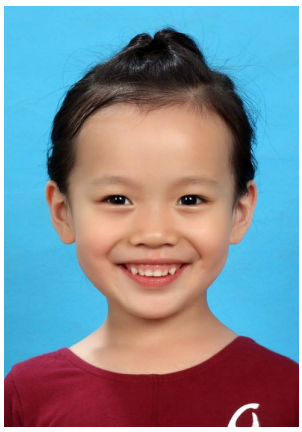
计算机视觉附加题 2 作业报告

陈铭涛

16340024

作业要求

输入图像:

输出:

根据 Image Morphing 的方法完成中间 11 帧的差值，得到一个 Image Morphing 的动画视频。

基本方法

Image Morphing 的主要目标是找到起点与终点图片的以范围为(0, 1) 的 t 为权值的带权均值。然而若只是对两图片的每个像素求带权均值，当两张图片的各构图对象并不对齐时，就可能造成输出的图像出现不同物件的重影。因此若要使输出的图像转换足够顺畅，需要对两张图片的各个特征物件先进行扭转，将图像中标注的各个特征对齐，再求其均值。当输出需要多张中间帧时，使用上面的同样方法求出带权的平均特征位置，再将两张图 warp 到该平均位

置。 这一变换过程主要有两种方法：第一种是使用 Delaunay Triangulation 将输入的所有特征点划分为一个三角形集，其中没有任何一个三角形处于其他任意三角形的外接圆内部。完成划分后对每一个三角形的内部的每一个像素进行变换使得变换后三角形的三个顶点可以对应。

第二种方法是使用 Beier & Neely Algorithm，将输入的图像特征点划分为向量，对于变换后输出图像的每一个像素，对于每一个向量求出其与原图像中对应像素位置的偏差，每个像素与向量的对应位置算法如下：

$$u = \frac{(X-P)\cdot(Q-P)}{||Q-P||^2}$$

$$v = \frac{(X-P)\cdot perp(Q-P)}{||Q-P||}$$

$$X' = P' + u\cdot(Q'-P') + \frac{v\cdot perp(Q'-P')}{||Q'-P'||}$$

其中带'的符号代表原图的数据。求出偏差后带权相加求出其在原图像中对应的像素。权值的算法为$weight = \left(\frac{length^p}{a+dist}\right)^b$，其中 length 为向量长度，dist 为目标像素距离向量的距离。

使用一种方法对图像进行 warping 后，使用 t 对 warping 后的两张图片带权求均值即可得到 t 对应的中间帧输出。

在本次提交的代码中，使用了第二种方法进行实现。

代码简述

提交代码中主要的算法部分在 ImageMorphing 类中，其中包含的类成员与说明如下：

```
ImageMorphing(string s, string d);
```

构造函数，传入原图和目标图的文件路径

```
typedef pair<int, int> point;
```

图像中每一个点的位置视为两个整形的 pair

```
pair<CImg<unsigned char>,CImgList<unsigned char>> morph(const
vector<pair<point, point>>& points, unsigned intermediates);
```

进行图像变换的函数，传入的第一个参数是两图对应的特征点的列表，第二个参数是中间帧的数量，返回的第一个对象是生成的所有帧水平连接在一起的 CImg 对象，第二个返回的对象是所有帧的列表，总帧数为中间帧数+2.

```
static vector<pair<point, point>> readPointsFromFile(string filename);
```

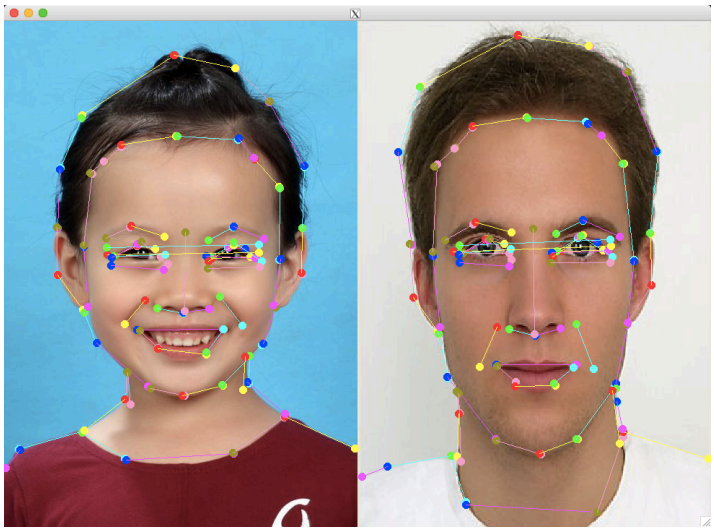
从指定文件中读取所有对应特征点，参数为文件路径。

```
static void savePointToFile(const vector<pair<point, point>>& points,
string filename);
```

将给定的对应特征点列表保存到文件，传入特征点列表和保存的文件路径

```
vector<pair<point, point>> selectPoints(const vector<pair<point, point>>&
existing = vector<pair<point, point>>());
```

打开一个 CImgDisplay 窗口来给用户选择特征点，返回选定的特征点的列表，可以传入一个已存在的特征点列表，在其基础上继续选择特征点。选择过程中每两个特征点会加一条线表示两点被视为一条向量。对应特征点颜色相同。效果如下：



```
static vector<pair<point, point>> makeLines(const vector<point>&
points)
```

从传入的特征点列表中构建一个向量的列表。

```
static CImg<unsigned char> warp(const CImg<unsigned char> & src,
vector<pair<point, point>> oldPt, vector<pair<point, point>> newPt);
```

对传入的 CImg 对象进行 warping，传入参数为要进行操作的对象，原图的向量数据和目标图的向量数据。返回完成 warping 后的 CImg 对象。Warping 中使用的 a, b, p 值分别为 1，1，2。

程序测试运行

使用 main.cpp 内的代码对程序进行测试，bin 文件夹中已包含了 Mac, Linux 以及 Windows 下的可执行文件，编译方式如下：

```
#Linux, macOS
cmake .
make
#Windows
cmake -G "MinGW make"
minge32-make
```

编译后运行程序的方法如下：

```
./morphing -s source -d destination [{--savepoints|--loadpoints} pointsFile]
[--edit] [-f frames] [-fps fps]
```

程序接收的各命令行参数说明如下：

- -s 与 –d 分别接进行图像变换的原图和目标图，是必须指定的参数。
- --savepoints 后接保存特征点数据的目标文件，使用该参数将会使程序调用 `selectPoints` 方法打开选择特征点的窗口，并在选择完特征点后保存至文件。
- --loadpoints 后接读取特征点数据的目标文件，程序将从该文件读取数据用于图像变换。
- --edit 选项仅在使用 --loadpoints 时有效，程序会读取特征点数据并打开

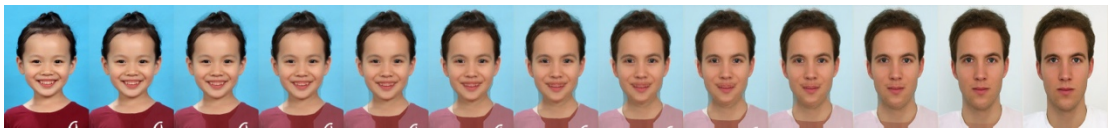
选择特征点的窗口使用户可以继续添加特征点。

- -f 后接帧数，代表中间帧的数量，默认为 11，最终输出的帧数量为中间帧数+2。
- -fps 后接生成的视频的 fps 值，默认为 25。

程序执行完图像变换后会将获得的所有帧从左到右合为一张图片存入 result.bmp，并将所有帧使用 ffmpeg 存入 result.mp4 文件。若需要 gif 文件则需要在程序执行完后再在命令行中执行以下命令获得：

```
ffmpeg -i result.mp4 result.gif
```

对题目要求的图片使用命令./morphing -s 1.jpg -d 2.bmp --loadpoints points.txt 执行程序。效果如下：



生成的 MP4 文件和 gif 文件也一并提交在 testimage 文件夹下。

其他的测试样例效果（位于 testimage/extraCases 文件夹下）：

./morphing -s kris.png -d lilem.png --loadpoints krisem.txt -f 3

（中间帧为 3 帧）