**Python大作业说明文档MosaicPuzzle.py**

蒙太奇照片，或称相片马赛克、蒙太奇拼贴，是一种影像处理的艺术技巧，利用这个方式做出来的图片，近看时是由许多张小照片合在一起的，但远看时，每张照片透过光影和色彩的微调，组成了一张大图的基本像素，就叫做相片马赛克技巧。最先是由一个美国大学生发明的，但当时限于计算机性能，无法大量应用。目前这个技巧几乎全部是用电脑软件做成的，第一次大量出现可能是在电影海报《楚门的世界》当中。（来源于维基百科[相片马赛克](https://zh.wikipedia.org/wiki/%E7%9B%B8%E7%89%87%E9%A6%AC%E8%B3%BD%E5%85%8B)）



1. **基本原理**

蒙太奇照片的实现原理，提供一个目标图像，把大图分割为若干小图，在素材库中寻找与小图最为相似的图片，寻找图的时候首先考虑的是颜色的相似，另外还需要考虑形状的相似。因此要实现这一效果，需要大量的图片。

1. **数据准备**

数据选择使用的是ImageNet数据集中的验证集，五万张图片。

1. **图片预处理**

由于每张图片大小不一致，拼图的时候不好处理，首先使用OpenCV resize每一张图片至100\*100，imagePath代表图片路径，savePath表示resize后图片存储的路径：

**#图片准备阶段：数据集图片归一化**

**def resizeImage(imagePath,savePath):**

**files=os.listdir(imagePath)**

**#遍历所有图片文件，归一化**

**for file in files:**

**imgPath=imagePath+"//"+file**

**img=cv2.imread(imgPath)**

**img=cv2.resize(img, (100, 100))**

**cv2.imwrite(savePath + "\\" + file, img)**

**cv2.waitKey()**

**return True**

1. **建立图片索引**

建立索引就是保存每张图片出现次数最多的像素值，然后保存到文件中。格式为 文件名称：B,G,R。建立好索引文件之后，就可以知道每个图片和他们最常见的颜色。在使用这些图片的时候，就可以读取整个索引文件，然后定位到图片文件本身。如果没有索引文件，那么生成一次马赛克拼图就要求遍历一次所有图片的最多像素，太耗时间。imagePath表示图片路径,txtPath：索引文件保存路径。

**#图片准备阶段：数据集建立索引，保存每张图片出现次数最多的像素值然后保存到文件中**

**def createIndex(imagePath,txtPath):**

**files=os.listdir(imagePath)**

**txtFile=open(txtPath,'w')**

**for file in files:**

**li=[]**

**txt=''**

**imgPath=imagePath+"//"+file**

**img=cv2.imread(imgPath)**

**shape=np.shape(img)**

**height=shape[0]**

**width=shape[1]**

**for i in range (height):**

**for j in range(width):**

**b = img[i, j, 0]**

**g = img[i, j, 1]**

**r = img[i, j, 2]**

**li.append((b, g, r))**

**#计算图片中像素值出现最多的颜色**

**most = collections.Counter(li).most\_common(1)**

**txt+=file**

**txt+=":"**

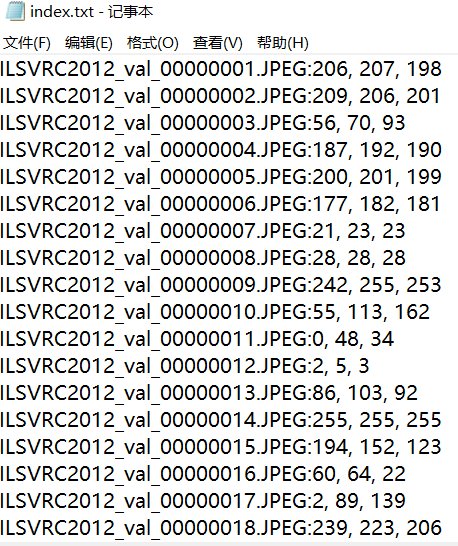
**txt+= str(most[0][0]).replace("(","").replace(")","")**

**txt+="\n"**

**txtFile.write(txt)**

**txtFile.close()**

最后生成的索引文件如图：



1. **生成图片**

假设目标图片是n\*m的，那么新建一个n\*100，m\*100的图片（这里是100的整数倍是因为刚才把图片集中的每个图片都更改为100\*100大小的了）。这是一个很大的图片。对于原图片的每个像素，都遍历索引文件，找到和BGR像素值的欧式距离不大于一个阈值的索引文件中的值，然后根据找到的BGR从索引文件中定位到具体的图片，把图片放到大图上的具体位置。

**#开始制作马赛克 读取索引文件**

**def readIndex(txtPath):**

**file=open(txtPath,"r")**

**dic=[]**

**#按行搜索**

**for line in file.readlines():**

**#按 : 分割字符 文件名+B,G,R**

**temp=line.split(":")**

**imgName=temp[0]**

**bgr=temp[1].split(",")**

**#转为0~255**

**b=int(bgr[0])**

**g=int(bgr[1])**

**r=int(bgr[2])**

**dic.append((imgName,(b,g,r)))**

**return dic**

**if \_\_name\_\_ == '\_\_main\_\_':**

**imagePath="D://pythonData//GrandsCausses2.jpg"**

**txtPath="D://BaiduNetdiskDownload//index.txt"**

**filepath=""**

**saveImagePath="D://BaiduNetdiskDownload//resize"**

**resultPath="D://pythonData//GrandsCaussesResult.jpg"**

**#resizeImage(imagePath,saveImagePath)**

**#createIndex(saveImagePath,txtPath)**

**image=cv2.imread(imagePath)**

**shape=np.shape(image)**

**height = shape[0]**

**width = shape[1]**

**result=np.zeros((100\*height,100\*width,3),dtype=np.uint8)**

**list=readIndex(txtPath)**

**for i in range(height):**

**for j in range(width):**

**b = image[i, j, 0]**

**g = image[i, j, 1]**

**r = image[i, j, 2] # 获取图像当前位置的BGR值**

**for item in list:**

**imgb = item[1][0]**

**imgg = item[1][1]**

**imgr = item[1][2] # 获取索引文件的RGB值**

**distance = (imgb - b) \*\* 2 + (imgg - g) \*\* 2 + (imgr - r) \*\* 2 # 欧式距离**

**if distance < 100:**

**filepath = saveImagePath + "\\" + str(item[0]) # 定位到具体的图片文件**

**break**

**little = cv2.imread(filepath) # 读取整个最相近的图片**

**result[i \* 100:(i + 1) \* 100, j \* 100:(j + 1) \* 100] = little # 把图片放到大图的相应位置**

**cv2.imwrite(resultPath, result) # 输出大图到文件中**

最后效果：

resize 耗时 25分钟

建立索引耗时 10分钟

目标图片大小为400\*300 耗时10分钟

运行速度受目标图片大小和硬件性能影响。

原图：



效果图：



附：数据集下载地址：

链接:https://pan.baidu.com/s/1xsynTjcitN86ep2VWRbSeQ 提取码:c1i0