计算机视觉 (实验一)
利用基元检测方法实现机械臂的运动检测

智科三班　严中圣　222020335220177

2022 年 11 月 14 日

1　实验目的

图像处理 (image processing)，用计算机对图像进行分析，以达到所需结果的技术。图像处理将会是物联网产业发展的重要支柱之一。本实验目的在于加深本课程中图像预处理、基元检测、后处理等重要的知识点的理解和实践，并实现针对视频的机械臂运动情况检测算法，最终生成标注后的视频/流媒体。

2　实验环境

- Visual Studio Code 1.73.1
- OS: Windows 11 22H2
- CPU: 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz
- Packages: python 3.9.12, opencv 4.6.0.66, numpy 1.21.5

3　实验内容

(1) 对视频（固定拍摄位置）进行逐帧检测，检测其中的机械臂边缘；

(2) 根据运动时间阈值，判断机械臂的运动情况；

(3) 对检测结果进行后处理，将结果在图像中进行标注，并实时播放或者保存视频；

(4) 效率思考，若逐帧处理效率较低，考虑提高优化方法。

4　实验步骤

(1) 获取视频并进行逐帧操作

第一步我们可利用 cv.VideoCapture() 从摄像头或视频传入数据，并逐帧进行读取。同时利用 cv.VideoWriter() 建立保存视频对象。

(2) 利用 ROI 进行机械臂边缘预标定

在图像处理领域，感兴趣区域 (ROI) 是从图像中选择的一个图像区域，这个区域是你的图像分析所关注的重点。由于拍摄中噪声和其他影响因素较多，我们首先利用 ROI 截取机械臂附近区域以便于后续特征提取。具体使用函数为 cv.selectROI()，选取 ROI 区域可通过手动或提前传入参数的方式进行。

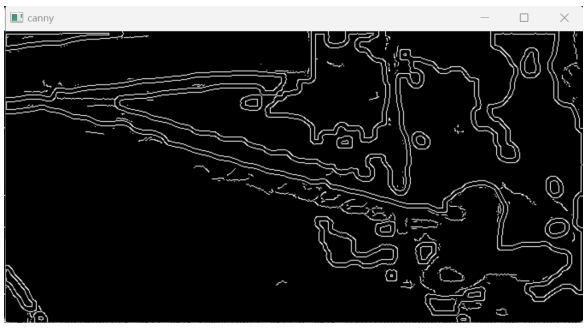
(a) ROI 选取操作

(b) ROI 选取结果

(3) 图像轮廓与边缘提取

在截取 ROI 区域后，我们对该区域首先进行灰度处理，再利用 cv2.blur() 进行均值滤波，对噪声进行抑制；其次利用 cv.morphologyEx() 先进行开运算去除背景噪声，再继续闭运算填充目标内的孔洞，如此操作使得下面 cv.threshold(closed,128,255,cv.THRESH_BINARY) 提取到的轮廓更加连续与饱和，避免了图形块间断带来的影响。在这之后我们利用 Canny 算子进行边缘提取，即可得到机械臂的边缘特征，且很大程度上的过滤掉了场景噪声。

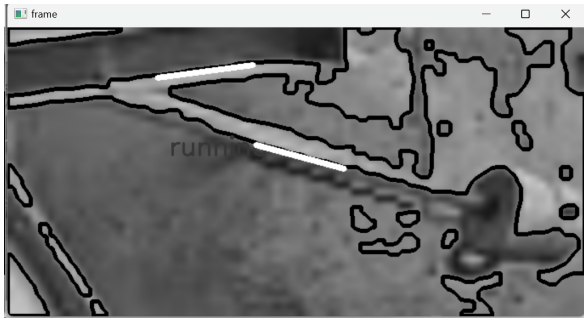
(c) 图像轮廓提取结果



(d) Canny 边缘提取结果

(4) 机械臂的运动检测

对于机械臂的运动检测，我们首先在第三步边缘图的基础上利用概率霍夫变换进行了直线检测，提取图像中出现的直线，并根据斜率将异常值滤去，将机械臂所对应的直线斜率与初始状态斜率进行比较，同时为了避免由于惯性造成的扰动影响结果的判断，我们设定了一个阈值以判断机械臂的运动状态。最后将结果在原始视频中进行标注，并输出最终的结果保存。



(e) 直线检测与运动检测

(f) 结果标注

A 附录

机械臂运动检测 (python)

```python
import numpy as np
import cv2 as cv
cap = cv.VideoCapture("flapping_demo_video.avi")
fourcc = cv.VideoWriter_fourcc(*'DIVX')
out = cv.VideoWriter('output.avi', fourcc, 20.0, (640, 368))
# fourcc2 = cv.VideoWriter_fourcc(*'DIVX')
out2 = cv.VideoWriter('output_crop.avi', fourcc, 20.0, (640, 368), 0)
ret, frame = cap.read()
gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
roi = cv.selectROI(windowName="roi", img=gray,
                   showCrosshair=True, fromCenter=False)
x, y, w, h = roi
cap = cv.VideoCapture("flapping_demo_video.avi")
last_k = 0
k_flag = False
frame_count = 0
state = 0
word_x = 190
word_y = 150
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret == True:
        # process frame by frame
        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        cropped = gray[y:y+h, x:x+w]  # transform to gray image
        cropped = cv.resize(cropped, (640, 368))  # crop the roi region

        # pre-processing
        cropped = cv.blur(cropped, (5, 5))
        kernel = cv.getStructuringElement(cv.MORPH_RECT, (9, 9))
        opened = cv.morphologyEx(cropped, cv.MORPH_OPEN, kernel)
        closed = cv.morphologyEx(opened, cv.MORPH_CLOSE, kernel)
        ret, binary = cv.threshold(closed, 128, 255, cv.THRESH_BINARY)
```

```
        contours, hierarchy = cv.findContours(
            binary, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)  # find the contours
        cv.drawContours(cropped, contours, -1, (0, 0, 255), 3)  # draw the
            contours
        if (frame_count % 3 == 0):
            edges = cv.Canny(cropped, 30, 50, (3, 3))  # extract the edges
            # motion detection
            lines = cv.HoughLinesP(
                edges, 1, np.pi/180, 100, minLineLength=100, maxLineGap=10)  #
                    detect the lines
            if type(lines) == type(None):
                continue
            if (len(lines) != 0):
                for line in lines:
                    x1, y1, x2, y2 = line[0]
                    if ((x1 == x2) or (y1 == y2) or ((x2-x1)/(y2-y1) > 5)):
                        continue
                    k = (x2-x1)/(y2-y1)
                    if (k < 0):
                        continue
                    if (k_flag == True):
                        if (abs(k-last_k) < 0.1):
                            state = 0
                        else:
                            state = 1
                    cv.line(cropped, (x1, y1), (x2, y2), (255, 0, 0), 5)
                    if (k_flag == False):
                        last_k = (x2-x1)/(y2-y1)
                        k_flag = True
        if (state):
            cv.putText(cropped, 'running', (word_x, word_y),
                                cv.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)
            cv.putText(frame, 'running', (word_x, word_y),
                        cv.FONT_HERSHEY_SIMPLEX, 1, (55, 255, 155), 2)
        else:
            cv.putText(cropped, 'stopped', (word_x, word_y),
                                cv.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)
            cv.putText(frame, 'stopped', (word_x, word_y),
                        cv.FONT_HERSHEY_SIMPLEX, 1, (55, 255, 155), 2)
        cv.imshow("frame", cropped)
        out.write(frame)
        out2.write(cropped)
        # cv.imshow("frame", cropped)
        frame_count += 1
        if cv.waitKey(1) & 0xFF == ord('q'):
            break
    else:
```

```
79          break
80  cap.release()
81  cv.destroyAllWindows()
```