USTC_CG HW3 PoissonImageEditing

张继耀,PB20000204

2023 年 6 月 24 日

目录

1 问题介绍

1.1 主要目的

- 实现 Poisson Image Editing 算法
- 实现多边形光栅化的扫描线转换算法
- 学习使用Eigen库、OpenCV
- 通过矩阵预分解实现实时拖动区域显示结果

2 算法设计

2.1 程序框架

下面我们首先给出整个项目的类视图

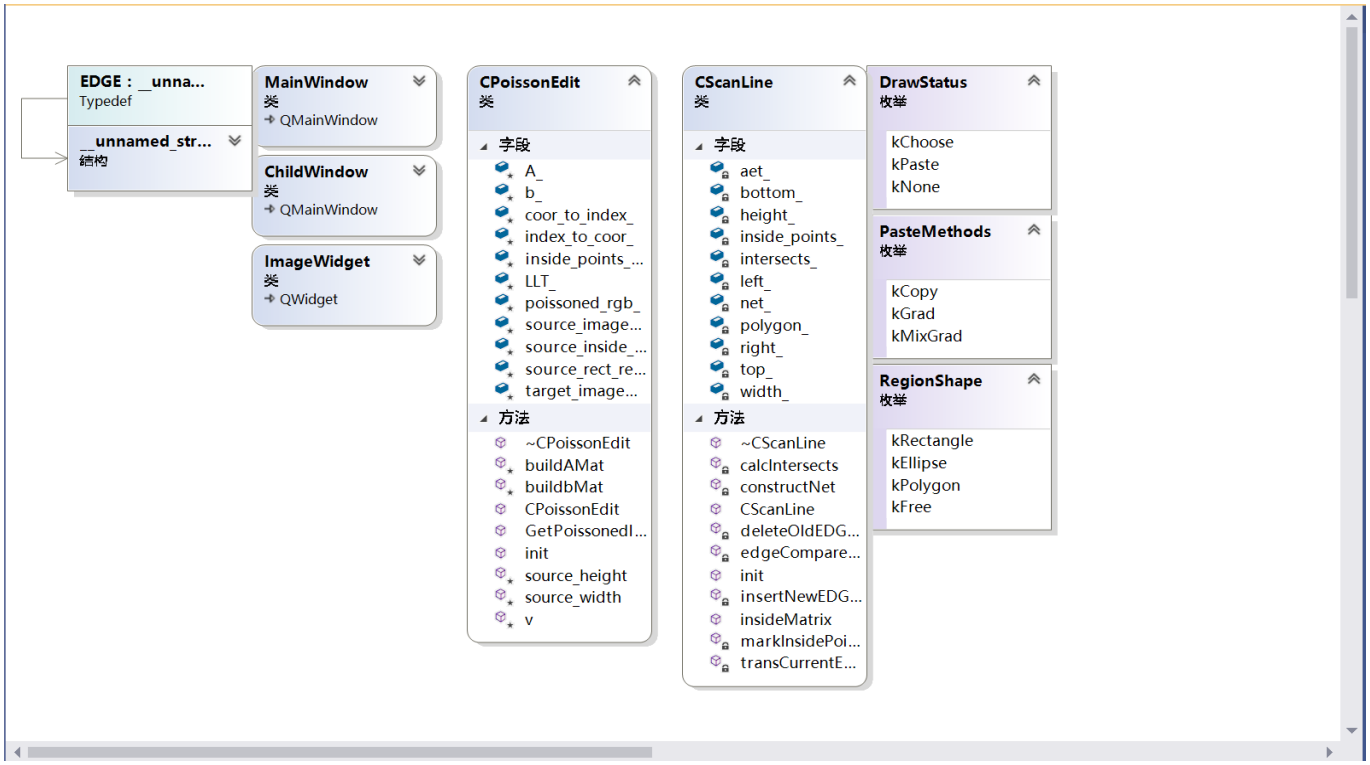

图 1: 项目的类视图

2.2 Poisson Image Editing 算法

该算法最初由Patrick Pérez、Michel Gangnet和Andrew Blake于2003年提出，可以在保持图像连续性的同时，将一张图像的内容嵌入到另一张图像中。

该算法将原图像目标区域的梯度与被粘贴图像中的实际梯度相匹配，求解以下方程来实现图像融合：

$$\begin{cases} min_f \iint_\Omega |\nabla f|^2, f|_{\partial\Omega} = f^*|_{\partial\Omega} \\ \Delta f = 0 \ \ over \ \ \Omega, f|_{\partial\Omega} = f^*|_{\partial\Omega} \end{cases}$$

在本问题中，求解图像的像素点事实上是一个离散的问题，转化成以下方程:

$$\forall p \in \Omega, |N_p| f_p - \sum_{q \in N_p \cap \Omega} f_q = \sum_{q \in N_p \cap \partial\Omega} f_q^* + \sum_{q \in N_p} v_{pq}$$

其中 $v = \nabla g$为向量场，g为图像的像素点。N_p表示与像素点p相邻的四个像素集合。注意以上方程在内部时可以有更简单的形式。

$$|N_p|\, f_p - \sum_{q \in N_p} f_q = \sum_{q \in N_p} v_{pq}$$

论文 [1]中给出了两种选择梯度场的方法，一种是完全使用原图像的内部梯度

$$\forall \langle p, q \rangle, v_{pq} = g_p - g_q$$

另一种是用混合梯度:

$$\forall x \in \Omega, v(x) = \begin{cases} \nabla f^*(x), |\nabla f^*(x)| > |\nabla g^*(x)| \\ \nabla g^*(x), \ otherwise \end{cases}$$

对图像的每个像素点列出以上的方程，就得到了一个线性方程组。系数矩阵A是一个稀疏矩阵，每行至多有5个非零元素。于是这个问题实质上又变成了解线性方程组的问题。

2.3 矩阵预分解

如果我们希望让用户在拖动时实时得到处理后的结果，就要采用矩阵预分解的方法。因为对于线性方程组$AX = b$，每次的矩阵A都是相同的(原图信息)。每次改变的事实上只是边界的信息，对矩阵A采取预分解可大大提高效率。

在用户点击菜单时后台进行预分解，即可节省时间来实现实时拖动。这样能让用户体验更好。

对于本问题，构造的矩阵A事实上是对称正定的。对称较易看出，因为两个点若为邻居一定是互为邻居的。正定则是因为矩阵的对角元元素都是大于0的。

因此我们可以采用LLT方法进行预分解，这个方法的效率是最高的。

2.4 多边形扫描线算法

以下是扫描线算法的具体步骤:

- 首先得到多边形对应的y坐标的最小值与最大值
- 然后从多边形的最低顶点开始，依次向上递增扫描线的位置

  若共享顶点的两条边分别落在扫描线两边，交点只算一个

  若共享顶点的两条边落在扫描线同一边，交点算两个或零个
- 求出扫描线与多边形的交点并排序
- 将得到的顶点配对，每对顶点之间的点为内点
- 重复以上步骤，得到多边形内部所有点

3 结果展示

3.1 程序界面

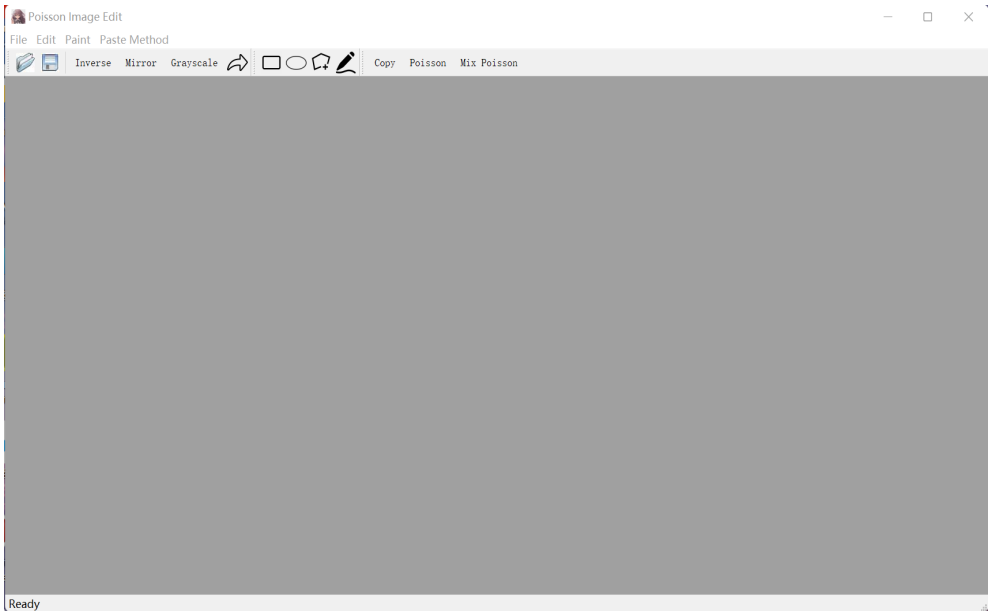

图 2: 程序界面

3.2 功能讲解

- 最左边的图标是打开、保存图片， Redo可以恢复图片原始的样子,Inverse、Mirror、Grayscale可以对图像直接处理
- 中间的矩形、椭圆、多边形是选择划线区域的方式，用户可自由选择所划的区域。选择多边形方式时，双击鼠标左键代表结束选取。
- 最右边是选择复制的方式，依次为:直接像素粘贴、内部梯度粘贴、混合梯度粘贴
- 交互时，先选中要复制的图像，划出要复制的区域，再选中目标图像，选择复制方式，然后直接拖动鼠标即可实时得到结果。

3.3 实验结果

3.3.1 选择不同区域

选择矩形区域时

图 3: 所选区域

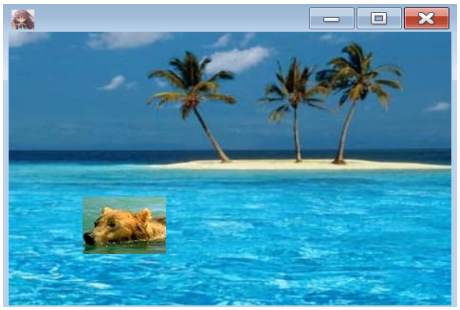

图 4: 结果

选择多边形区域时

图 5: 所选区域

图 6: 结果

自由绘制区域时

图 7: 所选区域

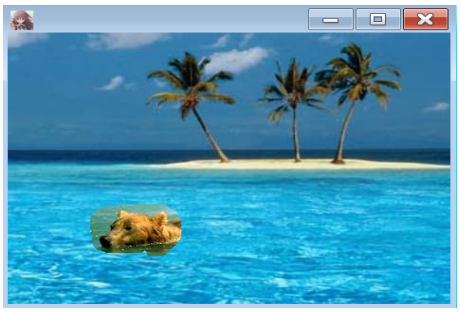

图 8: 结果

3.3.2 内部梯度与混合梯度的对比

下面是使用两种梯度方法得到的结果。可以明显看出，混合梯度是更优一些的

图 9: 所选区域

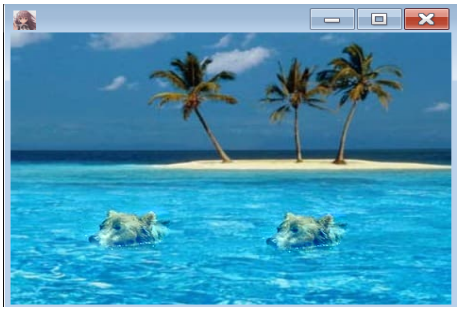

图 10: 结果:左边为内部梯度，右边为混合梯度

3.3.3 最终结果

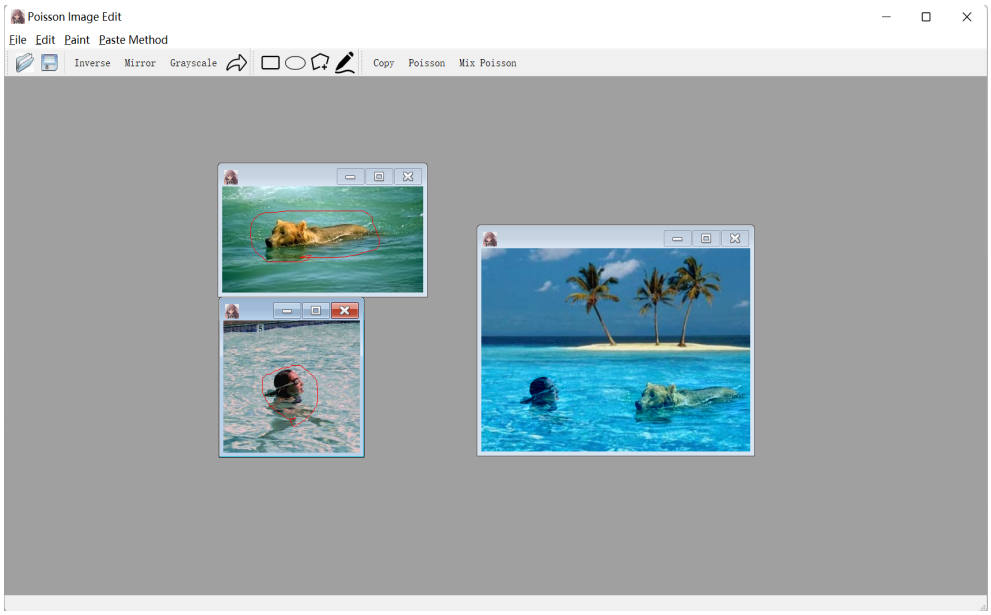

图 11: 最终得到的结果

4 总结与讨论

4.1 问题与不足

- 对于较大的图片，拖动时所需时间较长，效果不好
- 存在内存泄漏问题，不过貌似是因为opencv的mat库引起的
- 程序的交互性和鲁棒性还可以做的更好些，时间问题来不及优化了

4.2 反思

- 配置环境时一定要注意对应的版本!! 因为本课程是2020年的，我自己一开始用的VS2022+opencv4.7.0在配置时踩了很多坑
- 构建好项目可能会出现缺少opencv_world470.dll的问题，可以利用网络资源来解决，如CSDN、ChatGPT等
- 特别注意OpenCV mat的行列问题!!

参考文献

[1] Perez, Patrick. "Poisson Image Editing." ACM Transactions on Graphics (TOG), vol. 22, no. 3, 2003, pp. 313-318.