目录

身份证识别

0. 项目简介

a 提取一张正常拍摄的身份证图片的信息。
b 支持识别各种正常角度的身份证图片。

效果预览

图 1 效果

下面是简单的步骤分析

1. 图像校正

图像校正的意思是将原图片里旋转的或者歪扭的身份证变换成水平方正的样子。主要用到 SIFT 特征点匹配来匹配图中身份证的位置，并调用透视变换来把身份证摆正。摆正后的图片保存为 standard.jpg

图 2 进行特征点匹配的模板

```
def siftImageAlignment(img1,img2):
    _,kp1,des1 = sift_kp(img1)
    _,kp2,des2 = sift_kp(img2)
```

```
    goodMatch = get_good_match(des1,des2)
    if len(goodMatch) > 4:
        ptsA= np.float32([kp1[m.queryIdx].pt for m in goodMatch]).reshape(-1, 1, 2)
        ptsB = np.float32([kp2[m.trainIdx].pt for m in goodMatch]).reshape(-1, 1, 2)
        ransacReprojThreshold = 4
        H, status =cv2.findHomography(ptsA,ptsB,cv2.RANSAC,ransacReprojThreshold);


        imgOut = cv2.warpPerspective(img2, H, (img1.shape[1],img1.shape[0]),flags=cv2.INTER_LIN
EAR + cv2.WARP_INVERSE_MAP)
        return imgOut,H,status
```

code1 特征点匹配和透视变换的核心代码

图 3 校正效果，左边是原图，右边是校正后的图片 standard.jpg

2. 文字区域截取

得到了校正的图片之后,我们要把文字区域截取出来。

第一步，我根据身份证的文字信息分布特征制作了一张图片(mask.jpg)。可以清晰看见，下左图的空白处正是对应了是身份证的关键信息处。这里我故意将空白区域设计得大一点，下面会讲解原因。

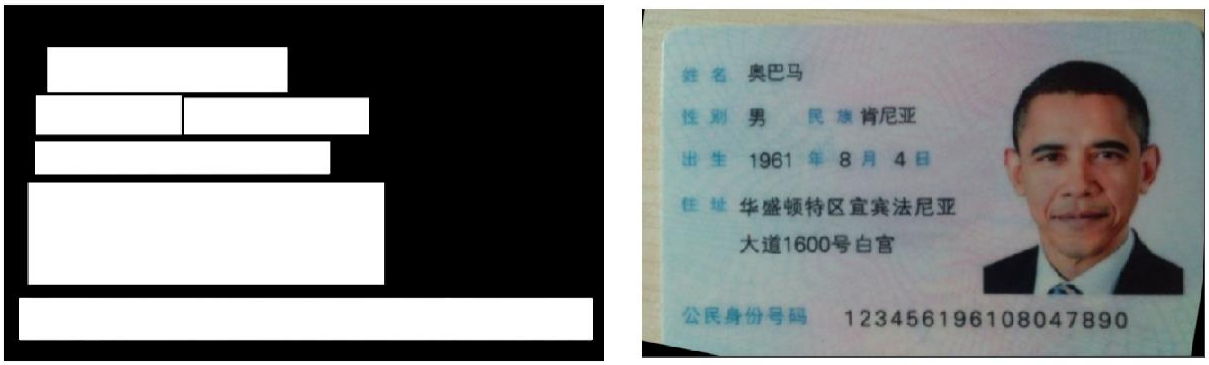

图 4 特殊图片 mask.jpg

第二步, 在程序中读入 mask.jpg 并转换成和 standard.jpg 一样的尺寸。

第三步，查找转换后的 mask.jpg 中的轮廓，并把这些轮廓的坐标点保存下来。因为 mask.jpg 中的白色区域都是矩形的，而且界限清晰，所以每一次都能找到所有矩形轮廓。又因为 mask.jpg 的尺寸已经和 standard.jpg 一致，所以接下来只要用这一步得到的坐标点就能在 standard.jpg 中裁剪出关键信

息

第四步,利用第三步得到的坐标点裁剪出区域文字图片，并保持下来。

```
def get_boxs(img):
    backup = img
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # 二值化图片
    ret, th = cv2.threshold(img, 100, 255, cv2.THRESH_BINARY)
    kernel = np.ones((10, 20), np.uint8)
    # 开运算
    closing = cv2.morphologyEx(th, cv2.MORPH_OPEN, kernel)
    # 腐蚀
    kernel = np.ones((5, 10), np.uint8)
    dilation = cv2.erode(closing, kernel, iterations=1)
    #  查找和筛选文字区域
    region = []
    #  查找轮廓
    img2, contours, hierarchy = cv2.findContours(dilation, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIM
PLE)
    # 利用以上函数可以得到多个轮廓区域，存在一个列表中。
    #  筛选那些面积小的
    for i in range(len(contours)):
        # 遍历所有轮廓
        # cnt 是一个点集
        cnt = contours[i]
        # 计算该轮廓的面积
        area = cv2.contourArea(cnt)
        # 面积小的都筛选掉、这个 300 可以按照效果自行设置
        if (area < 300):
            continue
        # 找到最小的矩形，该矩形可能有方向
        rect = cv2.minAreaRect(cnt)
        # box 是四个点的坐标
        box = cv2.boxPoints(rect)
        box = np.int0(box)
        # 计算高和宽
        height = abs(box[0][1] - box[2][1])
        width = abs(box[0][0] - box[2][0])
        # 筛选那些太细的矩形，留下扁的
        if (height > width):
            continue
        region.append(box)
    return region
```

code2 获取 mask.jpg 的轮廓的代码

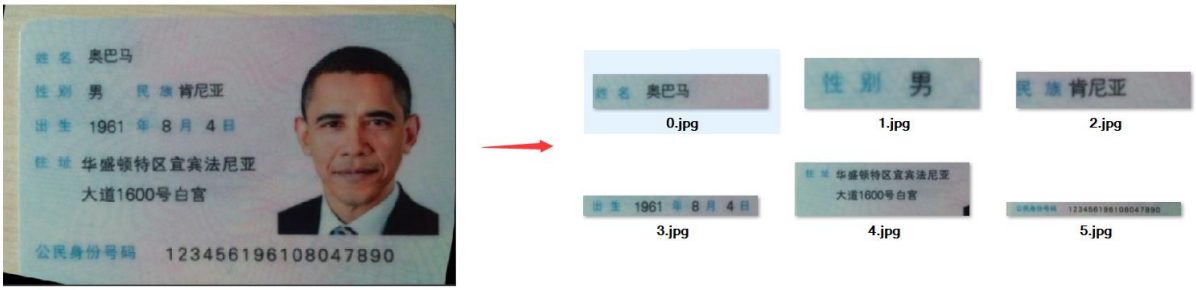

图 5 截取文字区域图片效果图

3. 调用百度 OCR API 识别图片

这步比较简单，只需要在注册一个百度开发者账号，新建一个文字识别的应用，然后看说明文档，学习怎么调用 ocr 接口就行。最后只要遍历读取识别上面保存的文字区域图片并调用 api 即可识别出文字。这个 api 返回的是包含很多信息的字典，所以后续要自己进一步加工信息。

```
{'words_result': [{'words': '住址华盛顿特区宜宾法尼亚'}, {'words': '大道 1600 号白宫'}], 'log_id':
 1412799783726533144, 'words_result_num': 2}
```

Code3 文字识别 api 返回的数据结构示例

```
""" 你的 APPID AK SK """
APP_ID = '24511458'
API_KEY = 'NrVbW2kGmNQsI47004QHyUwF'
SECRET_KEY = 'k3iGLx1twoDjRRPteAn6UnXqxATztQ6B'
client = AipOcr(APP_ID, API_KEY, SECRET_KEY)

""" 读取图片 """
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()
#返回值是字典
def get_pic_info(filename):
    image = get_file_content(filename)
    """ 如果有可选参数 """
    options = {}
    options["language_type"] = "CHN_ENG"
    options["detect_direction"] = "false"
    options["detect_language"] = "false"
    options["probability"] = "false"
```

```
    """ 带参数调用通用文字识别，图片参数为本地图片 """
    return client.basicAccurate(image, options)
```

Code4　调用百度 api 的核心代码

4. 附录(源码)

项目目录结构

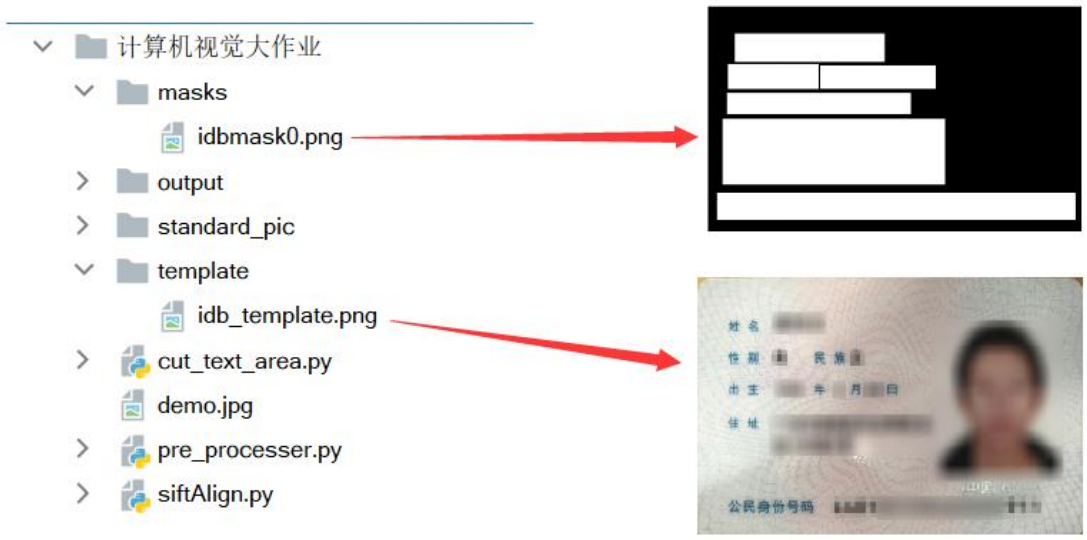

图 6 项目目录结构

siftAlign.py

```
#sifiAlign.py
import cv2
import numpy as np
import os

def sift_kp(image):
    gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    sift = cv2.xfeatures2d_SIFT.create()
    kp,des = sift.detectAndCompute(image,None)
    kp_image = cv2.drawKeypoints(gray_image,kp,None)
    return kp_image,kp,des

def surf_kp(image):
```

```
    '''SIFT(surf)特征点检测(速度比 sift 快)'''
    height, width = image.shape[:2]
    size = (int(width * 0.2), int(height * 0.2))
    shrink = cv2.resize(image, size, interpolation=cv2.INTER_AREA)
    gray_image = cv2.cvtColor(shrink,cv2.COLOR_BGR2GRAY)
    surf = cv2.xfeatures2d_SURF.create()
    kp, des = surf.detectAndCompute(gray_image, None)
    return kp,des

def get_good_match(des1,des2):
    bf = cv2.BFMatcher()
    matches = bf.knnMatch(des1, des2, k=2)
    good = []
    for m, n in matches:
        if m.distance < 0.75 * n.distance:
            good.append(m)
    return good

def siftImageAlignment(img1,img2):
   _,kp1,des1 = sift_kp(img1)
   _,kp2,des2 = sift_kp(img2)

   goodMatch = get_good_match(des1,des2)
   if len(goodMatch) > 4:
       ptsA= np.float32([kp1[m.queryIdx].pt for m in goodMatch]).reshape(-1, 1, 2)
       ptsB = np.float32([kp2[m.trainIdx].pt for m in goodMatch]).reshape(-1, 1, 2)
       ransacReprojThreshold = 4
       H, status =cv2.findHomography(ptsA,ptsB,cv2.RANSAC,ransacReprojThreshold);
       imgOut = cv2.warpPerspective(img2, H, (img1.shape[1],img1.shape[0]),flags=cv2.INTER_LIN
EAR + cv2.WARP_INVERSE_MAP)
       return imgOut,H,status

def get_sandard_card(img):
    """
    :param img: 传入待摆正的图片
    :param ctype: 传入一个整数,0 代表身份证背面，1 代表身份证正面 2 代表社保卡背面 3 代表社保卡反面
    :return: 返回一个图片
    """
    img1 = cv2.imread(r'./template/idb_template.png')

    result, h, status = siftImageAlignment(img1,img)
    result, h, status = siftImageAlignment(img1,result)
    #保存中间结果
    cv2.imwrite('./standard_pic/'+'standard.jpg',result)
    return result
```

cut_text_area.py

```
import sys
import numpy as np
import cv2

def cut_text(img,pts):
    pts = pts.tolist()
    minx = 99999
    miny = 99999
    maxx = -11
    maxy = -11
    for pt in pts:
        maxy = max(maxy,pt[1])
        maxx = max(maxx,pt[0])
        minx = min(minx,pt[0])
        miny = min(miny,pt[1])
    return img[miny:maxy,minx:maxx]

def get_boxs(img):
    backup = img
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # 二值化图片
    ret, th = cv2.threshold(img, 100, 255, cv2.THRESH_BINARY)
    kernel = np.ones((10, 20), np.uint8)
    # 开运算
    closing = cv2.morphologyEx(th, cv2.MORPH_OPEN, kernel)
    # 腐蚀
    kernel = np.ones((5, 10), np.uint8)
    dilation = cv2.erode(closing, kernel, iterations=1)
    #  查找和筛选文字区域
    region = []
    #  查找轮廓
    img2, contours, hierarchy = cv2.findContours(dilation, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIM
PLE)
    # 利用以上函数可以得到多个轮廓区域，存在一个列表中。
    #  筛选那些面积小的
    for i in range(len(contours)):
        # 遍历所有轮廓
        # cnt 是一个点集
        cnt = contours[i]
        # 计算该轮廓的面积
```

```
        area = cv2.contourArea(cnt)
        # 面积小的都筛选掉、这个 300 可以按照效果自行设置
        if (area < 300):
            continue
        # 找到最小的矩形，该矩形可能有方向
        rect = cv2.minAreaRect(cnt)
        # box 是四个点的坐标
        box = cv2.boxPoints(rect)
        box = np.int0(box)
        # 计算高和宽
        height = abs(box[0][1] - box[2][1])
        width = abs(box[0][0] - box[2][0])
        # 筛选那些太细的矩形，留下扁的
        if (height > width):
            continue
        region.append(box)
    return region

def getvalue(pts):
    pts = pts.tolist()
    minx = 99999
    miny = 99999
    maxx = -11
    maxy = -11
    for pt in pts:
        maxy = max(maxy, pt[1])
        maxx = max(maxx, pt[0])
        minx = min(minx, pt[0])
        miny = min(miny, pt[1])
    return miny

def get_text_area(img):
    mask = cv2.imread('./masks/idbmask0.png')
    mask = cv2.resize(mask, (img.shape[1], img.shape[0]), interpolation=cv2.INTER_CUBIC) #大小
    boxs = get_boxs(mask)
    names = ['name','sexual','nation','date','addr','id']
    #按坐标排序
    boxs= sorted(boxs, key=getvalue)
    idx = 0;
    img_dict = {}
    for pt in boxs:
        # 保存到本地，方便查看中间结果
        img_dict[names[idx]] = cut_text(img, pt)
        idx += 1
    return img_dict
```

pre_processer.py

```
import cut_text_area as Atc
import siftAlign as Ats
import cv2
from aip import AipOcr
import time

""" 你的 APPID AK SK """
APP_ID = '24511458'
API_KEY = 'NrVbW2kGmNQsI47004QHyUwF'
SECRET_KEY = 'k3iGLx1twoDjRRPteAn6UnXqxATztQ6B'
client = AipOcr(APP_ID, API_KEY, SECRET_KEY)
""" 读取图片 """
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

def get_pic_info(filename):
    image = get_file_content(filename)
    """ 如果有可选参数 """
    options = {}
    options["language_type"] = "CHN_ENG"
    options["detect_direction"] = "false"
    options["detect_language"] = "false"
    options["probability"] = "false"
    """ 带参数调用通用文字识别，图片参数为本地图片 """
    return client.basicAccurate(image, options)

def formated(res):
    if '姓名' in res:
        res = '姓名: ' + res[2:]
    elif '性别' in res:
        res = '性别: ' + res[2:]
    elif '民族' in res:
        res = '民族: ' + res[2:]
    elif '出生' in res:
        res = '出生: ' + res[2:]
    elif '住址' in res :
        res = '住址: ' + res[2:]
    elif '公民身份号码' in res:
        res = '公民身份号码: ' + res[6:]
    return  res
```

```
# 按排列顺序
idb_info_list = ['name', 'sexual', 'nation', 'date', 'addr', 'id']

def pre_processer(path):
    # 0. 摆正身份证背面
    img = cv2.imread(path)
    # img0 是摆正好的图片
    standimg =  Ats.get_sandard_card(img)
    #获取文字区域
    res_dict = Atc.get_text_area(standimg)
    tmplist = idb_info_list
    # 按顺序保存
    for i in range(len(tmplist)):
        name = tmplist[i]
        img = res_dict[name]
        cv2.imwrite('output'+'\\'+str(i)+'.jpg',img)

if __name__ == '__main__':
    pre_processer('demo.jpg') #d 测试图片
    for i in range(6):
        info = get_pic_info('./output/'+str(i)+'.jpg')
        time.sleep(0.5) #识别太快 百度的 api 会报错
        infolist= info["words_result"]
        res = ''
        for item in infolist:
            res += item['words']
        print(formated(res))
```