《数值分析与算法》

人脸图像扭曲变形报告

自 61 张嘉玮
2016011528
2018/11/10

目录

关键词：人脸变形，TPS，B 样条，图像插值，人脸特征点检测。

1. 需求分析和基本原理

本次人脸图像扭曲变形是基于人脸特征点进行的，因此，可将任务分为两个大的步骤，第一步找到特征点以及非特征点的映射关系，第二步使用插值等方法进行像素值的求解。而对应的需求也将包括以下两个方面。

1.1 映射方案

人脸特征点的关系会因不同的人脸以及像素点的位置等特点，使得原特征点到目标特征的相似性，要找到一种连续的映射过程。两种主要的映射方法是径向基函数映射和 B 样条映射。

径向基函数的主要步骤有下面三部：

A. 获得原人脸和目标人脸上具有代表性的位置点，即特征点。特征点之间一一对应。

B. 使用径向基函数和特征点建立原人脸和目标人脸之间的映射关系。

C. 利用得到的映射关系，找到每一个非特征点的在目标图当中的位置，并将像素赋予对应的位置。

而在生物图像变形方面，一般选取下面的径向基函数：

Duchon 的薄板样条：$g(r)=r^2\log r^2$

对应的方法称作薄板样条（TPS）算法。

此方案看似看似映射关系很难求解，但是思路明晰，目标明确。

相比而言，另一种使用较为广泛的变形方案时基于离散的B样条，对图像进行变形。

B样条变形有其自身的特点，特征点的移动，只影响其附近的点做响应的变化。相比而言，在薄板样条的方案当中，控制点的移动会影响所有像素点的移动。B样条的步骤主要如下：

A. 根据特征点的移动确定选定的控制点的移动量。

B. 根据每一个点相对于特征点的位置，确定其位移量。

C. 将这些位移作用到原图像的每一个点，便可以得生成的图像。

B样条方法看似简单，但是在基于特征点的变形上，由于特征特征点的位置不均匀，分散过于零散。在这样的情况下，部分特征点特别是眼睛等细节位置的变化会被忽略，因此存在很大的潜在误差。

以上两种方法共同存在的一个问题是，在变化过程中，会出现：

A. 原图当中有多个点对应到目标图上的同一个点，这样就会造成像素点的重叠，即信息的丢失。

B. 目标图像当中会有很大的空洞没有对应的原图的像素值对应过来，这样会导致最终图像上会出现“裂痕”或者“空洞”。

以上两种问题在实验过程当中，都发生了。而为了解决以上的问题，采取了下面的解决方案：

A. 映射方向不做改变，对于空缺的位置，找到和他最“最相似”的像素值填补，即用其周围的像素值进行填补。

B. 改变映射方向。以上都是正向的映射，而当选取方向映射时，就能很好的避免上面两种情况的发生。即对于目标图像上的每一个

点，通过反解之前得到的映射关系，就会得到其在目标图像当中的位置。进而用该位置周围的正数点，插值得到其对应的像素值。

在方案选择时，对两种变形方案都予以实现，发现薄板样条方案更加符合脸的形状，B 样条会出现扭曲走样以及变化不连续等问题，因此最终选择了薄板样条（TPS）的变形方案。

1.2 插值方案

之所以需要插值，主要在进行逆映射求解时，求解出来的位置很有可能是浮点数，这对于数字图像是没办法解决的，这是，只能根据其周围像素值的表现，使用插值的方法，求得其像素值。

在介绍这些插值方案之前，先定义下面的符号

定义：$(i+x,j+y)$为待求像素的坐标，i，j 为整数，x，y 为小数。

在数字图像处理当中，能够使用的插值方法有很多，基于本课程的主要任务便是插值，因此尝试和实现了多个插值算法，主要包括下面五种：

A. 最近邻插值

最近邻插值是较为简单的插值方案，不需要计算，只需将求得的坐标四舍五入即可。比如 x<0.5，y>0.5，则点$(i,j+1)$的像素值即为所求像素值。

此方法计算量小，但是会造成插值生成的图像“不连续”，在像素变化比较大的区域，出现明显的锯齿状等等。

B. 双线性插值

双线性插值又称作双线性内插。核心思想是在两个方向上分别

进行一次线性插值。基于以上的定义，则此方法可表示为：

$$f(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}$$

C. 双三次插值

双三次插值是一种较为复杂的插值方案，它能构造出比以上两种方案都要平滑的图像边缘。具体思路是，对于任意一点的像素值，通过其周围 16 个点的像素值，进行加权插值得到。

三次插值的基函数：

$$S(x) = \begin{matrix} 1-2*|x|^2+|x|^3, |x| \le 1 \\ 4-8*|x|+5*|x|^2-|x|^3, 1<|x|<2 \\ 0, otherwise \end{matrix}$$

由此式对得到的权值对图像像素值进行加权平均即可。

D. Lanczos 插值

Lanczos 算法是一种将对称矩阵通过正交相似变换成对称三角矩阵的算法。是在图像重采样和滤波当中使用较多的一种算法。当选择插值窗口为 4*4 时，Lanczos 窗口定义如下：

$$L(x) = \begin{cases} 1 & if: x=0 \\ \dfrac{2*\sin(\pi x)*\sin(\pi x/2)}{\pi^2 x^2} & if: -2 \le x \le 2\_and\_x \ne 0 \\ 0 & otherwise \end{cases}$$

插值公式：

$$S(x,y)=\sum_{i=1:4}\sum_{j=1:4}s_{ij}*L(x+i)*L(y+j)$$

E. 反距离插值 Inverse Distance Weighted

反距离插值算法的权重主要依赖反距离的幂值，主要思想是控制点对被插值点的影响与距离成反比。不同的幂次强调次影响的大小。

在本次作业当中，选取的幂次为 1，控制点为 4.

加权函数：

$$W_i=\frac{h_i^{-1}}{\sum_{j=1}^{4}h_j^{-1}}$$

$$h_i=\sqrt{(x-x_i)^2+(y-y_i)^2}$$

五种方法各有特点，其中最近邻计算量最小，但会出现阶梯等问题。双线性较最近邻，得到的图像相对平滑，当中没有考虑到图像的“变化率”。双三次能够有效的解决以上两种方法带来的弊端，但是由于其计算量较大，计算过程会较慢。Lanczos 插值在提高图像的分辨率上有较大的作用，次算法和双三次插值大致一致，但是插值速度和二次插值相仿。反距离插值的优点是可以根据不同的情况确定不同的幂次，这在一些地图的放缩当中使用较为广泛。为了对比五种算法的区别和各自的特点，对五种算法都进行了实现。

1.3 人脸特征点检测

为了实现的课可扩展性，对附加题进行了简单的实现。由于openCV人脸识别算法准确率受限，人脸识别这一环节使用了于仕祺老师开源的的人脸检测库，其识别速度以及识别准确率能够高于OpenCV自带的人脸检测算法。

2. 方案设计与实现

2.1 方案设计

程序运行效果图

注：目标图为张嘉玮

方案设计：

整体运行结果如上图所示。可以将界面分为上下两个板块。上面主要是对原图，目标图，以及生成的图片进行的显示。下面是一些操作的按钮。

图像的显示根据窗口大小进行了调整，但是存储以及进行操作的都是原图。生成的图为了观察其细节，没有对其进行放缩，使用者可以通过滑动观察细节和效果。

第二行左边是将读取到的或者识别到的特征点在图像显示，包括原图以及目标图。显示的前提是原图或者目标图已经读取，特征点也已经通过读取或者是别的方式获得；若不满足这两个特点，程序会有响应的提示。

第二行中间是一些选项：

【变形函数】这里除了有将原图变形到目标图的 TPS 选项，还包括了原图到原图的变换。增加这一选项的主演目的是检测和比较。而默认选项是变形到目标图。

【变形方向】是在进行 TPS 变形时，会有两个方向，正向和反向。默认选项是效果更好的反向变换。正向变形无需插值，即只有最近邻。

【图像调整】这里面的选项主要是考虑到 TPS 变形时，由于特征点的位置以及脸部的大小进行适当的调整。以获得想要的结果。

【逆向插值方式】插值方法有三个，分别是最近邻、双三次、双线性。

第二行右边是一些按钮，对应进行不同的操作。

【原图】对原图的操作主要有三个，读取图片，读取特征点，识别特征点。在读取文件时若中途退出或者读取失败，程序会有相应的提示。而特征点的识别前提是已经读取了图片。若尚未读取，会有相应的提示说明。

【目标图】对目标图的操作更原题一样，对应的容错性设置更原图相似。

【生成图】主要有两个操作：根据所选参数已经方式，生成对应的图像同时进行显示。该过程需要很多数据，若所需数据不足或者参数有误，都会给予相应的提示说明。对图像的保存会允许使用者选取保存路径，而图像名字则是随机生成的三位数来命名的。保存图像若失败，会有相应的提示说明。

2.2 关键环节实现

2.2.1 求解映射关系

TPS 求解映射关系是首先得到如下方程组：

$$\begin{bmatrix} K & P \\ P^T & O \end{bmatrix} * \begin{bmatrix} a & b \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix}$$

其中：　K 是由径向基函数得到的 60*60 矩阵。

$$P = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_{68} & y_{68} \end{bmatrix}$$

0 为 3*3 的 0 矩阵。a,b 是要求解的系数，为 71*1。x,y 分别为目标特征点的横纵坐标。

TPS 薄板样条变换的关键便是求解一个 71*71 维的线性方程组，由于数值很大，位数很多，一般的解方程方法很难求解这种大矩阵。在进行一番尝试和比较后，选择了高斯消元求逆矩阵的方法。基本原理如下：

$$A * x = y$$

$$A^{-1} * A * x = A^{-1} * \mathrm{y}$$

$$x = A^{-1} * y$$

便可以求出系数矩阵。

正弦变换和方向变换的过程对称，求解线性方程组的方法一

样。

高斯消元虽然对大矩阵较好，但是只符合有唯一解的情况。当特征点选取的不好，方程有两个及以上数量个解时，此方法就会无效。本程序对这种情况也做了容错处理，会提示使用者对目标图或者原图做出调整。

2.2.2 求解生成图

生成图的求解主要在于求解每一像位置的像素值。

在正向求解时，会出现空洞或者裂痕。而使用最近领的思想，将这些位置用其周围的像素值进行赋值。为了使得图像不出现“线条”，交替使用其左边或者上边的像素值进行替代。

对于反向求解，首先找到目标图坐标在原图当中对应的做坐标，然后根据所选择的插值方法，进行插值。然后将插值结果付给对应的坐标。

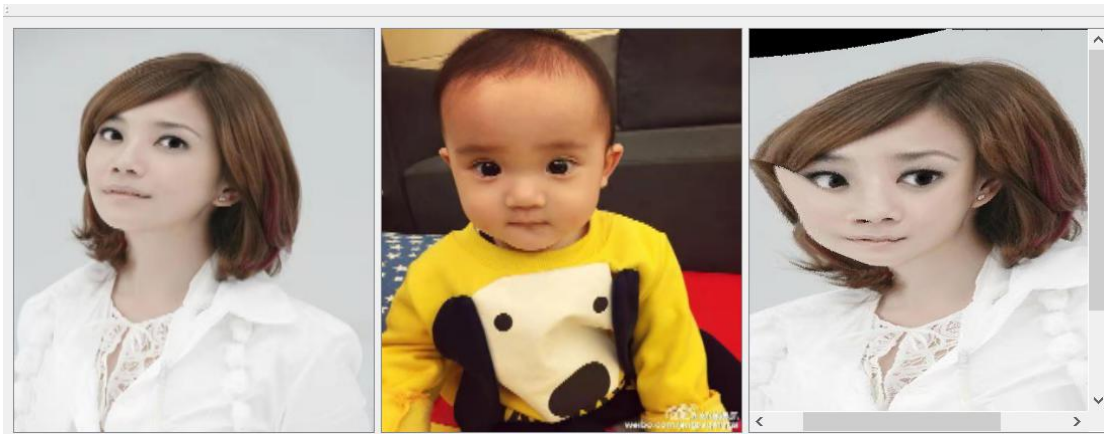
2.2.3 特征点识别

在使用了于仕祺老师的开源库之后，可以较为容易检测到目标图的人脸特征点位置。

库连接【https://github.com/ShiqiYu/libfacedetection】

3. 关键环节演示

A. 正向最近领插值



B. 逆向最近领

C. 逆向双线性

D. 逆向双三次

E. 逆向 Lanczos 插值

F. 反距离插值

4. 误差分析

4.1 舍入误差

舍入误差主要是由于计算机存储数据位数所限导致。在进行计算时，使用的类型皆为 double 型，有效数字有 15 位，对于像素值，最大为三位数，则由于 double 型数据长度导致的输入误差为 $0.5*10^{-12}$ 。

另一个舍入误差则来自像素值本身存储导致的误差。由于像素值本身只能是整数，在幅值时进行了四舍五入，这一环节的舍入误差为0.5。

将上面两个环节的误差相加，可得整体的舍入误差为0.5。

有这个情况可以看出，数字图像的本身的性质，导致了舍入误差在0.5。而计算机在计算时导致的舍入误差，对结果的影响很小。

4.2 方法误差

4.2.1 最近领插值

在最近邻插值时，点坐标采用了四舍五入的近似关系。误差为：

$$f^*(x,y)-f(x_1,y_1)=\frac{\partial f(x,y)}{\partial x}(x-x_1)+\frac{\partial f(x,y)}{\partial y}(y-y_1)$$

其中，(x_1,y_1)为(x,y)点四舍五入后的坐标。

两边取绝对值，得：

$$|f^*(x,y)-f(x_1,y_1)|\leq\left|\frac{\partial f(x,y)}{\partial x}\right|*|x-x_1|+\left|\frac{\partial f(x,y)}{\partial y}\right|*|y-y_1|$$

根据四舍五入的性质，得：

$$|x-x_1|\leq\frac{1}{2}$$

$$|y-y_1|\leq\frac{1}{2}$$

设：$M_1=\left\{\frac{\partial f(x,y)}{\partial x},\frac{\partial f(x,y)}{\partial y}\right\}$，则误差满足：

$$\left|R(x,y)\right| \le M_1$$

根据数字图像的特点，其每一个通道的值在 0 到 255 之间，则 $M_1 = 255$。即最近邻插值的误差上限为：$\left|R(x,y)\right| \le 255$。

4.2.2 双线性插值

双线性插值的基本原理是进行了三次一位的线性插值。在进行分析之前，先设 $\max\left\{\frac{\partial^2 f(x,y)}{\partial x^2}, \frac{\partial^2 f(x,y)}{\partial y^2}\right\} = M_2$，$\lfloor x \rfloor = i, \lfloor y \rfloor = j$。

第一步，在 X 方向进行插值，得到（x,j）点像素值。

$$R_x \le \frac{1}{2!}\left|\frac{\partial^2 f(x,y)}{\partial^2 x}\right| * \left|(x-i)*(x-i-1)\right\|$$

$$R_x \le \frac{1}{2} * M_2 * \frac{1}{4}$$

$$R_x \le \frac{1}{8} * M_2$$

第二步，在 X 方向进行插值，得到（x,j+1）点像素值。

同第一步方法，可得:

$$R_y \le \frac{1}{8} * M_2$$

第三步，沿 Y 方向进行插值，得到（x,y）点像素值。

$$R_{xy} \le \frac{1}{2!}\left|\frac{\partial^2 f(x,y)}{\partial^2 y}\right| * \left|(y-j)*(y-j-1)\right|$$

$$R_{xy} \le \frac{1}{2} * M_2 * \frac{1}{4}$$

$$R_{xy} \le \frac{1}{8} * M_2$$

则双线性对应的误差为：

$$\left|R(x,y)\right| \le M_x + M_y + M_{xy} = \frac{3}{8} * M_2$$

4.2.3 双三次插值

同双线性插值，双三次插值可分解为两个方向的一位插值。同样设：$\max\left\{\left|\frac{\partial^4 f(x,y)}{\partial x^4}\right|,\left|\frac{\partial^4 f(x,y)}{\partial y^4}\right|\right\} = M_4$，$\lfloor x \rfloor = i, \lfloor y \rfloor = j$。

双三次插值等同于在 X 方向做四次一维插值，在 Y 方向做一次一维插值。

X 方向的第一次插值的截断误差：

$$R_{x1} \le \frac{1}{4!} M_4 * (x-i+1)*(x-i)*(i+1-x)*(i+2-x)$$

$$R_{x1} \le \frac{1}{24} * M_4 * \frac{9}{16}$$

$$R_{x1} \le \frac{3}{128} * M_4$$

同理的 X 方向的另外三次插值带来的截断误差：

$$R_{x2} \le \frac{3}{128} * M_4,\ R_{x3} \le \frac{3}{128} * M_4,\ R_{x4} \le \frac{3}{128} * M_4$$

最后一次在 Y 方向的插值带来的误差：

$$R_y \le \frac{3}{128} * M_4$$

则双三次带来的总截断误差为：

$$R(x,y) \le \frac{3}{128} * M_4 * 5$$

$$R(x,y) \le \frac{15}{128} * M_4$$

4.2.4 Lanczos 插值算法

误差分析近似同双三次插值，同为：

$$R(x,y) \le \frac{3}{128} * M_4 * 5$$

$$R(x,y) \le \frac{15}{128} * M_4$$

4.2.5 反距离插值 Inverse Distance Weighted

误差分析近似同双线性插值：

$$\left|R(x,y)\right| \le M_x + M_y + M_{xy} = \frac{3}{8} * M_2$$

4.3 插值方案的比较

五种插值方法各有各的特点以及缺点。最近邻插值速度快，但是图像在放大之后会看到明显的锯齿状过渡，同时，由误差分析可得，最近邻对应的阶段误差也是很大的。

双向性插值能较好的避免锯齿的出现，同时计算速度也适中。相比而言，双向性虽然误差小，能避免锯齿的出现，但是其计算速度慢，这在一些速度要求高的应用上是不能接受的。

Lanczos 插值算法和反距离插值 Inverse Distance Weighted 在实际应用当中更多的用于图像放缩和滤波当中，以提高图像 的分辨率。而像这种用于单点的插值，其实更多的使用上面介绍的三种线性插值算法。

正向

最近邻

双线性

双三次

反距离

Lanczos

通过观察不难看出，在虹膜附近，最近领的对比度比较高，但过渡不均匀。相比而言，双线性和双三次的过渡比较均匀，但是虹膜附近的对比度有所损失。另外反距离的效果是由于上面三种的，不仅对比度得到了保留，另外过渡也很均匀。Lanczos 插值方法过渡均匀，但出现了纹波现象，这也是由于其插值过程需要的点更多，并且根据其权值，有增加对比度的作用，进而产生了纹波。就眼睛这一位置来看，反距离插值与另外三种插值相比，有更好的插值效果。

5. 问题及收获

5.1 B 样条花费大量时间实现，但是效果不尽人意

在大作业的第一周，在对两种方法进行分析对比之后，感觉 B 样条更符合变与不变的思路，因此选取了 B 样条。整个第二周都在搞 B 样条，但效果很差，只好重新考量自己的方案，选择了 TPS 进行实现。最后也发现，B 样条适合一些小的变化，先一张脸有 68 个特征点，而且分布不均匀，这对于 B 样条其实是不太适合的。相反，TPS 在理论上，可以有任意多个控制点。

5.2 TPS 解方程

由于理论上方程时又觉得，因此便直接使用 Gaussion 消元的方法求解，结果总是 NaN。后来发现是数值超过了 double 型数据的范围，这也是第一次遇到这种超范围的情况，然我对计算机离散有限位存储数据有了新的认识。之后尝试了 LU 分解，同样失败。在和同学交流后，发现像这种比较大的求解，需要通过矩阵求逆，然后又回归到

Gaussion 消元求逆上，进而得出了想要的结果。

5.3 收获

首先是要多交流，刚开始自己单独搞，思路单一，遇到困难跳不出来，和同学交流后，便有了很多新的想法，受益匪浅。

锻炼了自己的编程能力。由于大二一年基本上是在用 python 和 MATLAB，导致自己对 C++的严谨的代码结构遗忘了许多。而这次大作业，让我对 C++得到了充分的回顾。同时也感受到了 C++运算速度的强大。

对离散的插值问题有了更加深入的理解和掌握。

6. 参考文献及资料

[1]. 李庆杨,王能超,易大义. 数值分析[M].第五版.北京：清华大学出版社.2008.
[2]. 陈秋燕,殷福亮.基于 博板样条的人脸变形计数[J].通信理论与信号处理学术年会论文集.2010.
[3]. Mono_玉鹏.如何使用于开源库 ibfacedetection[EB/OL].
https://blog.csdn.net/mono_1032290547/article/details/78912548.2017.