数值分析与算法

大作业一

班级：自 71

姓名：屈晨迪

学号：2017010928

1 需求分析

1.1 必做

本次大作业的目标是图像变形。在必做任务中，要求对给定的图片做旋转扭曲和畸变校正两种变形，并通过三种不同的方式——最近邻、双线性和双三次对图片进行插值，生成变形后的图像。程序以 GUI 呈现，用户可以调整旋转、畸变参数，选择不同插值方式，分析误差，对比三种插值函数的性能。

1.2 选做

选做任务要求利用 TPS 网格变形，对给定图像进行扭曲变换。程序提供包含 9 张人脸图片和面部关键点的库，用户随意选取其中两张图片，分别作为待修改人脸和目标人脸，输出一张以目标人脸关键点为控制点的修改后图片，实现人脸变形的效果，同样可以通过最近邻、双线性和双三次三种方式进行插值。程序以 GUI 形式呈现。

2 方案基本原理

实现图像变形有两种基本思路，正向插值和反向插值，考虑到算法复杂度问题，本次大作业选用复杂度较低的反向插值方法。首先将目标图片中的点集通过变形函数映射得到在原始图像上的坐标，而此坐标不是整型的像素点，需要通过插值，获取此坐标的像素值，再赋值给目标图片的对应点，从而生成目标图片。

下面针对变形函数和插值函数分别进行介绍。

2.1 变形函数

本次大作业共用到了三种变形函数——旋转扭曲、畸变矫正和 TPS 函数。下述讨论均使用(x', y')表示目标图像的整型坐标点，(x, y)表示原始图像中的浮点型坐标点。

（1）旋转扭曲

旋转扭曲首先应设定旋转半径$Radius$和最大旋转角度α_{max}，遍历目标图像中的像素点，计算到中心点的距离$Distance$，若在旋转半径之内，进行坐标变换，计算旋转角度

$$\alpha = \alpha_{max} \times \frac{Radius - Distance}{Distance}$$

计算对应点坐标

$$\begin{cases} x = x' \cdot cos\alpha - y' \cdot sin\alpha \\ y = x' \cdot sin\alpha + y' \cdot cos\alpha \end{cases}$$

输出(x, y)，示意图如图 1 所示。

（2）畸变矫正

畸变矫正分正畸变和负畸变，设定畸变半径$Radius$，遍历目标图像中的像素点，计算到中心点的距离$Distance$，有以下对应点坐标计算公式

$$\begin{cases} x = [\frac{Radius}{Distance} \cdot \arcsin(\frac{Distance}{Radius})] \cdot x' \\ y = [\frac{Radius}{Distance} \cdot \arcsin(\frac{Distance}{Radius})] \cdot y' \end{cases}$$

此时为正畸变，负畸变只需将系数取倒数即可，注意到公式中有$arcsin$，即实现中$\frac{Distance}{Radius}$有一定的范围限制[1]。输出(x, y)，畸变示意图如图 2.

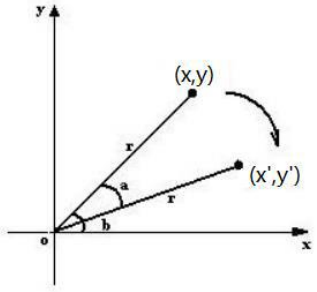

图 1 旋转扭曲示意图

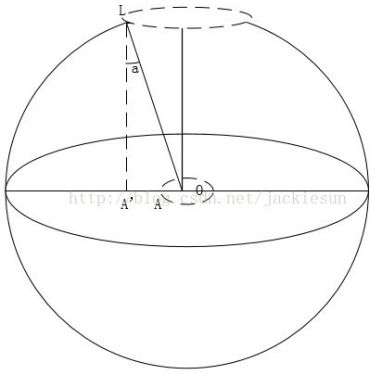

图 2 畸变矫正示意图

（3）TPS 函数

TPS 即薄板样条插值（Thin Plate Spline）是一种常见的插值模型，目标是寻找一个通过所有控制点的光滑曲面$f(x, y)$，使得能量函数I_f最小。

$$I_f = \iint_{R^2} \left(\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 f}{\partial y^2} \right)^2 \right) dxdy$$

设给定的 n 个控制点$P_1 = (x_1, y_1), P_2 = (x_2, y_2), \dots, P_n = (x_n, y_n)$，目标特征点$P_1' = (x_1', y_1'), P_2' = (x_2', y_2'), \dots, P_n' = (x_n', y_n')$，记

$$K = \begin{bmatrix} 0 & U(r_{12}) & \cdots & U(r_{1n}) \\ U(r_{21}) & 0 & \cdots & U(r_{2n}) \\ \cdots & \cdots & \cdots & \cdots \\ U(r_{n1}) & U(r_{n2}) & \cdots & 0 \end{bmatrix}$$

$$P = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \cdots & \cdots & \cdots \\ 1 & x_n & y_n \end{bmatrix} \qquad L = \begin{bmatrix} K & P \\ P^T & \mathbf{0} \end{bmatrix}$$

[1] 可将 arcsin 用泰勒展开的前两项替换，则 Distance 的范围无限制

其中r_{ij}为两个控制点P_i, P_j之间的欧氏距离，径向基函数U有如下定义

$$U(r)=\begin{cases} r^2\log\left(r^2\right), r\neq 0 \\ 0, \quad r=0 \end{cases}$$

记

$$V=\begin{bmatrix} x_1' & x_2' & \cdots & x_n' \\ y_1' & y_2' & \cdots & y_n' \end{bmatrix} \qquad Y=\left(V\begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}\right)^T$$

则有 TPS 函数

$$f(x,y)=\left[f_x(x,y), f_y(x,y)\right]^T=\boldsymbol{a}_1+\boldsymbol{a}_x x+\boldsymbol{a}_y y+\sum_{i=1}^{n}\boldsymbol{w}_i U(|P_i-(x,y)|)$$

其中$\boldsymbol{a}_1, \boldsymbol{a}_x, \boldsymbol{a}_y, \boldsymbol{w}$为线性方程组$L\left[\boldsymbol{w}_1, \cdots, \boldsymbol{w}_n, \boldsymbol{a}_1, \boldsymbol{a}_x, \boldsymbol{a}_y\right]^T=Y$的解。基于该模型，对任意给定的一组坐标$(x,y)$，可以得到映射后的坐标$f(x,y)$.

2.2 插值函数

本次作业要求运用三种插值方法——最近邻、双线性、双三次，分别介绍如下。假设目标图像(x',y')经变换后得到原图像坐标(x,y).

（1）最近邻

最近邻插值即选取距离(x,y)最近的像素点的像素值。

$$f'(x',y')=f([x+0.5],[y+0.5])$$

其中$[x]$表示向下取整。

（2）双线性

双线性插值使用到坐标点$P=(x,y)$周围四个像素点的像素值信息，如图 3，设$Q_{11}=(i,j)$，有其他三个点$(i+1,j),(i,j+1),(i+1,j+1)$，插值函数

$$f'(x',y')=f(i+u,j+v)=[1-u\ \ u]\begin{bmatrix} f(i,j) & f(i,j+1) \\ f(i+1,j) & f(i+1,j+1) \end{bmatrix}\begin{bmatrix} 1-v \\ v \end{bmatrix}$$

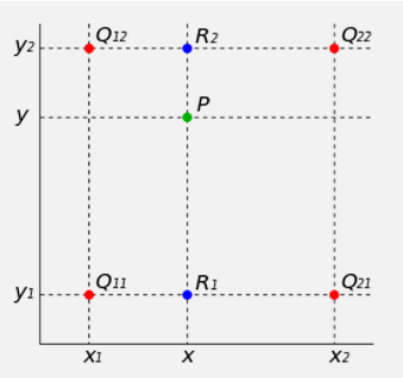

图 3 双线性插值

（3）双三次

双三次用到了坐标点(x,y)周围 16 个像素点的像素值信息，考虑(x,y)所在的4×4网格，令

$$\begin{cases} u=x-[x]=x-i \\ v=y-[y]=y-j \end{cases}$$

其中[]表示向下取整，定义

$$A = [S(u+1)\ \ S(u)\ \ S(u-1)\ \ S(u-2)]$$
$$C = [S(v+1)\ \ S(v)\ \ S(v-1)\ \ S(v-2)]$$
$$B = f(i-1{:}i+2, j-1{:}j+2)$$

其中

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

则

$$f'(x', y') = f(i+u, j+v) = ABC^T$$

3 方案设计

本程序为在 VS2012 平台上使用 C#语言编写的 Windows 应用程序，编译和运行于.NET Framework 4.5.

3.1 必做任务

（1）算法流程

程序开始，首先读入待变形图片，获取图片长和宽，新建一张与原图相同大小的图片。若选择进行旋转扭曲，获取用户设置的旋转半径和最大旋转角度，遍历新建图片中每一个像素点，用相应的变形函数计算其对应的原始图片中的坐标，利用选择的插值函数获取该点像素值，获得新的变形后的图片，展示在界面上；若选择进行正畸变/负畸变，获取用户设置的畸变半径，之后步骤同旋转扭曲。

（2）GUI 设计

初始用户界面如图 4 所示，默认的变换方式为旋转扭曲，插值方法为最近邻插值，用户可以选择其他变换方式和插值方法，并通过滚动条调整相应参数，参数选择完毕后，点击“变换“按钮，生成变形后图片，如图 5 所示。

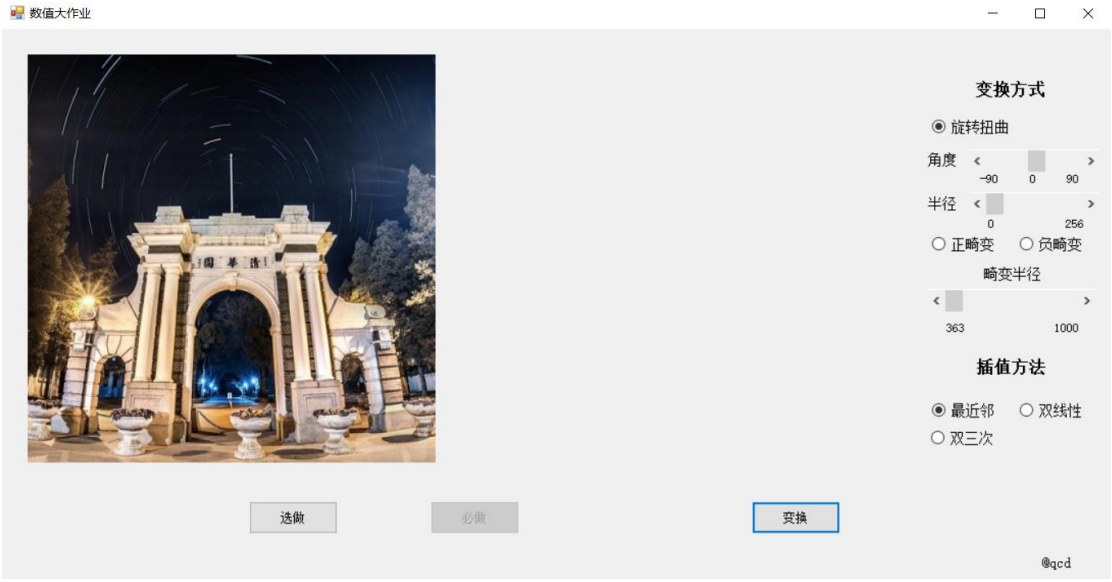

图 4 必做初始用户界面

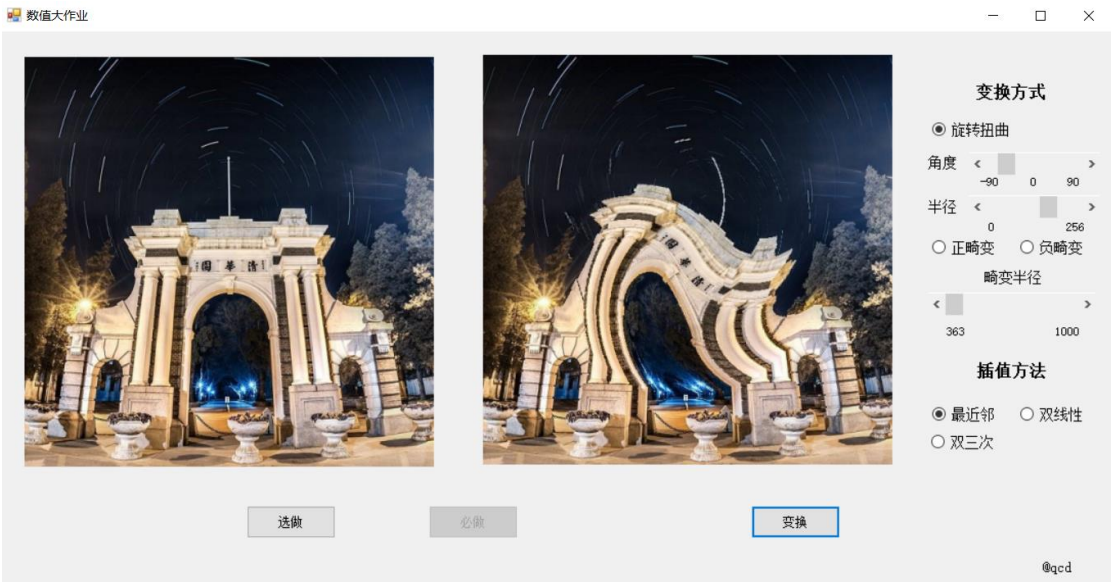

图 5 旋转变换后界面

必做界面下“必做”按钮被锁定，用户可以点击“选做”按钮切换至选做任务界面。

3.2 选做任务

（1）算法流程

程序开始，首先读入待变形图片 src、目标图片 dest 以及其特征点坐标信息，其中需要将坐标信息从科学计数法转成整型数，存入$Point2d$类 List，获取两张图片的长和宽，新建一张和目标图片相同大小的图片。

接着需要求出 TPS 函数的各项参数，以二维数组的形式生成并存储矩阵K, P, L, V, Y，求解方程组$L[\boldsymbol{w}_1, \cdots, \boldsymbol{w}_n, \boldsymbol{a}_1, \boldsymbol{a}_x, \boldsymbol{a}_y]^T = Y$，有$[\boldsymbol{w}_1, \cdots, \boldsymbol{w}_n, \boldsymbol{a}_1, \boldsymbol{a}_x, \boldsymbol{a}_y]^T = L^{-1}Y$，即要求出 L 的逆矩阵，利用初等变换的方法求逆矩阵[2]，在 L 的右边拼接一个单位阵 I，将 L 变换为单位阵，对 I 进行同样的变换操作，最终的 I 即为L^{-1}，与 Y 相乘即获得 TPS 的函数参数。

遍历新建图片中每一个像素点，用相应的变形函数计算其对应的原始图片中的坐标，若坐标在待变形图片的坐标范围内，利用选择的插值函数获取该点像素值，若不在坐标范围内，将像素值设为 0，显示黑色。展示生成的新图片。

（2）GUI 设计

选做初始界面如图 6 所示，用户可以在右侧下拉菜单中选择希望变换的图片，选择好后，点击“确定”，图片展示在界面上，选择插值方法，点击“变换”，在待变形图片处展示变换后的新图片，如图 7 所示。

此界面下“选做”按钮被锁定，用户可以点击“必做”按钮切换到必做界面。

[2] 也可利用其他方法如算术余子式求逆，但考虑矩阵阶数较大，初等变换的方法效率较高

图 6 选做初始界面

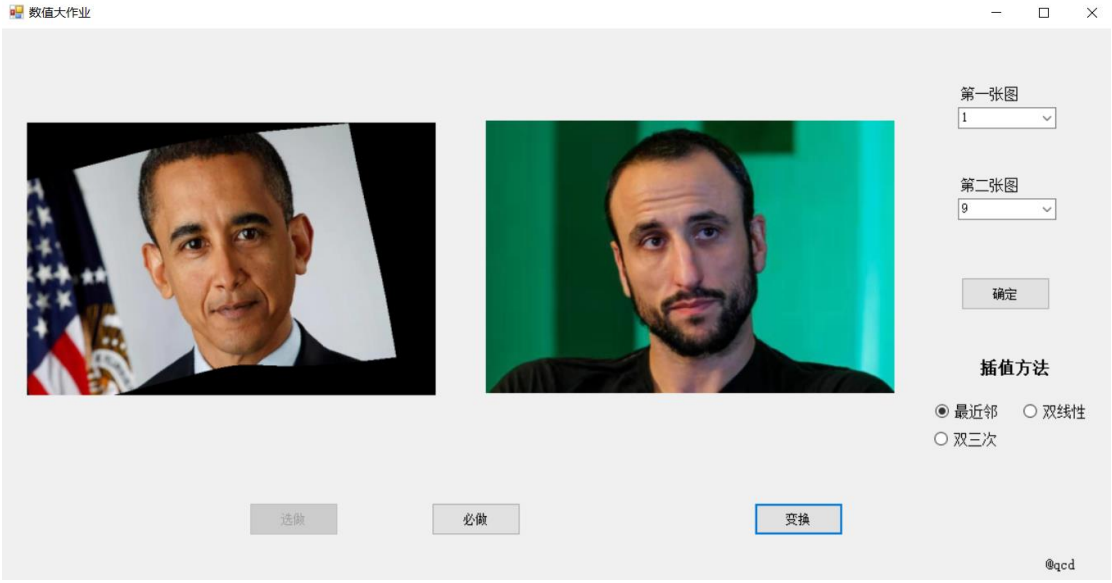

图 7 将 pic1 变换为 pic9 结果

4 误差分析

一般而言，误差来源分为四种，观测误差、模型误差、方法误差和舍入误差。在此案例中，我们假定给出的数学模型是合理的，故忽略不计；而数学模型中由观测物理量，如温度、长度、电压等产生的误差称为观测误差，本实验中不存在，且上述两种误差通常在“数值分析”中不予讨论[3]。以下重点讨论方法误差和舍入误差。

4.1 方法误差

当数学模型不能得到精确解时，通常用数值方法求其近似解，其间存在的误差称为方法误差。本实验中方法误差主要存在于插值环节，各插值函数在 2.2 中给出。

[3] 《数值分析于科学计算引论》，李庆杨 王能超 易大义编，P4

（1）最近邻插值

在 x,y 两个方向上分析误差，有

$$|R| = \left|R_x + R_y\right| \le |R_x| + |R_y|$$

其中

$$|R_x| \le \frac{M_1}{1!}|\Delta x| \le M_1|x - (x - [x])| \le \max\left|\frac{\partial f}{\partial x}\right| \cdot \frac{1}{2}$$

$$|R_y| \le \frac{M_1}{1!}|\Delta y| \le M_1|y - (y - [x])| \le \max\left|\frac{\partial f}{\partial y}\right| \cdot \frac{1}{2}$$

其中[x],[y]均为向下取整，有

$$|R| \le \frac{1}{2} \cdot \left(\max\left|\frac{\partial f}{\partial x}\right| + \max\left|\frac{\partial f}{\partial y}\right|\right)$$

（2）双线性插值

在 x 和 y 两个方向上分析，双线性相当于做两次一次线性插值，设映射得到的坐标$(i+u, j+v)$，周围四个格点的坐标分别为$(i,j)(i,j+1)(i+1,j)(i+1,j+1)$，在 x 方向上有，

$$R_x = f_x - L_x$$

$$|R_x| \le \frac{M_2}{2!}|u(1-u)| \le \frac{1}{2} \cdot \max\left|\frac{\partial^2 f}{\partial x^2}\right| \cdot \frac{1}{4}$$

同理在 y 方向上

$$\left|R_y\right| \le \frac{M_2}{2!}|v(1-v)| \le \frac{1}{2} \cdot \max\left|\frac{\partial^2 f}{\partial y^2}\right| \cdot \frac{1}{4}$$

即有

$$|R| \le |R_x| + \left|R_y\right| \le \frac{1}{8} \cdot \left(\max\left|\frac{\partial^2 f}{\partial x^2}\right| + \max\left|\frac{\partial^2 f}{\partial y^2}\right|\right)$$

（3）双三次插值

在 x 和 y 两个方向上分析，双三次插值相当于二维的三次样条插值，已知一维三次样条插值的方法误差为

$$|R| \le \frac{5}{384} M_4 \left(\frac{b-a}{n}\right)^4$$

则在 x 和 y 上分别有

$$\begin{cases} |R_x| \le \dfrac{5}{384} \cdot \max\left|\dfrac{\partial^4 f}{\partial x^4}\right| \\ \left|R_y\right| \le \dfrac{5}{384} \cdot \max\left|\dfrac{\partial^4 f}{\partial y^4}\right| \end{cases}$$

即有

$$|R| \le |R_x| + \left|R_y\right| \le \frac{5}{384} \cdot \left(\max\left|\frac{\partial^4 f}{\partial x^4}\right| + \max\left|\frac{\partial^4 f}{\partial y^4}\right|\right)$$

（4）对比分析

从上述对三个插值函数的误差分析可以看出，针对同等问题时，双三次的方法误差最小，但计算量也相应最大，在应用时应结合实际问题选择。

三种插值方法的比较见表 1，输出图像结果对比见图 8.

表 1 三种插值方法对比

插值方法	像素点信息量	误差表达式	运算速度	视觉效果
最近邻	1 个	$\|R\| \le \frac{1}{2} \cdot (\max\left\|\frac{\partial f}{\partial x}\right\| + \max\left\|\frac{\partial f}{\partial y}\right\|)$	较快	有较明显像素断点
双线性	4 个	$\|R\| \le \frac{1}{8} \cdot (\max\left\|\frac{\partial^2 f}{\partial x^2}\right\| + \max\left\|\frac{\partial^2 f}{\partial y^2}\right\|)$	适中	较平滑
双三次	16 个	$\|R\| \le \frac{5}{384} \cdot (\max\left\|\frac{\partial^4 f}{\partial x^4}\right\| + \max\left\|\frac{\partial^4 f}{\partial y^4}\right\|)$	较慢	平滑

(a) 最近邻

(b) 双线性

(c) 双三次

图 8 三种插值图像结果

4.2 舍入误差

在使用计算机做数值计算时，由于计算机的字长有限，实验存在舍入误差。

• 原始数据在计算机上表示时会产生舍入误差。本实验中，计算机在读入图象时，会将图像中的像素值离散化，转换成整数值存储，从而产生$|R_1| \leq \frac{1}{2}$的舍入误差；

• 计算过程中可能产生舍入误差。本次实验中计算均采用 double 类型双精度浮点数，每次运算都会会产生1×10^{-16}的舍入误差，将整个程序中的各项四则运算、矩阵运算的运算次数做一统计，大约在 10-100 次之间，即可认为误差大约在10^{-15}量级左右；另外，在获得每个像素点的 double 类型像素值后，最终要将 double 强制转换为 byte 类（8 位无符号整数）输出结果图像，该转换有$\frac{1}{2}$的舍入误差，是10^{-1}数量级，与10^{-15}相比占主导。因此我们可以近似认为该部分舍入误差$|R_2| \leq \frac{1}{2}$.

综上所述，总舍入误差$|R| = |R_1 + R_2| \leq |R_1| + |R_2| \leq 1$.

5 总结与反思

本次大作业让我充分了解了图像变形的方法和流程，掌握了反向插值的应用，更加深刻地理解了三种插值函数——最近邻、双线性和双三次的原理、计算和误差分析的方法，对比了其性能特点，并且复习了拉格朗日插值，将课堂上学习的误差分析方法运用到了实践中；在选做任务中，我学习了 TPS 变形，自行编写代码实现了包括乘法、求逆在内的多种矩阵运算。另外，我还锻炼了利用 C#设计编写 GUI 的能力，一个用户友好的界面是程序重要的一部分；学习了在 VS 中配置、调用 opencvsharp 库进行图片的读取。

在程序设计的过程中，我一开始使用 opencv 的 mat 类实现运算中的各个矩阵，但发现调用 mat 类的时间较慢，改用二维数组存储和表示，程序运行速率有了明显的提升。

本次大作业让我收获良多，最后，感谢老师和助教对我的帮助！