

《数值分析与算法》

大作业 1 图像扭曲变形

项目报告

班级：自 45

姓名：林子坤

学号：2014011541

目录

一、项目简介	4
二、程序属性	4
三、程序功能与使用方法	4
3.1 选择图片	4
3.2 选择图像变形方式	5
3.3 设置参数与属性	5
3.3.1 设置旋转扭曲参数	5
3.3.2 设置图像畸变属性	5
3.3.3 设置 TPS 网格变形属性	5
3.4 选择插值方法	5
3.5 处理图片	6
3.6 保存图片	6
四、程序输出效果	6
4.1 旋转扭曲	6
4.2 图像畸变	7
4.3 TPS 网格变形	7
五、需求分析	8
六、方案原理与设计	8
6.1 插值方法	8
6.1.1 最近邻插值	8
6.1.1.1 基本原理	8
6.1.1.2 误差分析	8
6.1.1.3 算法设计	8
6.1.2 双线性插值	8
6.1.2.1 基本原理	8
6.1.2.2 误差分析	9
6.1.2.3 算法设计	9
6.1.3 双三次插值	9
6.1.3.1 基本原理	9

6.1.3.2 误差分析	10
6.1.3.3 算法设计	10
6.2 扭曲变形方式	10
6.2.1 旋转扭曲	10
6.2.1.1 基本原理	10
6.2.1.2 算法设计	11
6.2.2 图像畸变	11
6.2.2.1 基本原理	11
6.2.2.2 算法设计	11
6.2.3 TPS 网格变形	12
6.2.3.1 基本原理	12
6.2.3.2 算法设计	13
七、实验结果与分析	13
7.1 旋转扭曲	13
7.2 图像畸变	14
7.2.1 中心外凸	14
7.2.2 中心内凹	14
7.3 TPS 网格变形	15
7.4 实验结果分析	15
八、程序框图	16
九、项目总结与思考	17

大作业 1 图像扭曲变形

项目报告

林子坤

(自动化系 自 45 班 2014011541)

一、项目简介

本次大作业项目通过不同图像扭曲变形方法，并结合不同的插值方式进行图像处理。其中所使用的图像变形方法包括：1、旋转扭曲，2、图像畸变，3、TPS（薄样条插值）图像变形；所使用的插值方法包括：1、最近邻插值，2、双线性插值，3、双三次插值。

二、程序属性

本程序使用 C#语言编写，使用 Visual Studio 2013 Community 编译器。支持的运行环境是 Windows 7 至 Windows 10 各版本。

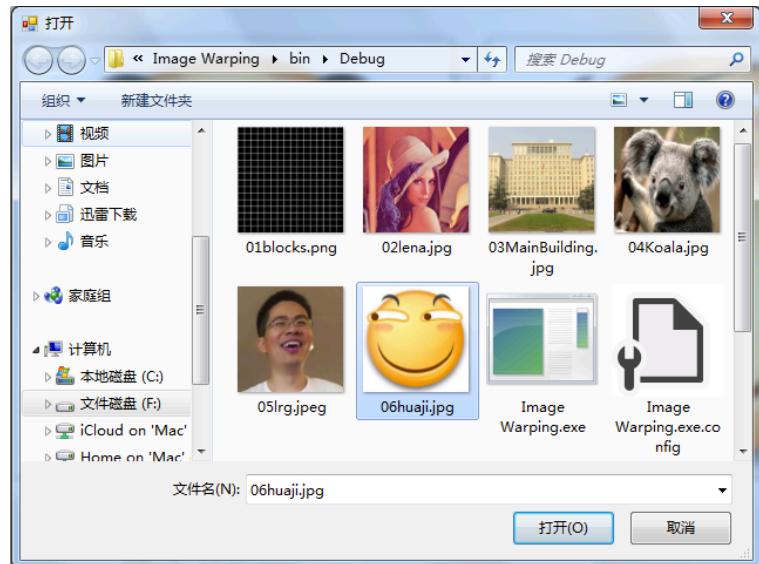
程序界面如下图所示：



三、程序功能与使用方法

3.1 选择图片

使用时，点击“选择图片”按钮，将弹出选择图片的对话框，从中选取图片并点击“打开”按钮将图片导入程序。



3.2 选择图像变形方式

使用时，点击“扭曲变形方式”一栏中的“旋转扭曲”、“图像畸变”、“TPS 网格变形”三者之一，选择不同的图像变形方式对图像进行处理。

3.3 设置参数与属性

3.3.1 设置旋转扭曲参数

在旋转扭曲变形方式中，需要设置最大旋转半径和旋转角度两个参数。其中，最大旋转半径的设置不得超出图像尺寸的一半，旋转角度的设置可在 -360° 至 360° 中选择，每次选择可递增 5° 。其中，负角度代表逆时针旋转扭曲，正角度代表顺时针旋转扭曲。

3.3.2 设置图像畸变属性

在图像畸变变形方式中，需要选择图像外凸或者内凹的属性，并设置图像的畸变程度。使用时，点击“图像畸变属性”一栏中的“中心外凸”、“中心内凹”二者之一，并设置1~7之间的畸变程度。

3.3.3 设置 TPS 网格变形属性

在TPS网格变形方式中，需要设置控制点的组数（注意：控制点组数 >6 时，运算时间可能会特别长，请谨慎）。若需要清除特征点，可点击“清除特征点”按钮，则特征点信息将被全部清除。

随后，需要在原始图像上点击选择与控制点组数相符的特征点对。其中，红色点代表控制点，蓝色点代表目标点。各组控制点与目标点以绿色线段连接。

3.4 选择插值方法

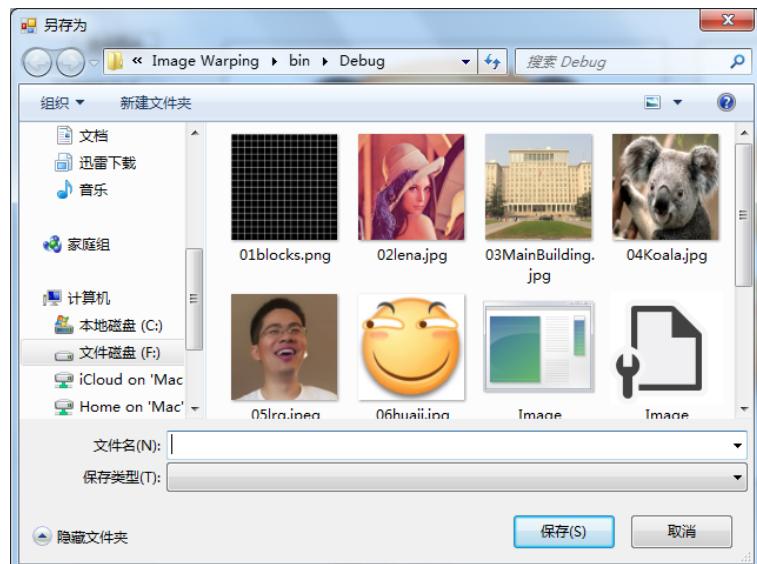
使用时，点击“插值方法”一栏中的“最近邻插值”、“双线性插值”、“双三次插值”三者之一，选择不同的插值方式对处理后的图像进行插值处理。

3.5 处理图片

属性设置完毕后，点击下方的“处理图片”按钮进行图片处理。

3.6 保存图片

处理图片后，点击最下方的“保存图片”按钮进行图片保存。使用时，点击“选择图片”按钮，将弹出选择图片的对话框，从中选取图片并点击“保存”按钮将图片保存至相应位置。



四、程序输出效果

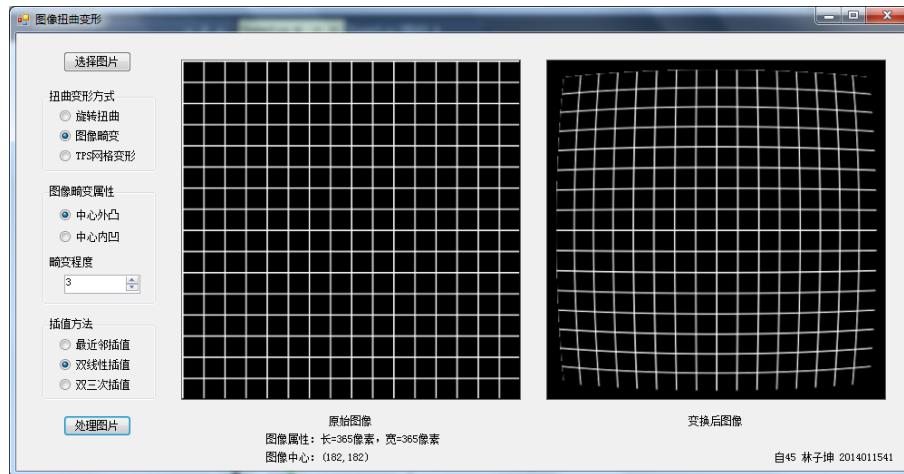
4.1 旋转扭曲

设置最大旋转半径为 180，旋转角度为-120°（逆时针旋转扭曲 120°）。

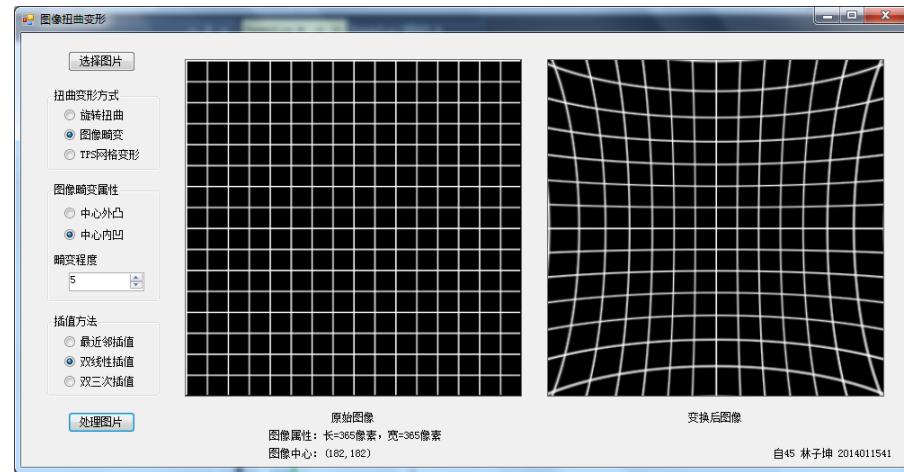


4.2 图像畸变

设置畸变程度为 3 的图像外凸。

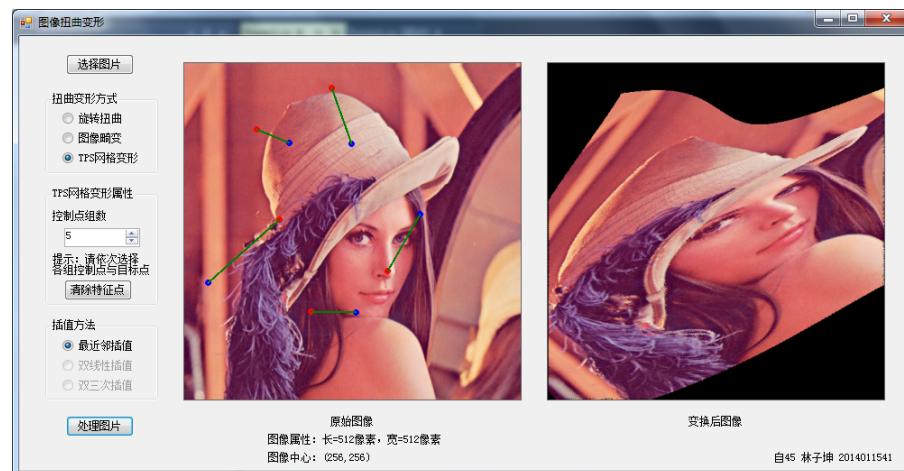


设置畸变程度为 5 的图像内凹。



4.3 TPS 网格变形

选择 5 组如图所示的特征点（注：红色代表控制点，蓝色代表目标点）。



五、需求分析

本次大作业的目的是通过不同图像扭曲变形方法，并结合不同的插值方式进行图像处理。为了实现以上需求，本人设计 3×3 的处理方式，即 3 种图像扭曲变形方法均可以通过 3 种插值方式进行处理实现，这也要求 3 种插值方式的设计具有普适性。

在 3 种图像扭曲变形方法中，TPS 网格变形与前面两种变形方式不同，它的变形结果取决于我们选取的点，因此它的变形结果具有一定的随机性，也有可能出现一些难以预料的问题。在方案设计过程中，也需要通过对不同方向的薄样条插值进行尝试与修正。

六、方案原理与设计

6.1 插值方法

6.1.1 最近邻插值

6.1.1.1 基本原理

最近邻插值是将变换后的图像中的原像素点最邻近像素的 RGB 值赋给原像素点的方法。插值函数如下所示。这种方法的方法原理较为简单，将获得比较快速的插值效率。

$$f(i+u, j+v) = f([i+u+0.5], [j+v+0.5])$$

6.1.1.2 误差分析

一维情况下，最近邻插值的误差余项为：

$$R(x) = f'(\varepsilon)(x - x_0)$$

将其扩展到二维平面上的直线 $y = y_0$ 上：

$$R(x, y_0) = \frac{\partial}{\partial x} f(x, y)|_{x=\varepsilon_x, y=y_0} (x - x_0)$$

将其扩展至整个二维平面：

$$R(x, y) = \nabla f(x, y)|_{x=\varepsilon_x, y=\varepsilon_y} (x - x_0)(y - y_0)$$

6.1.1.3 算法设计

最近邻插值的算法较为简单，在进行插值之前，应该对计算出的原像素点进行条件判断，即判断原像素点是否属于原图尺寸范围内。若计算出的原像素点坐标超出原图范围，则赋以黑色（R,G,B=0,0,0）。若计算出的原像素点在原图尺寸范围内，则通过四舍五入的方法将最邻近的整点（横纵坐标均为整数的点）的 RGB 色彩属性返回图像变形函数。

6.1.2 双线性插值

6.1.2.1 基本原理

双线性插值，又称为双线性内插。在数学上，双线性插值是有两个变量的插值函数的线性插值扩展，其核心思想是在两个方向分别进行一次线性插值。插值函数如下所示。由于双线性插值选取了对应点附近的四个点做加权平均，因此将获得比较精确的插值结果，但计算效率会受到一定影响。

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

6.1.2.2 误差分析

一维情况下，最近邻插值的误差余项为：

$$R(x) = f''(\varepsilon)(x - x_1)(x - x_2)$$

将其扩展到二维平面上的直线 $y = y_0$ 上：

$$R(x, y_0) = \frac{\partial^2}{\partial x^2} f(x, y)|_{x=\varepsilon, y=y_0} (x - x_1)(x - x_2)$$

将其扩展至整个二维平面：

$$R(x, y) = \nabla^2 f(x, y)|_{x=\varepsilon_x, y=\varepsilon_y} (x - x_1)(x - x_2)(y - y_1)(y - y_2)$$

6.1.2.3 算法设计

双线性插值需要提取对应点附近的四个点进行加权平均，在本算法中，选择的四个点为原像素点、原像素点右边的点、原像素点下方的点和原像素点右下方的点。在进行插值之前，应该对计算出的原像素点进行条件判断，即判断原像素点是否属于原图范围内。若计算出的原像素点坐标超出原图范围，则赋以黑色（R,G,B=0,0,0）。若选择的这四个点均在原图范围内，则根据基本原理中的公式对这四个点进行加权平均，计算出它的RGB值。若加权平均结果超过RGB色彩表示范围（0~255），则赋以0或255。最终将计算得到的RGB色彩属性返回图像变形函数。若选择的这四个点中有不在原图范围内的点，则返回原像素点的四舍五入整点值。

6.1.3 双三次插值

6.1.3.1 基本原理

双三次插值是一种更加复杂的插值方式，它能创造出比双线性插值更平滑的图像边缘。

$$\begin{aligned} f(i+u, j+v) &= ABC^T \\ A &= [S(u+1) \quad S(u) \quad S(u-1) \quad S(u-2)] \\ C &= [S(v+1) \quad S(v) \quad S(v-1) \quad S(v-2)] \\ B &= f(i-1:i+2, j-1:j+2) \end{aligned}$$

其中 $S(x)$ 为三次插值核函数，可由如下式子近似：

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

在进行插值之前，应该对计算出的原像素点进行条件判断，即判断原像素点是否属于原图范围内。若计算出的原像素点坐标超出原图范围，则赋以黑色（R,G,B=0,0,0）。

6.1.3.2 误差分析

一维情况下，最近邻插值的误差余项为：

$$R(x) = f^{(4)}(\varepsilon)(x - x_1)(x - x_2)(x - x_3)(x - x_4)$$

将其扩展到二维平面上的直线 $y = y_0$ 上：

$$R(x, y_0) = \frac{\partial^4}{\partial x^4} f(x, y)|_{x=\varepsilon_x, y=y_0} (x - x_1)(x - x_2)(x - x_3)(x - x_4)$$

将其扩展至整个二维平面：

$$R(x, y) = \nabla^4 f(x, y)|_{x=\varepsilon_x, y=\varepsilon_y} \prod_{i=0}^4 (x - x_i) \prod_{i=0}^4 (y - y_i)$$

6.1.3.3 算法设计

双三次插值需要提取对应点附近的 16 个点进行加权平均，在本算法中，选择的 16 个点将围绕原像素点分布在左上、右上、左下、右下方向。在进行插值之前，应该对计算出的原像素点进行条件判断，即判断原像素点是否属于原图范围内。若计算出的原像素点坐标超出原图范围，则赋以黑色（R,G,B=0,0,0）。若选择的这 16 个点均在原图范围内，则根据基本原理中的公式对这四个点进行加权平均，计算出它的 RGB 值。若加权平均结果超过 RGB 色彩表示范围（0~255），则赋以 0 或 255。最终将计算得到的 RGB 色彩属性返回图像变形函数。若选择的这 16 个点中有不在原图范围内的点，则返回原像素点的四舍五入整点值。

6.2 扭曲变形方式

6.2.1 旋转扭曲

6.2.1.1 基本原理

旋转扭曲主要是利用距离中心点距离不同，旋转角度不同来扭曲图像。具体变换公式为：

$$\begin{cases} x = r \cos(\alpha + \theta \frac{row - r}{row}) \\ y = r \sin(\alpha + \theta \frac{row - r}{row}) \end{cases}$$

其中， (r, α) 为原始坐标的极坐标表示， row 为最大旋转半径， θ 为旋转角度。

6.2.1.2 算法设计

为了实现以上变换，算法采取“遍历新图像，求取各个原始点”的方法设计。根据上文给出的旋转扭曲变换公式，可以得出对于新图像中的点 (x, y) ，其原始点 (x_0, y_0) 求取如下：

$$\begin{cases} x_0 = r \cos(\tan^{-1}\left(\frac{y}{x}\right) - \theta \frac{row - r}{row}) \\ y_0 = r \sin(\tan^{-1}\left(\frac{y}{x}\right) - \theta \frac{row - r}{row}) \end{cases}$$

由于图像是以其中央为中心点进行旋转扭曲的，因此在进行实际运算时，需要对以中心点为原点的四个象限进行分别考虑，且对于 $\tan^{-1}\left(\frac{y}{x}\right)$ 中 $x = 0$ 的情况进行单独赋值。

6.2.2 图像畸变

6.2.2.1 基本原理

图像畸变的原理是在一个二维平面上覆盖一个半球壳，将原图覆盖在半球壳上，投影在原二维平面上的图像即为畸变图像。具体变换公式为：

中心外凸：

$$\begin{cases} \rho = R \sin\left(\frac{\rho_0}{R}\right) \\ \theta = \theta_0 \end{cases}$$

中心内凹：

$$\begin{cases} \rho = R \sin^{-1}\left(\frac{\rho_0}{R}\right) \\ \theta = \theta_0 \end{cases}$$

其中， (ρ_0, θ_0) 为原始坐标的极坐标表示， R 为半球壳半径。

6.2.2.2 算法设计

为了实现畸变程度的调整，对每一个畸变程度 e ，设置相应的半球壳半径：

$$R = \frac{2d}{e}$$

其中， d 为图像对角线长度。可以看出，其原理是：半球壳半径越小，畸变程度越大。

为了实现以上变换，算法采取“遍历新图像，求取各个原始点”的方法设计。根据上文给出的图像畸变变换公式，可以得出对于新图像中的点 (x, y) ，其原始点 (x_0, y_0) 求取如下：

中心外凸：

$$\begin{cases} x_0 = r \cos\left(\tan^{-1}\left(\frac{y}{x}\right)\right) = R \sin^{-1}\left(\frac{x^2 + y^2}{R}\right) \cos\left(\tan^{-1}\left(\frac{y}{x}\right)\right) \\ y_0 = r \sin\left(\tan^{-1}\left(\frac{y}{x}\right)\right) = R \sin^{-1}\left(\frac{x^2 + y^2}{R}\right) \sin\left(\tan^{-1}\left(\frac{y}{x}\right)\right) \end{cases}$$

中心内凹:

$$\begin{cases} x_0 = r \cos(\tan^{-1}(\frac{y}{x})) = R \sin(\frac{x^2 + y^2}{R}) \cos(\tan^{-1}(\frac{y}{x})) \\ y_0 = r \sin(\tan^{-1}(\frac{y}{x})) = R \sin(\frac{x^2 + y^2}{R}) \sin(\tan^{-1}(\frac{y}{x})) \end{cases}$$

由于图像是以其中央为中心点进行旋转扭曲的, 因此在进行实际运算时, 需要对以中心点为原点的四个象限进行分别考虑, 且对于 $\tan^{-1}(\frac{y}{x})$ 中 $x = 0$ 的情况进行单独赋值。

6.2.3 TPS 网格变形

6.2.3.1 基本原理

薄板样条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, 使得能量函数

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

最小。可以证明该问题有解析解。

给定 n 个控制点 $P_1(x_1, y_1) \dots P_n(x_n, y_n)$, 记

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

$$P = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \cdots & \cdots & \cdots \\ 1 & x_n & y_n \end{bmatrix}, 3 \times n;$$

and

$$L = \left[\begin{array}{c|cc} K & P \\ \hline P^T & O \end{array} \right], (n+3) \times (n+3),$$

假设目标点为 $P'_1(x'_1, y'_1) \dots P'_n(x'_n, y'_n)$, 记

$$V = \begin{bmatrix} x'_1 & x'_2 & \cdots & x'_n \\ y'_1 & y'_2 & \cdots & y'_n \end{bmatrix}, \quad Y = \left(V \middle| \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \right)^T$$

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

其中 $\mathbf{a}_1, \mathbf{a}_x, \mathbf{a}_y, \mathbf{w}$ 为方程组 $L[\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_n, \mathbf{a}_1, \mathbf{a}_x, \mathbf{a}_y] = Y$ 的解。

其中 $U(r) = \begin{cases} r^2 \log(r^2), & r \neq 0 \\ 0, & r = 0 \end{cases}$ 为径向基函数, 实际上定义了控制点周围的变形插值函数。

基于上述模型，对于平面/图像上的任一点，都可以得到对应的目标点。

6.2.3.2 算法设计

由于以上变换的逆变换较为复杂，因此为了实现以上变换，算法采取正变换的方法，即“遍历旧图像，求取各个变换后的点”的方法设计。根据选取的各个点建立如上图所示的矩阵，计算出每一个变换后的点的坐标。

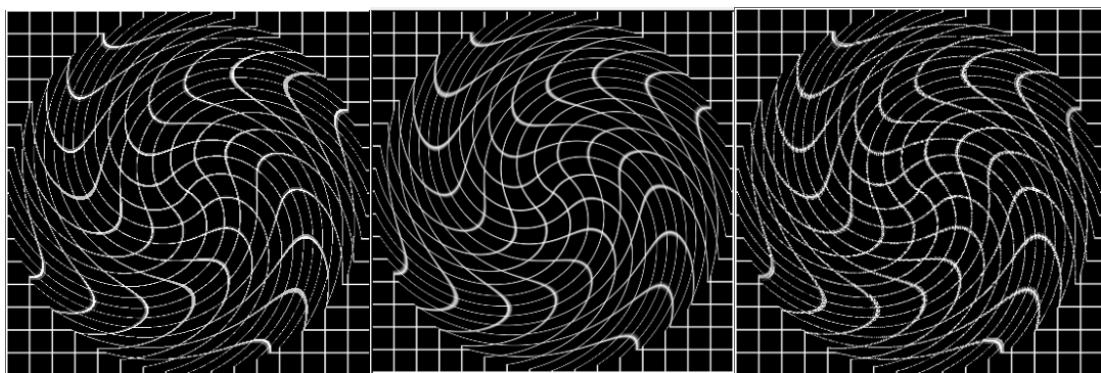
值得注意的是，由于 TPS 网格变换采取最近邻插值的插值方法，在一些变换过于剧烈的情况下，可能会产生变换后的图像遍布黑色空洞的情况。这是因为对于原图面积较小的一块区域，TPS 网格变换可能会将其变换至一块较大的区域内，从而不能实现每个点的对应。为了减缓这个问题带来的影响，在 TPS 网格变换后，对于新图像中每一个黑色的像素点，如果其附近的点不是黑色（说明不是原本的黑色点或是不在变换范围内的黑色点），则将其附近的像素点值赋给它。经过实践证明，这种方法较为便捷，且有效地改善了黑色空洞问题，图像具有较强的光滑性。

七、实验结果与分析

7.1 旋转扭曲

注：以下变换最大旋转半径为图像最大半径，旋转角度为 -150° 。

以下为黑白网格图在三种插值方式下所得到的实验结果：



以下为彩色照片在三种插值方式下所得到的实验结果：

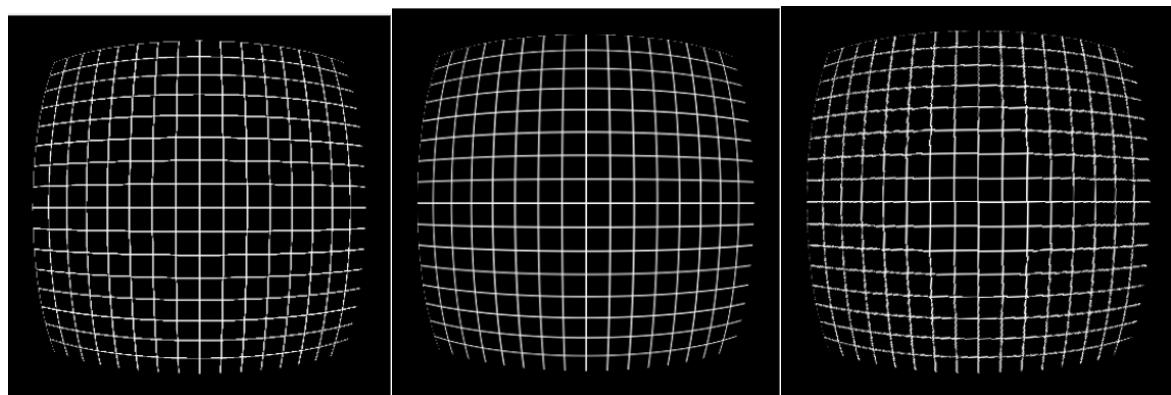


7.2 图像畸变

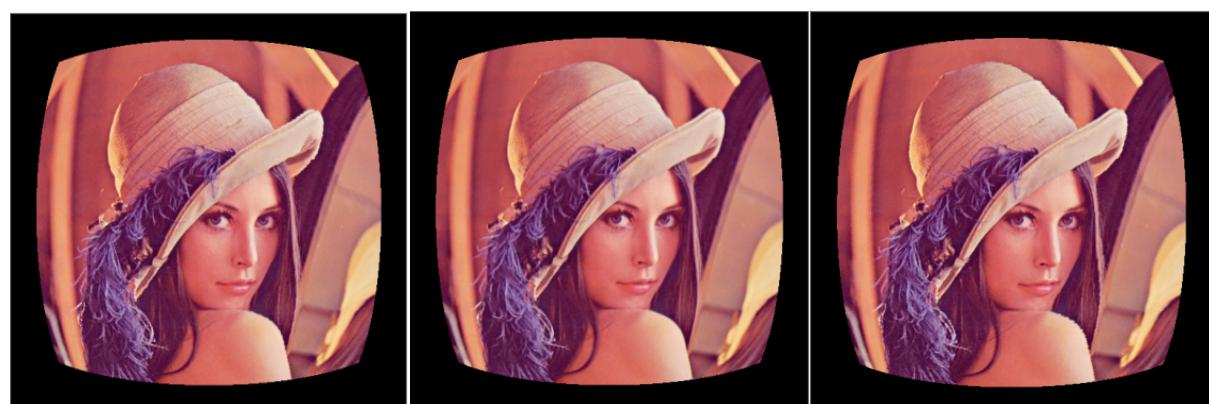
注：以下变换畸变程度均为 5。

7.2.1 中心外凸

以下为黑白网格图在三种插值方式下所得到的实验结果：

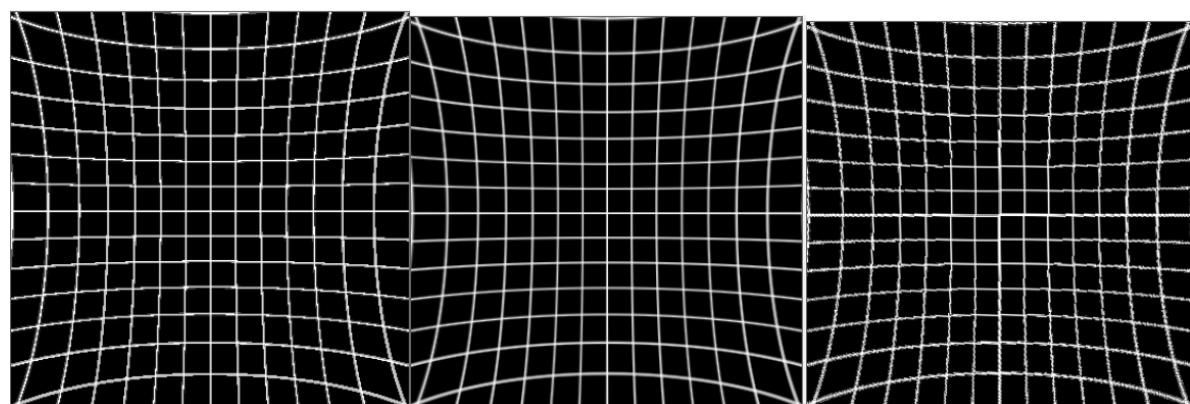


以下为彩色照片在三种插值方式下所得到的实验结果：



7.2.2 中心内凹

以下为黑白网格图在三种插值方式下所得到的实验结果：

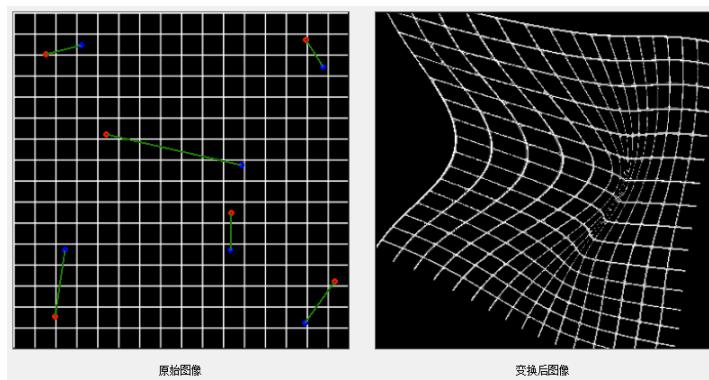


以下为彩色照片在三种插值方式下所得到的实验结果：



7.3 TPS 网格变形

以下为黑白网格图在 6 组控制点约束下所得到的实验结果：



以下为彩色照片在 6 组控制点约束下所得到的实验结果：



7.4 实验结果分析

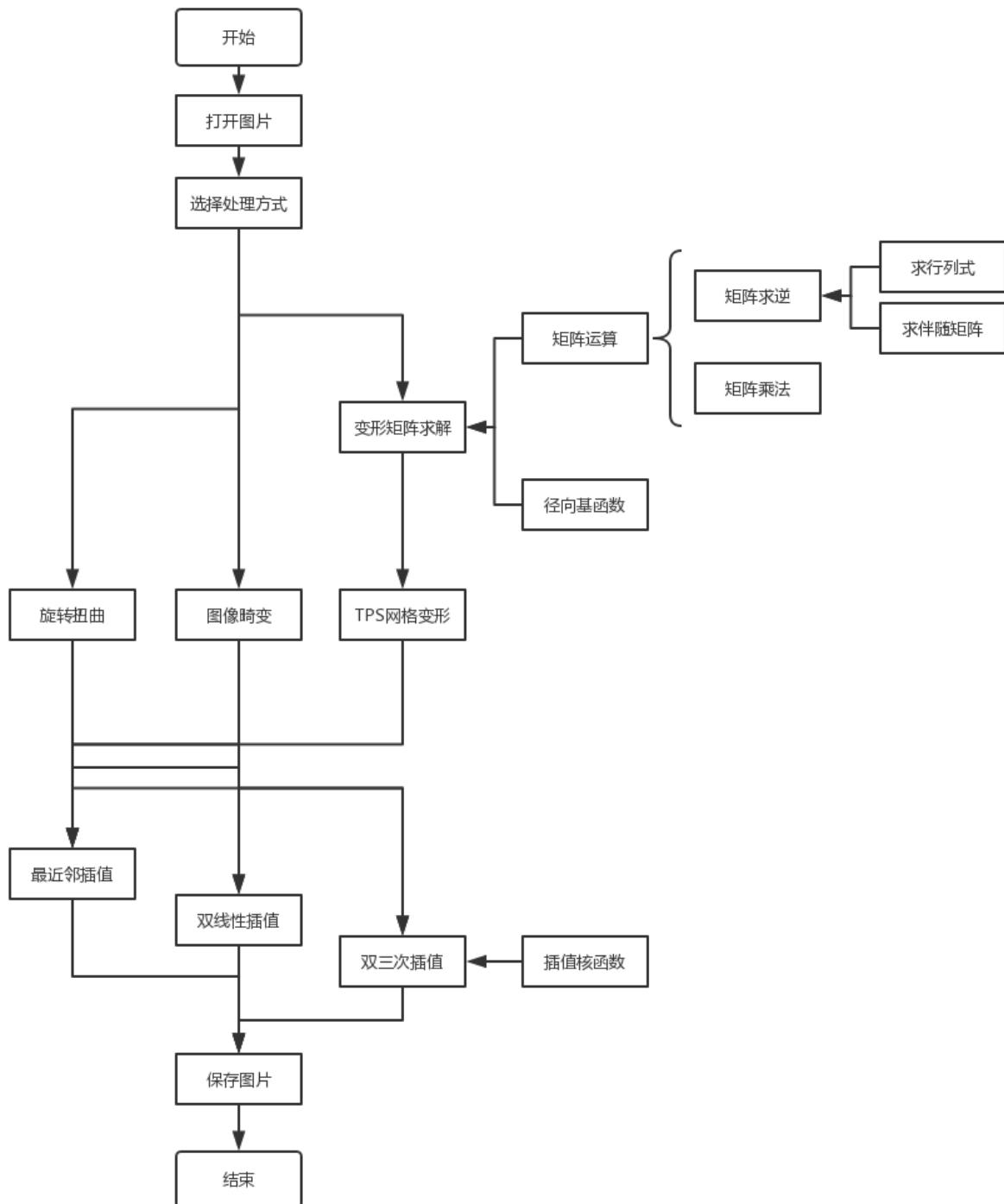
通过以上实验结果，我们可以发现对于黑白网格图和彩色照片，双线性插值的插值效果总是优于最近邻插值；而对于彩色照片，双三次插值的插值效果要优于以上两种插值方法，且图像更加精确。这可以从前文所述的插值方法误差分析中可以得到这种现象的原因。

然而，令我比较意外的是，对于黑白网格图，双三次插值的插值效果并不是特别理想，并没有显现出与运行时间对等的优越性。经过分析，本人提出以下猜测：由于双三次插值需要对 16 个像素点进行加权平均，而在黑白网格图中，白色部分十分细，且在黑白交界处不具备彩色照片的光滑性，因此在这 16 个像素点内很有可能既有黑色又有白色，在加权平均

后会呈现出各种深度的灰色，造成图像的显示比较粗糙。

在运行过程中，我们也能发现，最近邻插值是运行速度最快的插值方法，其次是双线性插值，而双三次插值计算时间基本上需要好几秒。因此，实际运行程序时我们也应当根据时间与效果需求合理选择插值方式。

八、程序框图



九、项目总结与思考

这次的项目和我之前两年所做的项目大有不同，这次项目不是多人合作项目，且要求的方法比较庞杂，对界面的交互性也有一定要求，需要徒手独立完整写上 800 行左右代码并对界面进行一定的设计与规划。

在算法编写过程，尤其是 TPS 算法编写过程中，由于算法比较复杂，在对数组的处理中时常遇到一些小问题。在多次查阅参考文献后，我对 TPS 变形算法有了更深的理解，并最终获得了较为理想的结果。

本次大作业至此已经完全完成，在完成过程中，通过对于算法的思考和优化令我收获颇丰，也大大加深了我对搜索相关知识的了解。