,frame,mask=image_mask)
    cv2.imshow('Original',frame) # 显示原始图像
    cv2.imshow('Output',output) # 显示输出图像
    if cv2.waitKey(1) == ord("q"): # 等待按键
        break
cv2.destroyAllWindows()
cam.release()
```

这个代码相当于按照色调、饱和度、明度生成一个滤镜，可以将摄像头读入的视频信息进行滤制。

2.3 形状识别

从图像中识别特定的形状是一种常见的任务，而识别形状的基础是边界的识别。OpenCV 中包含一个 Canny 边界探测器，它的工作原理如下：

1. 通过高斯算法过滤图像噪声
2. 计算图像的梯度（基于 L1 范数或者 L2 范数）
3. 压缩高梯度线的宽度，获得比较细的边界线
4. 两个阈值参数用来选出图像的最终边界：低于 threshold1 的梯度会被排出边界，高于 threshold2 的梯度将包含在最终输出的边界中。而两个阈值之间的点必须与高于 threshold2 的点相连才被包含在输出中。

在边界探测的基础上就可以识别形状了，例如 cv2.HoughCircles() 用来识别图像中的圆形。它包含如下几个参数：

- 输入图像
- 识别算法，目前只能是 HOUGH_GRADIENT
- dp, 图像分辨率与算法分辨率的倒数

$$dp = \frac{accumulatorresolution}{imageresolution}$$

- minDist, 识别圆心的最小距离
- param1, Canny 算法中的 threshold2
- param2, 圆心识别算法的阈值
- 圆形半径的最小和最大值

下面是一个识别圆形的例子：

```
#!/bin/python
import cv2
cam = cv2.VideoCapture(0) # 打开摄像头
while (True):
ret , frame = cam.read() # 获取摄像头数据
grey = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY) # 色彩变换
blur = cv2.blur(grey,(5,5)) # 过滤噪声
circles = cv2.HoughCircles(blur, # 识别圆形
method=cv2.HOUGH_GRADIENT,dp=1,minDist=200,
param1=100,param2=33,minRadius=30,maxRadius=175)
if circles is not None: # 识别到圆形
for i in circles [0,:]: # 画出识别的结果
cv2.circle(frame,(i[0],i[1]),i[2],(0,255,0),2)
cv2.circle(frame,(i[0],i[1]),2,(0,0,255),3)
cv2.imshow('Detected',frame) # 显示识别图像
if cv2.waitKey(1) == ord("q"): # 等待按键
break
cv2.destroyAllWindows() # 关闭窗口
cam.release() # 释放摄像头
```

这个示例只能实现球形物体的识别，而不能实现物体的跟踪。

2.4 摄像头的使用

OpenCV 有直接对摄像头的支持，前面的示例代码也有对摄像头的使用。例如下面的代码就可以将系统中的摄像头所拍摄的内容显示在窗口中。

```
#!/bin/python
import cv2 # 加载 OpenCV 库
```

```
cam = cv2.VideoCapture(0) # 打开摄像头
cam.set(3, 1024) # 设置图像宽度
cam.set(4, 768) # 设置图像高度
while(True):
    ret, frame = cam.read() # 读入一帧图像
    cv2.imshow('Video Test',frame) # 显示图像
    if cv2.waitKey(1) == ord("q") : # 等待按键
        break
cam.release() # 释放摄像头硬件
cv2.destroyAllWindows() # 关闭全部窗口
```

代码中 cv2.VideoCapture(0) 用来打开系统中的摄像头，一般具有设备文件/dev/video0，如果系统中有多个摄像头，可以改变该函数的参数来选择不同设备。在树莓派中，上面的代码只适用于支持 v4l 驱动的摄像头，对于树莓派 CSI 接口的摄像头（图 7.2），就必须使用不同的方法[2]

树莓派系统包含一个 python-picamera 的软件包用来简化 CSI 接口摄像头的使用。附录A.1中给出了一份代码，可以用以进行摄像头功能的测试。

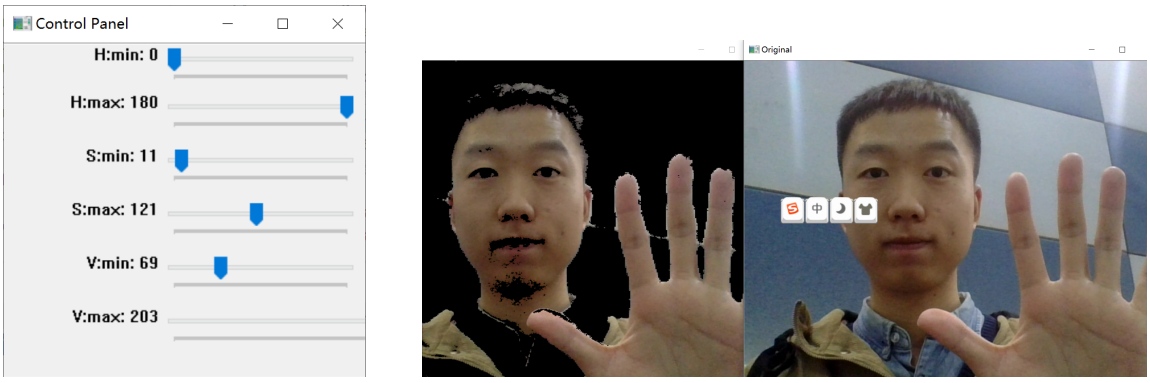
3 实验内容

3.1 摄像头输入测试

采用A.1中的代码进行测试，观察摄像头是否能正常打开。

3.2 色彩识别

在摄像头拍摄范围内将皮肤的颜色过滤出来。通过采用模板的方式，将摄像头拍摄的内容除了皮肤的部分过滤掉。在测试的时候可以用手来做测试，屏幕输出的内容应该只有手的图像。



(a) 调色板　　(b) 滤色方案效果

图 1

考虑到三个量，六对值，每一次调整都重启一次程序，调试成本极高。我们在这里使用了滑动条方式对着三个量进行调整，下面这三个函数分别用于创建窗格，创建一个滑动条，获取滑动条中的值：

[2]通过加载 bcm2835-v4l2 驱动（modprobe bcm2835-v4l2），树莓派也可以采用标准的 v4l 接口使用摄像头。

```
cv2.namedWindow('Control Panel')
cv2.createTrackbar('H:min', 'Control Panel', 0, 180, nothing)
hl = cv2.getTrackbarPos('H:min', 'Control Panel')
```

利用这段代码类似地创建一系列进度条如图1(a) 如图1(b)是创建了一个合理的滤波调色方案，可以滤除肤色相近颜色（卡其色、棕色，包括木质家具）之外的部分。代码见A.2

3.3 形状识别

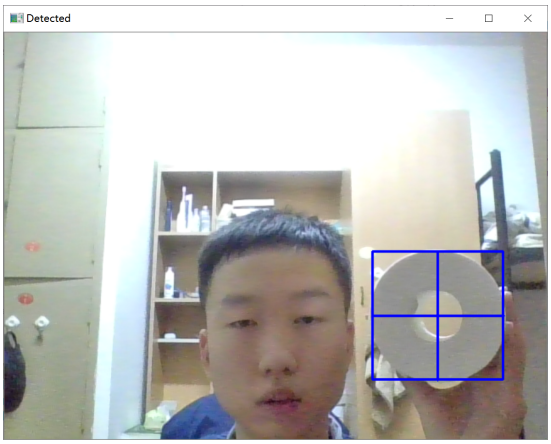
利用示例代码已经能很好地实现圆形的识别（实例代码没有给出）。

但这并不够。由于并不能跟踪，所以视觉识别的稳定性相对较差，锚点在屏幕中四处飘动。

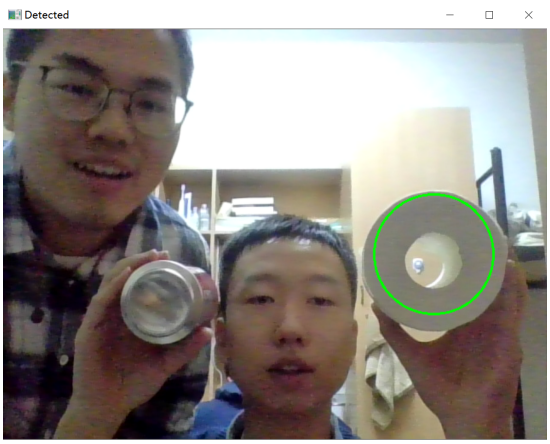
所以我们通过调用 tracker 组件中的 KCF 算法模块进行追踪。调用组件的方法如下：

```
tracker = cv2.TrackerKCF_create() # 创建跟踪对象
tracker.init(frame, initBB) # 锚定图像内容
(success, box) = tracker.update(frame) # 跟踪图像内容
if success:
    (x, y, w, h) = [int(v) for v in box] # 导出跟踪位置
```

选中之后，图像能实现追踪定位，而不会出现中心点闪动和大小不一的情况了。如图2(b)中给出了圆形干扰，但并没有出现识别异常。代码见A.3



(a) 选中图像



(b) 定向跟踪

A 附录代码

代码统一附录于此。没有加行号，可复制粘贴运行。

A.1 cam_test.py

```
# !/bin/python3
# 用于对摄像头进行测试
```

```
import cv2 # 加载 OpenCV 库
cam = cv2.VideoCapture(0) # 打开摄像头
cam.set(3, 1024) # 设置图像宽度
cam.set(4, 768) # 设置图像高度
def cam_clear():
    cam.release() # 释放摄像头硬件
    cv2.destroyAllWindows() # 关闭全部窗口
try:
    while(True):
        ret, frame = cam.read() # 读入一帧图像
        cv2.imshow('Video Test',frame) # 显示图像
        if cv2.waitKey(1) == ord("q") : # 等待按键
            cam_clear()
            break
except KeyboardInterrupt:
    cam_clear()
```

A.2 color_recog.py

```
1  import numpy as np
2  import cv2
3  cam = cv2.VideoCapture(0)  # 初始化摄像头
4
5  # create trackbars for color change
6
7
8  def nothing(sth):
9      pass
10 cv2.namedWindow('Control Panel')
11 cv2.createTrackbar('H:min', 'Control Panel', 0, 180, nothing)
12 cv2.createTrackbar('H:max', 'Control Panel', 180, 180, nothing)
13 cv2.createTrackbar('S:min', 'Control Panel', 0, 255, nothing)
14 cv2.createTrackbar('S:max', 'Control Panel', 255, 255, nothing)
15 cv2.createTrackbar('V:min', 'Control Panel', 0, 255, nothing)
16 cv2.createTrackbar('V:max', 'Control Panel', 255, 255, nothing)
17
18 while (True):
19     ret, frame = cam.read()  # 读取摄像头数据
20     hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)  # 转换颜色空间
21     # 通过颜色设计模板
22     hl = cv2.getTrackbarPos('H:min', 'Control Panel')
```

```
    hh = cv2.getTrackbarPos('H:max', 'Control Panel')
    sl = cv2.getTrackbarPos('S:min', 'Control Panel')
    sh = cv2.getTrackbarPos('S:max', 'Control Panel')
    vl = cv2.getTrackbarPos('V:min', 'Control Panel')
    vh = cv2.getTrackbarPos('V:max', 'Control Panel')
    image_mask = cv2.inRange(hsv, np.array(
        [hl, sl, vl]), np.array([hh, sh, vh]))
    # 计算输出图像
    output = cv2.bitwise_and(frame, frame, mask=image_mask)
    cv2.imshow('Original', frame)  # 显示原始图像
    cv2.imshow('Output', output)  # 显示输出图像
    if cv2.waitKey(1) == ord("q"):  # 等待按键
        break
cam.release()
cv2.destroyAllWindows()
```

A.3 shape_recog.py

```
#!/bin/python
import cv2
from imutils.video import FPS

cam = cv2.VideoCapture(0)  # 打开摄像头
tracker = cv2.TrackerKCF_create()

initBB = None
fps = None

while True:
    ret, frame = cam.read()  # 获取摄像头数据
    grey = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # 色彩变换
    blur = cv2.blur(grey, (5, 5))  # 过滤噪声
    circles = cv2.HoughCircles(blur,  # 识别圆形
                method=cv2.HOUGH_GRADIENT, dp=1, minDist=200,
                param1=100, param2=33, minRadius=30, maxRadius=175)
    # if circles is not None:  # 识别到圆形
    #     for i in circles[0, :]:  # 画出识别的结果
    #         cv2.circle(frame, (i[0], i[1]), int(i[2]), (0, 255, 0), 2)
    #         cv2.circle(frame, (i[0], i[1]), 2, (0, 0, 255), 3)
    if initBB is not None:
        # grab the new bounding box coordinates of the object
```

```
        (success, box) = tracker.update(frame)
        # check to see if the tracking was a success
        if success:
            (x, y, w, h) = [int(v) for v in box]
            # if circles is not None:  # 识别到圆形
            #     target = 0; mindiff = 2147483647
            #     for i in circles[0, :]:  # 画出识别的结果
            #         if abs(i[0] - (x+w//2)) + abs(i[1] - (y+h//2)) < mindiff:
            #             target = i
            #             mindiff = abs(i[0] - (x+w//2)) + abs(i[1] - (y+h//2))
            #     cv2.circle(frame, (target[0], target[1]), int(target[2]), (0, 255,
            #     cv2.circle(frame, (target[0], target[1]), 2, (0, 0, 255), 3)
            cv2.circle(frame, (x + w//2, y + h//2), w//2,(0, 255, 0), 2)
        # update the FPS counter
        fps.update()
        fps.stop()

    key = cv2.waitKey(1) & 0xFF
    if key == ord("s"):
        initBB = cv2.selectROI("Detected", frame, fromCenter=False,
                               showCrosshair=True)
        tracker.init(frame, initBB)
        fps = FPS().start()
    cv2.imshow('Detected', frame)  # 显示识别图像
    if key == ord("q"):  # 等待按键
        break
cv2.destroyAllWindows()  # 关闭窗口
cam.release()  # 释放摄像头
```

A.4 hand_recog.py

```
#!/bin/python
import cv2
from imutils.video import FPS

cam = cv2.VideoCapture(0)  # 打开摄像头
tracker = cv2.TrackerKCF_create()

initBB = None
fps = None
```

```
while True:
    ret, frame = cam.read()  # 获取摄像头数据
    grey = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # 色彩变换
    blur = cv2.blur(grey, (5, 5))  # 过滤噪声
    circles = cv2.HoughCircles(blur,  # 识别圆形
                               method=cv2.HOUGH_GRADIENT, dp=1, minDist=200,
                               param1=100, param2=33, minRadius=30, maxRadius=175)
    # if circles is not None:  # 识别到圆形
    #     for i in circles[0, :]:  # 画出识别的结果
    #         cv2.circle(frame, (i[0], i[1]), int(i[2]), (0, 255, 0), 2)
    #         cv2.circle(frame, (i[0], i[1]), 2, (0, 0, 255), 3)
    if initBB is not None:
        # grab the new bounding box coordinates of the object
        (success, box) = tracker.update(frame)
        # check to see if the tracking was a success
        if success:
            (x, y, w, h) = [int(v) for v in box]
            # if circles is not None:  # 识别到圆形
            #     target = 0; mindiff = 2147483647
            #     for i in circles[0, :]:  # 画出识别的结果
            #         if abs(i[0] - (x+w//2)) + abs(i[1] - (y+h//2)) < mindiff:
            #             target = i
            #             mindiff = abs(i[0] - (x+w//2)) + abs(i[1] - (y+h//2))
            #     cv2.circle(frame, (target[0], target[1]), int(target[2]), (0, 255,
            #     cv2.circle(frame, (target[0], target[1]), 2, (0, 0, 255), 3)
            cv2.circle(frame, (x + w//2, y + h//2), w//2,(0, 255, 0), 2)
        # update the FPS counter
        fps.update()
        fps.stop()

    key = cv2.waitKey(1) & 0xFF
    if key == ord("s"):
        initBB = cv2.selectROI("Detected", frame, fromCenter=False,
                               showCrosshair=True)
        tracker.init(frame, initBB)
        fps = FPS().start()
    cv2.imshow('Detected', frame)  # 显示识别图像
    if key == ord("q"):  # 等待按键
        break
cv2.destroyAllWindows()  # 关闭窗口
cam.release()  # 释放摄像头
```