实验二 Image Warping

ID: 58 陈文博

February 26, 2020

1 实验要求

* 实现 Image Warping 的两种算法
  - Inverse distance-weighted interpolation method (IDW)
  - Radial basis functions interpolation method (RBF)
* 对实验产生的白色裂缝进行填补

2 开发环境

IDE： Microsoft Visual Studio 2019 community

CMake： 3.16.3

Qt： 5.14.1

Eigen： 3.3.7

ANN： 1.1.2

3 算法原理

3.1 基本原理

- 输入：n 对控制点对 $(\boldsymbol{p_i}, \boldsymbol{q_i}), i = 1, 2, \cdots, n$，其中 $p_i \in \mathbb{R}^2$ 为控制源点，$q_i \in \mathbb{R}^2$ 为控制目标点
- 目标：找到一个映射 $f : \mathbb{R}^2 \to \mathbb{R}^2$，满足 $f(\boldsymbol{p_i}) = \boldsymbol{q_i}, i = 1, 2, \cdots, n$

3.2 Inverse distance-weighted interpolation methods(IDW)[1]

IDW 算法基本原理是根据给定的控制点对和控制点对的位移矢量，计算控制点对周围像素的反距离加权权重影响，实现图像每一个像素点的位移。

选择 n 组控制点对 $(\boldsymbol{p_i}, \boldsymbol{q_i}), i = 1, 2, \cdots, n$，目标映射 $f : \mathbb{R}^2 \to \mathbb{R}^2$ 可表示为以下形式：

$$f(\boldsymbol{p}) = \sum_{i=1}^{n} \omega_i(\boldsymbol{p}) f_i(\boldsymbol{p}) \tag{3.1}$$

其中，权重 $w_i(\boldsymbol{p})$ 满足：

$$w_i(\boldsymbol{p}) = \frac{\sigma_i(\boldsymbol{p})}{\sum\limits_{j=1}^{n} \sigma_j(\boldsymbol{p})} \tag{3.2}$$

$\sigma_i(\boldsymbol{p})$ 反映第 i 对控制点对像素 $\boldsymbol{p}$ 的反距离加权权重影响程度，可以直接取

$$\sigma_i(\boldsymbol{p}) = \frac{1}{\|\boldsymbol{p} - \boldsymbol{p_i}\|^{\mu}} (\mu > 1) \tag{3.3}$$

也可以取 locally bounded weight：

$$\sigma_i(\boldsymbol{p}) = [\frac{(R_i - d(\boldsymbol{p}, \boldsymbol{p_i}))}{R_i d(\boldsymbol{p}, \boldsymbol{p_i})}]^{\mu} \tag{3.4}$$

f_i 为线性函数，满足：

$$f_i(\boldsymbol{p}) = \boldsymbol{q_i} + \boldsymbol{T_i}(\boldsymbol{p} - \boldsymbol{p_i}) \tag{3.5}$$

$\boldsymbol{T_i}$ 为二阶矩阵

$$\boldsymbol{T_i} = \begin{bmatrix} t_{11}^{(i)} & t_{12}^{(i)} \\ t_{21}^{(i)} & t_{22}^{(i)} \end{bmatrix} \tag{3.6}$$

矩阵 $\boldsymbol{T}$ 的确定，可通过求解最优化问题：

$$\min_{\boldsymbol{T_i}} E_i(\boldsymbol{T_i}) = \sum_{j=1, j\neq i}^{n} \sigma_i(\boldsymbol{p_j}) \| \boldsymbol{q_j} - f_i(\boldsymbol{p_j}) \|^2 \tag{3.7}$$

对上式矩阵 $\boldsymbol{T}$ 的各个元素分别求导，令方程为 0 可得

$$\boldsymbol{T} \sum_{j=1, j\neq i}^{n} \sigma_i(\boldsymbol{p_j}) \boldsymbol{\Delta p} \boldsymbol{\Delta p}^T = \sum_{j=1, j\neq i}^{n} \sigma_i(\boldsymbol{p_j}) \boldsymbol{\Delta q} \boldsymbol{\Delta p}^T \tag{3.8}$$

其中 $\boldsymbol{\Delta p} = (\boldsymbol{p_j} - \boldsymbol{p_i}\ \boldsymbol{0})$，$\boldsymbol{\Delta q} = (\boldsymbol{q_j} - \boldsymbol{q_i}\ \boldsymbol{0})$，当 $\sum_{j=1, j\neq i}^{n} \sigma_i(\boldsymbol{p_j}) \boldsymbol{\Delta p} \boldsymbol{\Delta p}^T$ 非奇异时，可得

$$\boldsymbol{T} = \left(\sum_{j=1, j\neq i}^{n} \sigma_i(\boldsymbol{p_j}) \boldsymbol{\Delta q} \boldsymbol{\Delta p}^T \right) \left(\sum_{j=1, j\neq i}^{n} \sigma_i(\boldsymbol{p_j}) \boldsymbol{\Delta p} \boldsymbol{\Delta p}^T \right)^{-1} \tag{3.9}$$

$\boldsymbol{T_i} (i = 1, 2, \cdots, n)$，确定以后映射 f 也就相应确定

3.3 Radial basis functions interpolation method(RBF) [2]

选择 n 组控制点对 $(\boldsymbol{p_i}, \boldsymbol{q_i}), i = 1, 2, \cdots, n$，目标映射 $f : \mathbb{R}^2 \to \mathbb{R}^2$ 可表示为以下形式：

$$f(\boldsymbol{p}) = \sum_{i=1}^{n} \alpha_i g_i(\| \boldsymbol{p} - \boldsymbol{p_i} \|) + \boldsymbol{A}\boldsymbol{p} + \boldsymbol{B} \tag{3.10}$$

其中，g_i 是径向基函数，通常可取 Hardy multiquadrics $g(t) = (t^2 + c^2)^{\pm \frac{1}{2}}$ 或高斯函数 $g_\sigma = e^{-t^2/\sigma^2}$，为了计算方便，这里取 Hardy multiquadrics：

$$\begin{aligned} g_i(d) &= (d + r_i)^{\pm \frac{1}{2}} \\ r_i &= \min_{j\neq i} d(\boldsymbol{p_i}, \boldsymbol{p_j}) \end{aligned} \tag{3.11}$$

对于线性部分分量 $\boldsymbol{A}\boldsymbol{p} + \boldsymbol{B}$，本例简单的取 $\boldsymbol{A} = \boldsymbol{I}$ 和 $\boldsymbol{B} = \boldsymbol{0}$

4 程序架构

4.1 文件结构

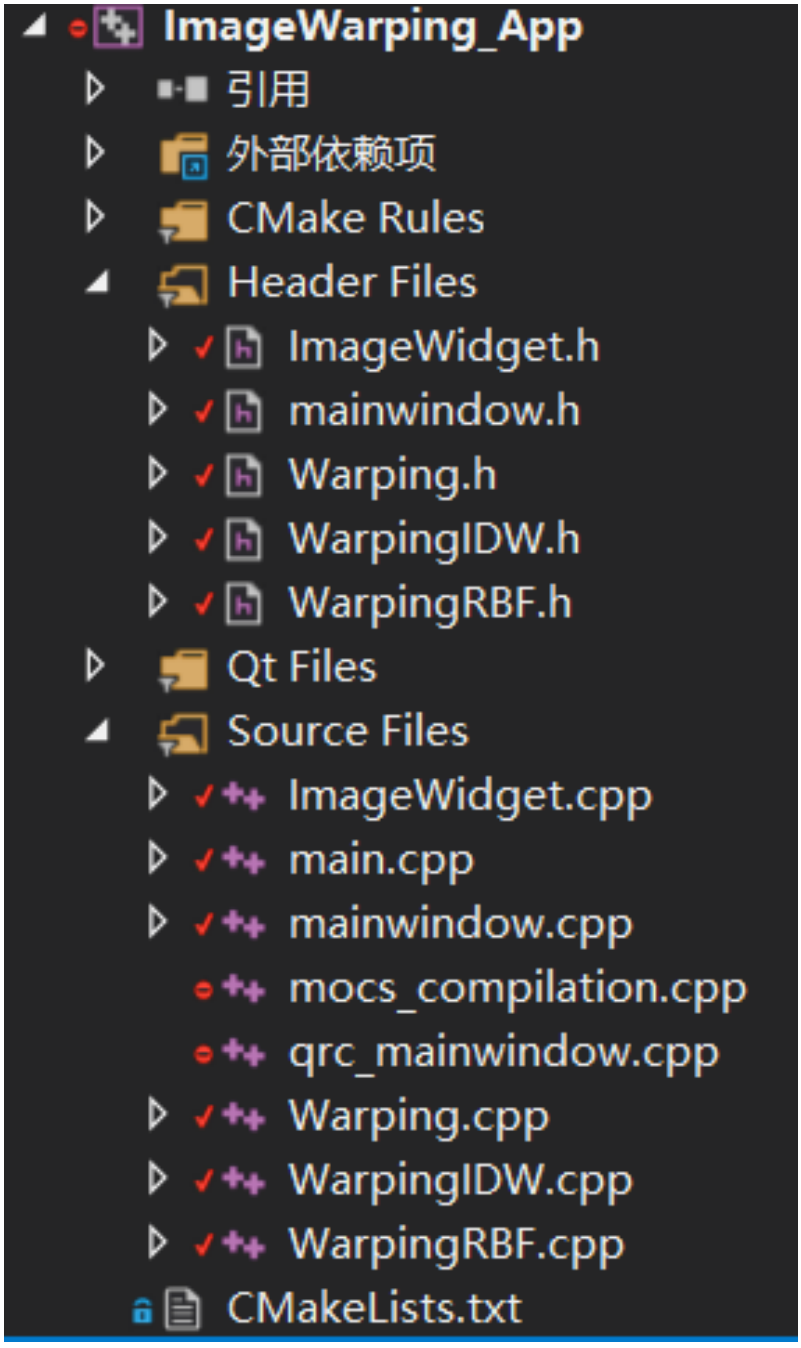

Figure 4.1: 文件结构

4.2 面向对象设计

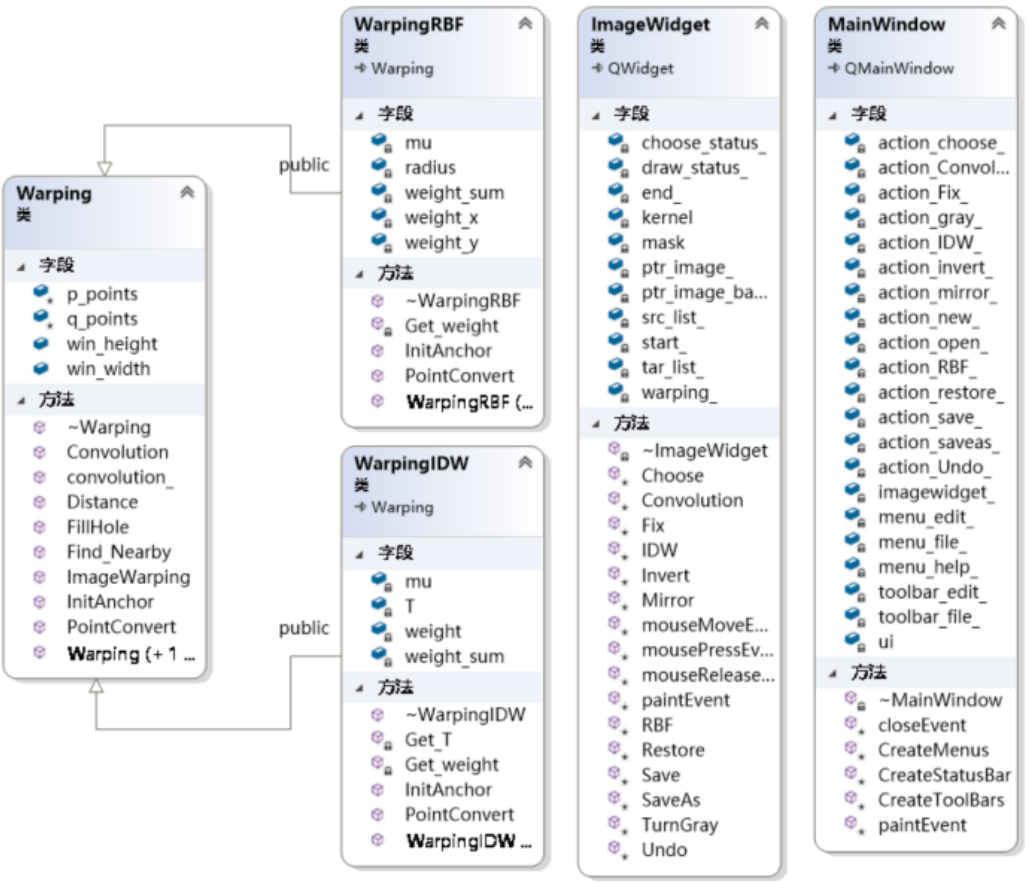

Figure 4.2: 类图

在样例的基础上新增 Warping 父类，声明了初始化控制点、计算距离、像素点坐标变换、白缝填补的方法，子类 WarpingRBF 和 WarpingIDW 继承 Warping，分别定义了各自的 Image Warping 算法。

5 设计难点与解决

5.1 图像位置的影响

样例中的图像是默认处于窗口正中心，这会导致读取的控制点坐标系和图像内像素点坐标系不一致。需要在控制点初始化过程中，传入窗口大小信息对控制点对序列做平移变换预处理

5.2 白缝的消除

由于算法映射不完全，处理后图像会出现许些白缝，这里采用 ANN 库提供的邻域搜索，利用空像素周边的非空像素取均值进行插值。在配置 ANN 的过程中遇到不少困难，该库官方只提供 Win32 的链接库，而本例采用 x64，无法正常链接。于是我利用 Homework0 中编译动态链接库的方法重新编译 ANN 源码，得到 x64 版本的链接库，经测试可正常使用。（注意：需要将 ANN.dll 置于/bin 文件夹中程序才能正常运行）

5.3 内存管理

为了避免内存泄漏，用 Qt 附带的模板类 QVector 代替 STL Vector，不需要显式释放内存，同时图像指针 ptr_image_ 和 ptr_image_backup_ 需要在析构函数中进行释放，否则会发生内存泄漏。

6 实验效果

6.1 标准图像测试

如下图所示，固定四角，蓝色为控制起始点，绿色为控制终止点

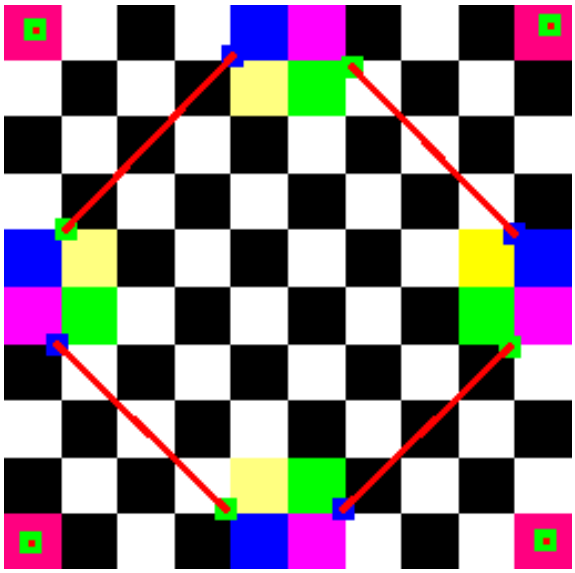

Figure 6.1: 拉伸情况

6.2 IDW 算法

6.2.1 $\mu = -1$ 情况

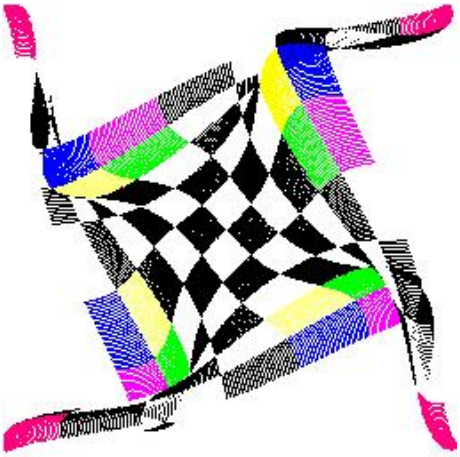

Figure 6.2: 修复前

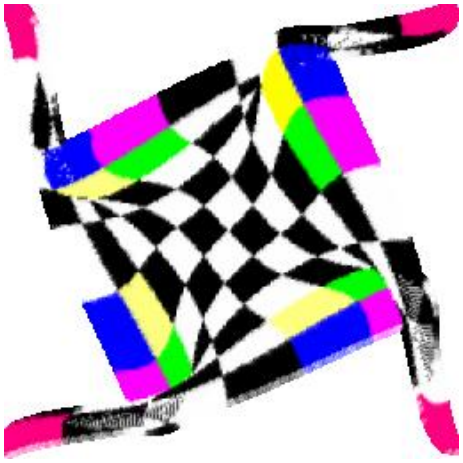

Figure 6.3: 修复后

6.2.2 $\mu = 1$ 情况

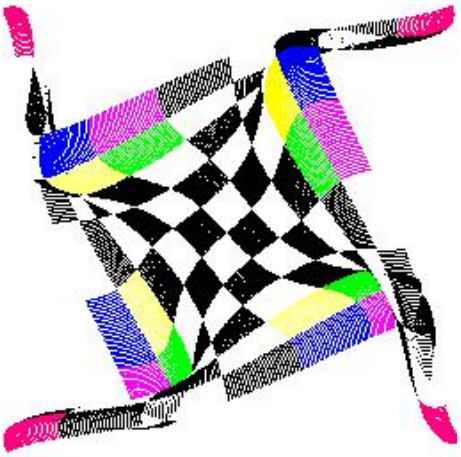

Figure 6.4: 修复前

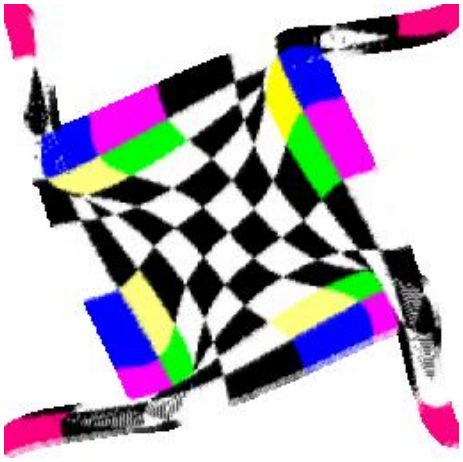

Figure 6.5: 修复后

6.3 RBF 算法

6.3.1 $\mu = 0.5$ 情况

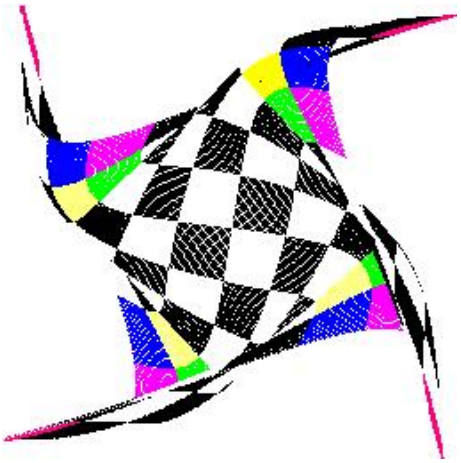

Figure 6.6: 修复前

Figure 6.7: 修复后

6.3.2 $\mu = -0.5$ 情况

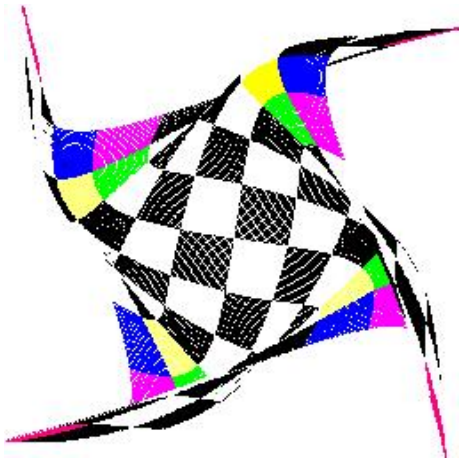

Figure 6.8: 修复前

Figure 6.9: 修复后

6.4 其他测试

6.4.1 柴犬表情包

原图片：

Figure 6.10: 原始图片

处理后：

Figure 6.11: Happy

Figure 6.12: Emmm...

6.4.2 平头哥也忍不住笑了

原图片：

Figure 6.13: 高冷

处理后：

Figure 6.14: 微笑

Figure 6.15: 扭曲的微笑

7 总结

本例中使用 IDW 和 RBF 两种方法进行图像的拉伸变换，理论上 IDW 和 RBF 的运算复杂度均为 $O(n^2+nN)$，而由于实际运算中，IDW 计算一个像素点的浮点乘法加法次数比 RBF 方法更多，在实验中也可以发现 RBF 处理速度要比 IDW 快 3 到 4 倍。

REFERENCES

[1] Detlef Ruprecht and Heinrich Muller. Image warping with scattered data interpolation. *IEEE Computer Graphics and Applications*, 15(2):37–43, 1995.

[2] Nur Arad and Daniel Reisfeld. Image warping using few anchor points and radial functions. In *Computer graphics forum*, volume 14, pages 35–46. Wiley Online Library, 1995.