**Project3-Morphing**

一、运行环境

操作系统：Windows11

Python版本：3.8

第三方库：matplotlib 3.7.1、Pillow 9.4.0、numpy 1.24.2、scipy 1.10.1、imageio 2.26.0

二、过程分析

①选择特征点对并进行三角剖分

使用matplotlib库显示源图像与目标图像，并定义鼠标点击事件以通过鼠标点击获取特征点对，然后调用scipy库的Delaunay函数对源图像进行三角剖分。

②获取平均形状

在每一对特征点之间进行线性插值，以获得源图像与目标图像间的平均形状。

③变形warping

寻找每一个像素点在平均形状中所属三角形面片，并计算其重心坐标。利用重心坐标求出每一个像素点在源图像和目标图像中的对应位置，再通过后向变换将源图像和目标图像都变形到平均形状。

④Cross-Dissolve

对同一平均形状下的源图像和目标图像进行插值，得到最终的图像。

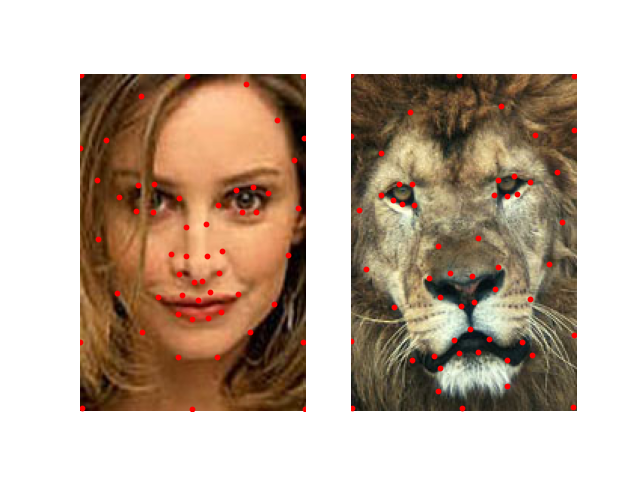
⑤生成GIF文件

使用imageio库将得到的一系列最终图像合成为一个GIF文件。

此外，为了提高计算速度，可以使用multiprocessing库创建多个进程，由各进程并行计算不同参数下的morphing图像。

三、运行结果

①选取特征点：



②Morphing后的图像：

③GIF文件：



四、源代码

|  |
| --- |
| **import** matplotlib.pyplot **as** plt **from** PIL **import** Image **import** numpy **as** np **from** scipy.spatial **import** Delaunay **import** os **import** imageio.v2 **as** imageio **import** multiprocessing   **def** click\_correspondences(im1, im2):  *# 判断两张图片是否为RGB三通道且大小相同* **assert** im1.mode == **'RGB' and** im2.mode == **'RGB'  assert** im1.size == im2.size   *# 创建窗口和两个子图* fig, (ax1, ax2) = plt.subplots(1, 2)  *# 显示图像* ax1.imshow(im1)  ax2.imshow(im2)  *# 设置子图坐标轴不可见* ax1.axis(**'off'**)  ax2.axis(**'off'**)   *# 创建两个列表，用于存放所选特征点坐标* im1\_pts = []  im2\_pts = []   *# 定义函数，获取鼠标点击位置* **def** on\_press(event):  ix, iy = event.xdata, event.ydata  **if** ix **is not None and** iy **is not None**:  **if** event.inaxes == ax1:  im1\_pts.append((int(ix), int(iy)))  ax1.scatter(ix, iy, s=10, color=**'red'**)  **elif** event.inaxes == ax2:  im2\_pts.append((int(ix), int(iy)))  ax2.scatter(ix, iy, s=10, color=**'red'**)  fig.canvas.draw() *# 更新图像   # 绑定鼠标点击事件* cid = fig.canvas.mpl\_connect(**'button\_press\_event'**, on\_press)   **while True**:  *# 按下空格键退出程序* **if** plt.waitforbuttonpress(0):  **if** len(im1\_pts) >= 3 **and** len(im1\_pts) == len(im2\_pts):  **break  else**:  print(**"ERROR: The number of feature points in both images must be the same and greater than 3!"**)   *# 解绑鼠标点击事件* fig.canvas.mpl\_disconnect(cid)  *# 关闭图像* plt.close(fig)  *# 返回选取的特征点* **return** im1\_pts, im2\_pts   **def** morph(im1, im2, im1\_pts, im2\_pts, warp\_frac, dissolve\_frac):  *# 判断两张图片是否为RGB三通道且大小相同* **assert** im1.mode == **'RGB' and** im2.mode == **'RGB'  assert** im1.size == im2.size   *# 获取特征点的x坐标和y坐标* im1\_pts\_x, im1\_pts\_y = np.stack(im1\_pts, axis=1)  im2\_pts\_x, im2\_pts\_y = np.stack(im2\_pts, axis=1)   *# 对两张图片的特性点进行插值* x\_mean = (1.0 - warp\_frac) \* im1\_pts\_x + warp\_frac \* im2\_pts\_x  y\_mean = (1.0 - warp\_frac) \* im1\_pts\_y + warp\_frac \* im2\_pts\_y  pts\_mean = np.column\_stack((x\_mean, y\_mean))   *# 创建正则化网格点坐标* width, height = im1.size  im\_mesh = np.zeros((width \* height, 2))  **for** i **in** range(height):  **for** j **in** range(width):  im\_mesh[i \* width + j, 0] = i  im\_mesh[i \* width + j, 1] = j   *# 三角剖分* tri = Delaunay(im1\_pts)  *# 寻找每个像素点所属三角形索引* t = tri.find\_simplex(im\_mesh)  *# tri\_vertex的每一行表示一个三角形的三个顶点的索引* tri\_vertex = tri.simplices   *# 计算重心坐标* barycentric = np.zeros((width \* height, 3))  **for** i **in** range(width \* height):  **if** t[i] < 0:  barycentric[i, :] = [np.nan, np.nan, np.nan]  **continue** ax = pts\_mean[tri\_vertex[t[i], 0], 0]  bx = pts\_mean[tri\_vertex[t[i], 1], 0]  cx = pts\_mean[tri\_vertex[t[i], 2], 0]  ay = pts\_mean[tri\_vertex[t[i], 0], 1]  by = pts\_mean[tri\_vertex[t[i], 1], 1]  cy = pts\_mean[tri\_vertex[t[i], 2], 1]  A = np.array([[ax, bx, cx], [ay, by, cy], [1, 1, 1]])  barycentric[i, :] = np.transpose(np.linalg.inv(A) @ np.array([[im\_mesh[i, 0]], [im\_mesh[i, 1]], [1]]))   *# 计算每个像素点对应于第一张图片的位置* im1\_crsp = np.zeros((width \* height, 2))  **for** i **in** range(width \* height):  **if** t[i] < 0:  im1\_crsp[i, :] = [np.nan, np.nan]  **continue** ax = im1\_pts\_x[tri\_vertex[t[i], 0]]  bx = im1\_pts\_x[tri\_vertex[t[i], 1]]  cx = im1\_pts\_x[tri\_vertex[t[i], 2]]  ay = im1\_pts\_y[tri\_vertex[t[i], 0]]  by = im1\_pts\_y[tri\_vertex[t[i], 1]]  cy = im1\_pts\_y[tri\_vertex[t[i], 2]]  A = np.array([[ax, bx, cx], [ay, by, cy], [1, 1, 1]])  X = A @ np.transpose(barycentric[i, :])  im1\_crsp[i, :] = [X[0], X[1]]   *# 计算每个像素点对应于第二张图片的位置* im2\_crsp = np.zeros((width \* height, 2))  **for** i **in** range(width \* height):  **if** t[i] < 0:  im2\_crsp[i, :] = [np.nan, np.nan]  **continue** ax = im2\_pts\_x[tri\_vertex[t[i], 0]]  bx = im2\_pts\_x[tri\_vertex[t[i], 1]]  cx = im2\_pts\_x[tri\_vertex[t[i], 2]]  ay = im2\_pts\_y[tri\_vertex[t[i], 0]]  by = im2\_pts\_y[tri\_vertex[t[i], 1]]  cy = im2\_pts\_y[tri\_vertex[t[i], 2]]  A = np.array([[ax, bx, cx], [ay, by, cy], [1, 1, 1]])  X = A @ np.transpose(barycentric[i, :])  im2\_crsp[i, :] = [X[0], X[1]]   *# 计算第一张图片变换后的结果* im1\_morph = []  **for** i **in** range(height):  **for** j **in** range(width):  x = im1\_crsp[i \* width + j, 0]  y = im1\_crsp[i \* width + j, 1]  **if** np.isnan(x) **or** np.isnan(y):  pixel = im1.getpixel((j, i))  **else**:  **try**:  pixel = im1.getpixel((y, x))  **except** IndexError:  pixel = im1.getpixel((j, i))  im1\_morph.append(pixel)   *# 计算第二张图片变换后的结果* im2\_morph = []  **for** i **in** range(height):  **for** j **in** range(width):  x = im2\_crsp[i \* width + j, 0]  y = im2\_crsp[i \* width + j, 1]  **if** np.isnan(x) **or** np.isnan(y):  pixel = im2.getpixel((j, i))  **else**:  **try**:  pixel = im2.getpixel((y, x))  **except** IndexError:  pixel = im2.getpixel((j, i))  im2\_morph.append(pixel)   *# 对变换后的两张图片进行插值* morphed\_im = Image.new(**'RGB'**, (width, height), **'white'**)  **for** i **in** range(height):  **for** j **in** range(width):  r1, g1, b1 = im1\_morph[i \* width + j]  r2, g2, b2 = im2\_morph[i \* width + j]  r = (1.0 - dissolve\_frac) \* r1 + dissolve\_frac \* r2  g = (1.0 - dissolve\_frac) \* g1 + dissolve\_frac \* g2  b = (1.0 - dissolve\_frac) \* b1 + dissolve\_frac \* b2  morphed\_im.putpixel((j, i), (int(r), int(g), int(b)))   **return** morphed\_im   **def** create\_gif(image\_folder):  gif\_name = **'morph.gif'** *# 当GIF文件已存在时将其删除* **if** os.path.exists(gif\_name):  os.remove(gif\_name)  *# 获取图片路径* file\_name = [os.path.join(image\_folder, f) **for** f **in** os.listdir(image\_folder)]  *# 对图片进行排序* images = sorted(file\_name, key=**lambda** x: int(x[len(image\_folder)+1:-4]))  **with** imageio.get\_writer(gif\_name, mode=**'I'**) **as** writer:  **for** image **in** images:  writer.append\_data(imageio.imread(image))  *# 最后一帧重复5次* **for** i **in** range(5):  writer.append\_data(imageio.imread(images[-1]))   *# 子进程函数* **def** function(num, images, im1, im2, im1\_pts, im2\_pts, dir\_name, k):  p = 1.0 / (num \* images)  **for** i **in** range(images):  No = k \* images + i  morphed\_img = morph(im1, im2, im1\_pts, im2\_pts, p \* No, p \* No)  morphed\_img.save(dir\_name + **f'/{**No**}.jpg'**)   **if** \_\_name\_\_ == **'\_\_main\_\_'**:  *# 获取两张图片的特征点* img1 = Image.open(**'img1.jpg'**)  img2 = Image.open(**'img2.jpg'**)  img1\_pts, img2\_pts = click\_correspondences(img1, img2)   *# 新建文件夹以存储morph图片* directory = **'Morphed\_Images'  if** os.path.exists(directory):  files = [os.path.join(directory, f) **for** f **in** os.listdir(directory)]  **for** f **in** files:  os.remove(f)  **else**:  os.mkdir(directory)   *# 创建多进程执行不同参数的morph函数* process\_num = 4  img\_per\_proc = 5  para\_list = [(process\_num, img\_per\_proc, img1, img2, img1\_pts, img2\_pts, directory, k) **for** k **in** range(process\_num)]  pool = multiprocessing.Pool(process\_num)  pool.starmap(function, para\_list)  pool.close()  pool.join()   final\_morphed\_img = morph(img1, img2, img1\_pts, img2\_pts, 1, 1)  final\_morphed\_img.save(directory + **f'/{**process\_num \* img\_per\_proc**}.jpg'**)   *# 将创建的morph图片合成为GIF文件* create\_gif(directory) |