

厚标题：新安江水库史上首次正式9孔全开泄洪

新华社杭州7月8日电（记者黄筱、许舜达）7月8日9时起，位于浙江建德的新安江水库开启9孔泄洪闸，据水库相关负责人介绍，此次泄洪是

新安江水库建成61年来首次正式9孔全开泄洪，水库的水位也达到历史最高值108.45米，而此前水库最高水位为1999年6月的108.37米。

智能系统与控制

树莓派：文件扫描 (内容识别, Tesseract)

于泓

鲁东大学

信息与电气工程学院

2022.1.23

任务：

1、 找到图像内的文件的位置

边缘检测，轮廓检测

2、 对文件进行校正

定位轮廓的四个顶点
利用仿射变换，实现图像矫正

3、 识别文件内容（文字）

利用Tesseract 实现

Tesseract的安装







(1) 软件安装

```
sudo apt install tesseract-ocr
```

```
sudo apt install libtesseract-dev
```

(2) 下载中文包 https://github.com/tesseract-ocr/tessdata_best

Tesseract 是目前最好的用于机器打印字符识别的开源 OCR 工具。20 世纪 80 年代由 Hewlett Packard 开发，2005 年开源，自 2006 年起由谷歌赞助开发。Tesseract 支持 Unicode (UTF-8) 字符集，可以识别超过包括简体中文在内的 100 种语言。

 ceb.traineddata	Initial import (on behalf of Ray)	3 years ago
 ces.traineddata	Initial import (on behalf of Ray)	3 years ago
 chi_sim.traineddata	Initial import (on behalf of Ray)	3 years ago
 chi_sim_vert.traineddata	Fix extra intra-word spacing for Chinese (GitHub issue #991)	14 months ago
 chi_tra.traineddata	Initial import (on behalf of Ray)	3 years ago
 chi_tra_vert.traineddata	Fix extra intra-word spacing for Chinese (GitHub issue #991)	14 months ago

(3) 将下载好的文件chi_sim.traineddata

保存到: /usr/share/tesseract-ocr/4.00/tessdata

注意: 保存之前

sudo chmod 777 -R /usr/share/tesseract-ocr/4.00/tessdata

改变文件夹的属性

运行 `tesseract --list-langs`

```
pi@raspberrypi:~ $ cd /usr/share/tesseract-ocr/4.00/tessdata
pi@raspberrypi:/usr/share/tesseract-ocr/4.00/tessdata $ ls
chi_sim.traineddata  configs  eng.traineddata  osd.traineddata  pdf.ttf  tessconfigs
pi@raspberrypi:/usr/share/tesseract-ocr/4.00/tessdata $ tesseract --list-langs
List of available languages (3):
chi_sim
eng
osd
```

(4) 安装Tesseract的Python支持包

`pip3 install pytesseract`

→ 在Python环境下调用
Tesseract 实现文字识别

代码 流程:

- 1 边缘检测
- 2 轮廓检测
- 3 顶点定位
- 4 图像矫正
- 5 文字识别

```
import cv2
import numpy as np
import pytesseract

# 滑块的响应函数
def empty(a):
    pass

# 预处理提取边缘图像，imgCanny边缘提取用到的两个阈值edge_min,edge_max，通过滑块获取
def preProcessing(img,edge_min,edge_max):
    imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray,(5,5),1)
    imgCanny = cv2.Canny(imgBlur,edge_min,edge_max)
    kernel = np.ones((5,5))
    imgDial = cv2.dilate(imgCanny,kernel,iterations=2)
    imgEdge = cv2.erode(imgDial,kernel,iterations=1)
    return imgEdge
```

```
# 轮廓提取
def getContours(img):

    # 需要获取的文件的四个顶点
    biggest = np.array([])
    maxArea = 0

    # 进行图像中轮廓的提取
    contours, hierarchy = cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

    # 对提取的轮廓进行分析
    for cnt in contours:
        # 计算每个轮廓包围的面积
        area = cv2.contourArea(cnt)

        # 对包围面积较大的轮廓进行处理
        if area > 5000:
            # 对轮廓进行折线近似
            peri = cv2.arcLength(cnt, True)
            approx = cv2.approxPolyDP(cnt, 0.02 * peri, True)

            # 找到面积最大且折线数目是四条的区域
            if area > maxArea and len(approx) == 4:
                biggest = approx
                maxArea = area

    # 画四个顶点
    cv2.drawContours(imgContour, biggest, -1, (0, 0, 255), 2)

    # 画四条边界线
    cv2.polylines(imgContour, [biggest], True, (0, 255, 0), 2)

    return biggest
```

2024/3/20

顶点定位

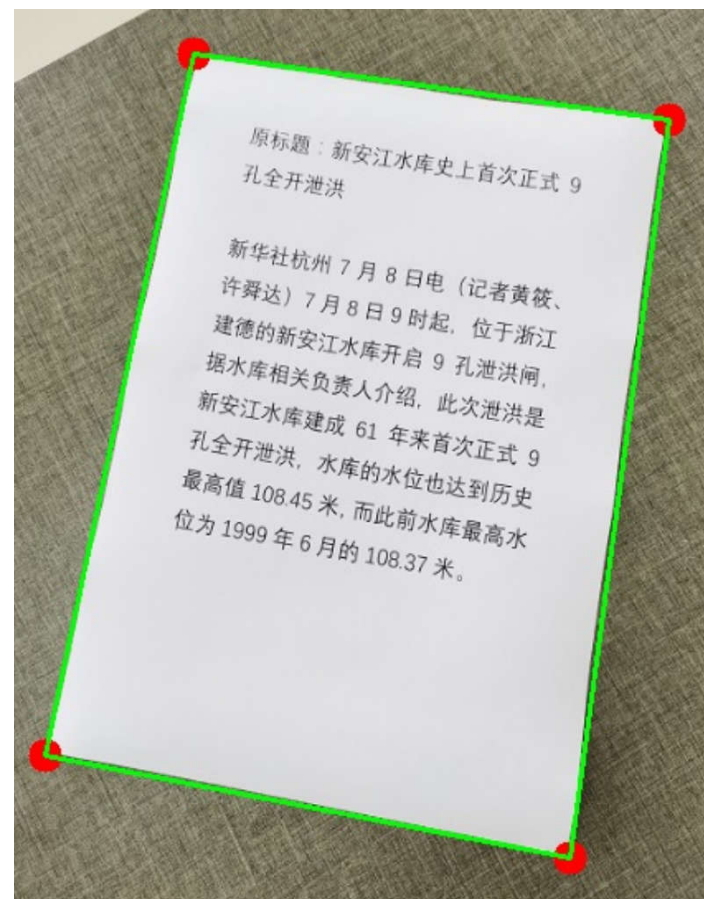
```
# 对获取的文件的四个顶点坐标进行分析定位
def reorder (myPoints):

    myPoints = myPoints.reshape((4,2))
    myPointsNew = np.zeros((4,1,2),np.int32)

    # 将横纵两个坐标相加，最小的就是左上角，最大就是右下角
    add = myPoints.sum(1)
    myPointsNew[0] = myPoints[np.argmin(add)]
    myPointsNew[3] = myPoints[np.argmax(add)]

    # 将纵横 两个坐标相减，最小为左上角，最大为右下角
    diff = np.diff(myPoints,axis=1)
    myPointsNew[1]= myPoints[np.argmin(diff)]
    myPointsNew[2] = myPoints[np.argmax(diff)]

    return myPointsNew
```




```
# 进行文件的剪裁
def getWarp(img_input, biggest, size_out):

    # 获取四个顶点, 对顶点坐标进行定位
    biggest = reorder(biggest)

    # 进行图像的剪裁
    pts1 = np.float32(biggest)
    pts2 = np.float32([[0, 0], [size_out[0], 0], [0, size_out[1]], [size_out[0], size_out[1]]])
    matrix = cv2.getPerspectiveTransform(pts1, pts2)
    imgOutput = cv2.warpPerspective(img, matrix, (size_out[0], size_out[1]))

    # 对边界进行裁剪
    imgCropped = imgOutput[20:imgOutput.shape[0]-20, 20:imgOutput.shape[1]-20]
    imgCropped = cv2.resize(imgCropped, (size_out[0], size_out[1]))

    return imgCropped

if __name__ == "__main__":

    # 创建参数调整滑块
    cv2.namedWindow("TrackBars")
    cv2.resizeWindow("TrackBars", 640, 60)
    cv2.createTrackbar("Edge Min", "TrackBars", 50, 255, empty)
    cv2.createTrackbar("Edge Max", "TrackBars", 50, 255, empty)

    # 剪切后图像的大小 (width, height)
    size_out = [460, 640]

    # 读取图像
    img = cv2.imread('paper.jpg')
```

变换前的坐标

变换后的坐标

```
while True:
    # 获取进行剪裁区域标记的图像
    imgContour = img.copy()
    # 获取边缘检测参数
    edge_min = cv2.getTrackbarPos("Edge Min", "TrackBars")
    edge_max = cv2.getTrackbarPos("Edge Max", "TrackBars")

    # 进行边缘检测
    imgEdge = preProcessing(img, edge_min, edge_max)

    # 剪裁部分定位
    biggest = getContours(imgEdge)

    # 图像显示
    cv2.imshow('Input', img)
    cv2.imshow('Edge', imgEdge)
    cv2.imshow('Contours', imgContour)

    # 获取按键信息, 按"w" 剪裁并识别文字, 按"q"退出
    key=cv2.waitKey(10) & 0xFF

    if key == ord('w'):
        if biggest.size !=0:

            # 图像剪裁
            imgWarped=getWarp(img, biggest, size_out)
            cv2.imshow("ImageWarped", imgWarped)
            key =0;

            # 识别文字
            pytesseract.pytesseract.tesseract_cmd = '/usr/bin/tesseract'
            imgWarped Gray = cv2.cvtColor(imgWarped, cv2.COLOR_BGR2RGB)
            # 识别结果打印
            print(pytesseract.image_to_string(imgWarped_Gray, lang='chi_sim'))
    elif key == ord('q'):
        break
```

定位文件位置