计算机图形学

第三次实验报告

Passion image editing

PB20000264

韩昊羽

一、实验要求

- 实现基础的 Poission Image Editing 算法，即使用矩形选择区域，满足基础的 possion 方程
- 实现多边形的扫描线填充算法
- 实现任意图型的扫描线填充
- 学会使用 Eigen 进行大型稀疏方程组的求解，学会 QR，LDLT,LLT 等不同分解方法
- 选做：实现实时拖动

二、 操作环境

2.1 QT 图形化编程

IDE：Microsoft Visual Studio 2019 community

QT: 5.12.12

2.2 Eigen 库

Eigen： 3.4.0

三、 功能介绍

1.打开图片（file dialog 实现）

2.绘制自由区域（与 minidraw 相同）

3.通过拖拽实现对区域的移动，同时实时更新图像

4.撤销操作

四、架构设计

4.1 文件结构

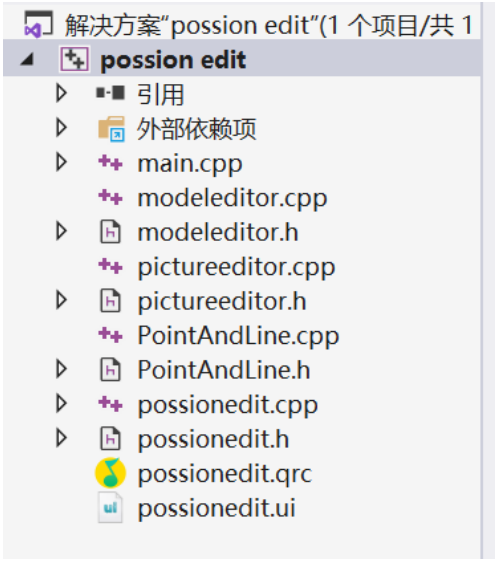
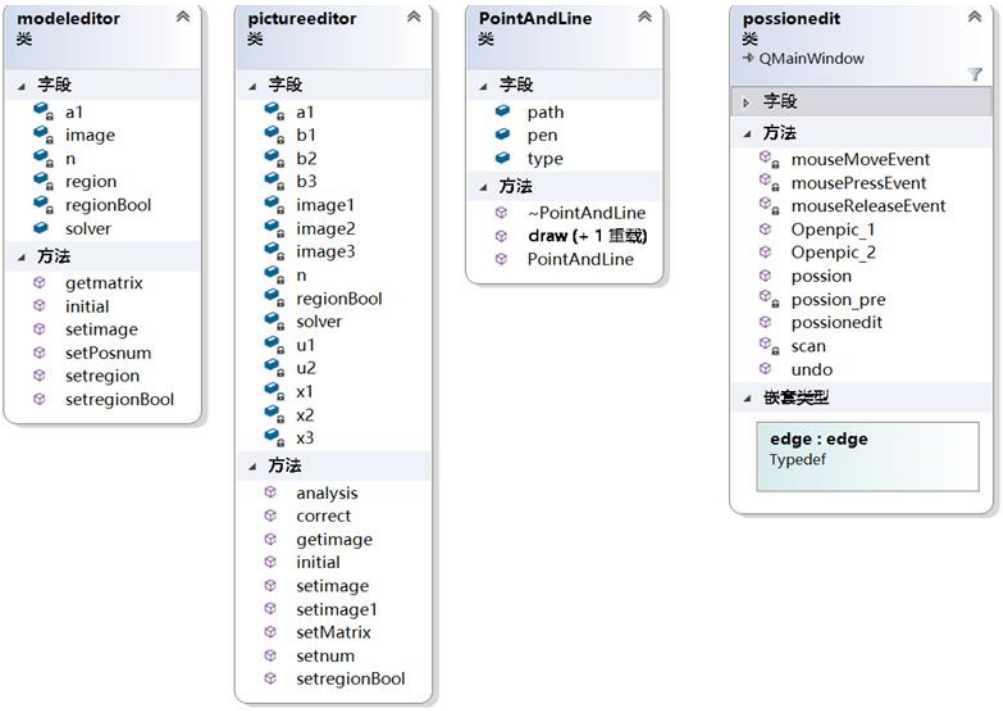


Figure.1 文件结构

对窗体以及鼠标事件的操作和扫描线算法(因为不长而且和窗体联系比较紧密,没有单独分出一个类)保存在主类 possionedit 里面，绘制方面的操作存储在了 PointAndLine 类中，对初始区域的初始化以及矩阵预分解（主要涉及到对矩阵的操作，即 AX=B 中的 A）放在了 modeleditor 类中，对目标区域的采样以及矩阵的求解（涉及到对向量的操作，即 B）放在了 pistureeditor 类中。

4.2 类图


modeleditor
类
字段
a1
image
n
region
regionBool
solver
方法
getmatrix
initial
setimage
setPosnum
setregion
setregionBool
pictureeditor
类
字段
a1
b1
b2
b3
image1
image2
image3
n
regionBool
solver
u1
u2
x1
x2
x3
方法
analysis
correct
getimage
initial
setimage
setimage1
setMatrix
setnum
setregionBool
PointAndLine
类
字段
path
pen
type
方法
~PointAndLine
draw (+ 1 重载)
PointAndLine
possionedit
类
QMainWindow
字段
方法
mouseMoveEvent
mousePressEvent
mouseReleaseEvent
Openpic_1
Openpic_2
possion
possion_pre
possionedit
scan
undo
嵌套类型
edge : edge
Typedef

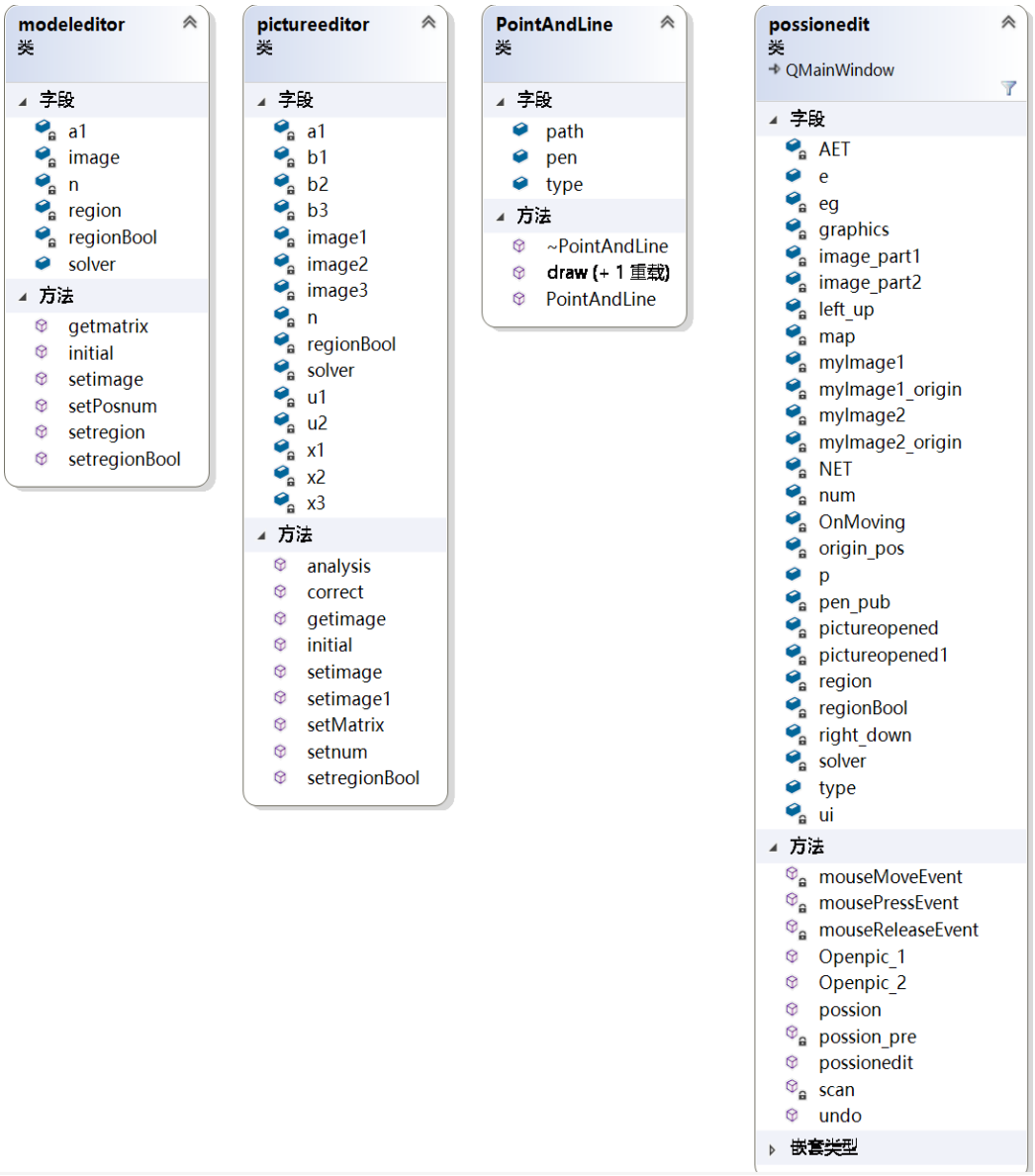


Figure.2 类图

PointAndLine 类最简单，path 存的是图形边界的点链，pen 是绘制的笔刷，type 是图形类型。

Modeleditor 类中 a1 是矩形，region 表示区域，是将处于区域中（边界和内部）的点进行标号，存储每个点的标号。RegionBool 则是针对边界和内部的区分，将内部的点存为 1，边界的点存为 2，其余是 0，方便对每个点属性的读取。Solver 是 LDLT 分解编辑器。除了设置和读取相关方法外，initial 是在传入相关区域后对矩阵赋值，

压缩，分解相关操作。

Pictureeditor 类中 a1 同为矩形，b1，b2，b3 分别是三个通道方程右边的向量（即 AX=B 的 B），image1 是原始图像，image2 是目标图像，image3 是更改之后的图像。X1，x2，x3 是解。同样除了获取和赋值相关函数外，initial 负责传入 A 并初始化，analysis 实时对 image2 进行更新。

Passioneditor 类中是对窗体和流程的主要操作，e ，p 是两个 Modeleditor 和 Pictureeditor 的实例，AET,NET 存的是活动边表和新边表，同时还自定义 struct edge 存储在边表里，这些涉及到扫描线算法。myimage 主要是打开的两张图片，pictureopened 1 / 2 是标记图片是否被打开，left_up 和 right_down 是自由绘制图形边框（矩形)的左上角和右下角。Onmoving 标记是否是拖拽中,origin_pos 存储刚开始拖拽的时候鼠标的位置，是为了保持鼠标和图像的相对距离不变。Graphics 存的是自由绘制的图形。对于方法，和 mouse 有关的三个方法涉及到鼠标事件，openpic 是打开两边的图片，undo 是撤销（回到初始），scan 是扫描线算法，possion 和 possion_pre 分别是调用 Modeleditor 预分解和调用 Pictureeditor 实时计算。

五、功能实现

打开图像功能和绘制功能和之前类似，不再叙述。

5.1 Possion 融合基础算法

我们希望将一块区域融入到一个图像中，当然首先希望二者在边界上是相同的，不仅如此，我们还希望能保留内部的梯度信息，即让新的图像和原来的梯度场相同。在这种思想下，对方程加以离散化就转化为了解线性方程组的的问题。

对于符号，我们约定Ω是区域的内部，$\partial\Omega$是区域的边界（都是二维区域），f待求解的目标函数（表示目标图像），f^*是要被融入的图像（保持边界相同），g是原始图像（保存梯度场）。

最基础的，我们希望f平滑，即求解最小二乘问题

$$\min_f \iint_\Omega |\nabla f|^2 \quad \text{满足} \quad f = f^*, \ on\partial\Omega$$

更进一步，我们让f和g的梯度场尽可能一样，即

$$\min_f \iint_\Omega |\nabla f - v|^2 \quad \text{满足} \quad f = f^*, \ on\partial\Omega, \ v\text{是}g\text{的梯度场}$$

这个最小二乘问题可以转化成一个偏微分方程的解，即

$$\Delta f = \nabla v = \Delta g \ \text{满足} \quad f = f^*, \ on\partial\Omega$$

但考虑到图像是散点，我们需要求解离散的微分方程，我们约定N_p是和 p 点相接的点（即上下左右四个点）。原方程转化为

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

$$f = f^*, \ on\partial\Omega$$

其中v_{pq}是梯度场v的投影，在离散意义下是$g_p - g_q$

求解这个线性方程组，f_p即是每一个像素的值。

5.2 Possion 融合混合算法

基础算法只保留了g的梯度场的信息，有些时候，我们希望图像尽量的自然，这样就需要考虑f^*的梯度场信息，我们将v_{pq}进行更新：

$$v_{pq} = \begin{cases} f_p^* - f_q^* \ , \ if \ |f_p^* - f_q^*| > |g_p - g_q| \\ |g_p - g_q|, otherwise \end{cases}$$

这样就可以同时保存f^*和g的梯度值。

5.3 扫描线算法

扫描线算法希望在通过对一条条平行线的遍历下，找到每一条扫描线和哪些边相交，并依据 x 进行排序，最终决定那些区间位于区域内部。

对扫描线和边的交点，以边表的形式进行操作，AET 即活动边表表示当前扫描线和哪些边相交，是一维链表，AET 中的数据类型 edge 主要存储三个值：当前扫描线和边的相交 x 值，每对 y 加 1 时 x 变化的值，边中 y 的最大值。同时使用 NET 存储在 y = i 时开始相交的边，是二维链表，这样每一次扫描时先将第一次和扫描线相交的边加入 AET（即 NET 的第 i 个），再将已经达到最大 y 的那些边去除，遍历获取 AET 中边的所有相交 x 值，并对 x 进行更新，最后把所有相交 x 值排序，一一配对，处于配对的两个值中间的值就是区域内部。但也要注意顶点：两个边在顶点一侧（以平行于 x 轴的线作为对称轴）时，顶点是假顶点，异侧是是真顶点，同时还要考虑和扫描线平行的边等等。

六、 难点难题

6.1 矩阵分解问题

开始写程序时，我一度很迷惑为什么文章中的方程要用$|N_p|$来代替 4，思考能不能直接定义内部是上下左右都有像素的点，然后直接带入 4，但在矩阵分解的时候遇到了问题。我想采用 LLT 分解，但这样解出来的矩阵总是全零，后来我才发现如果直接将 4 带入的矩阵并不是正定对称阵，所以又重新按照文章的方程重写矩阵。

6.2 稀疏矩阵赋值问题

需要将矩阵初始化！一开始矩阵的值并不是默认为 0。一开始我打算采用 insert 方法对稀疏矩阵进行赋值，后来发现 insert 方法每次在赋值时都要重新分配内存，速度很慢，后来我采用了先对矩阵 reserve 一下，标定每一行最大的非零元个数，省去了重新分配内存的步骤。

6.3 预分解问题

对于矩阵预分解，我本来设想是将预分解后的矩阵和求解器 solver 直接传入负责求解的 Pictureeditor 类中，但发现不允许对 solver（SimplicialLDLT 类）直接赋值，只能加入初始化函数然后传入分解前的矩阵，在初始化函数内进行预分解。

6.4 扫描线算法问题

一开始我并没有想用活动边表，希望在每绘制一条边时就将其对应的像素点都标记出来然后直接对每一行像素点进行扫描，这样就不用计算交点了，但后来发现如果斜率太小会导致在同一行和一条边相交两次，只能用活动边表更新交点来解决。

七、实验结果

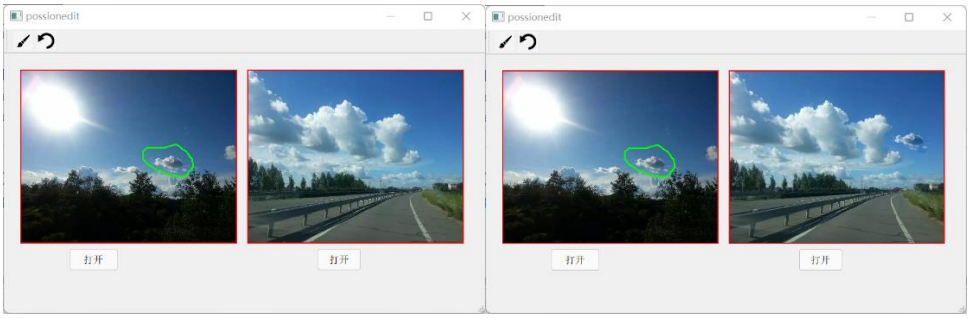
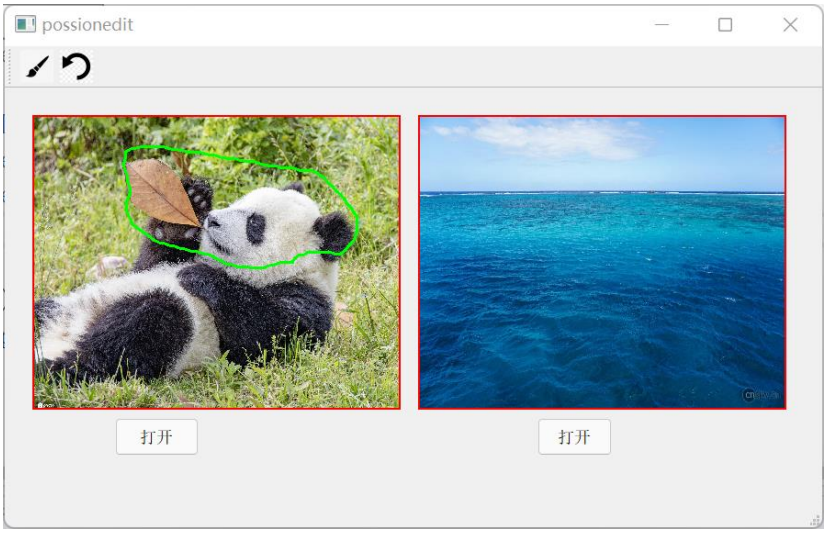


Figure.3 样例 1



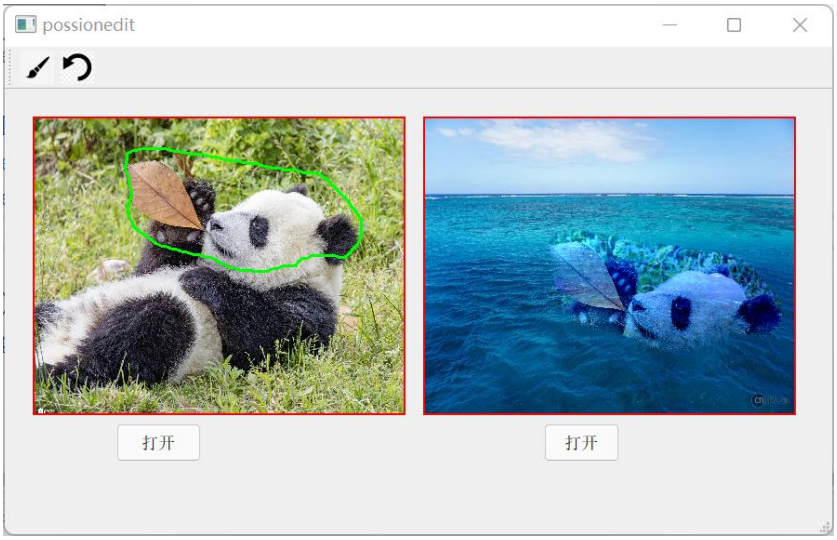
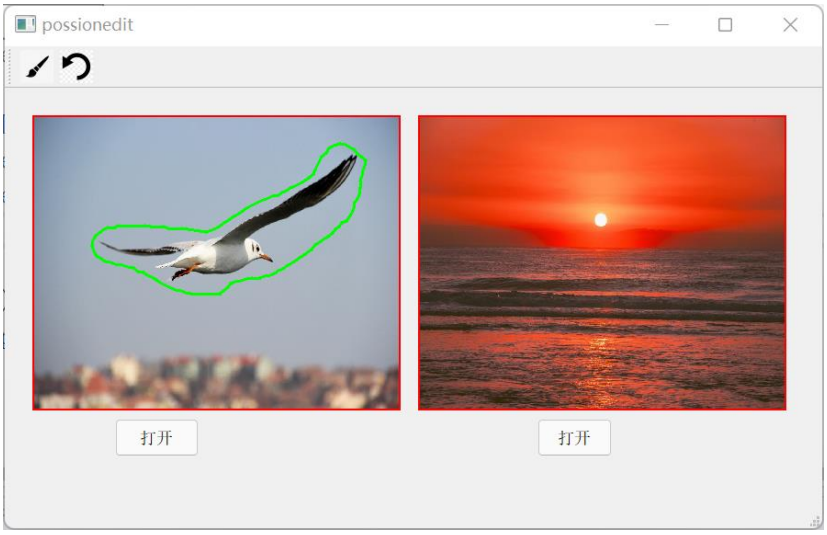


Figure.4 样例 2



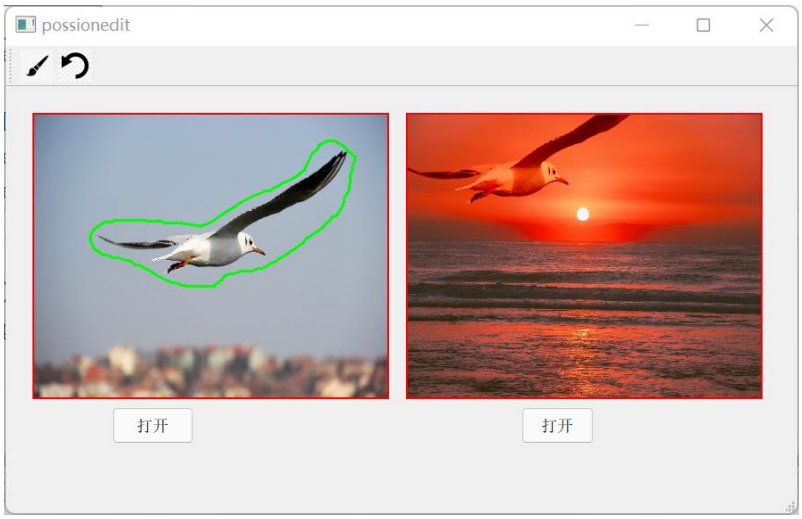


Figure.5 样例 3

八、问题与展望

8.1 遇到的问题

- 还没有更新多次选取区域时的拖动操作
- 没有更新图像的放大缩小（选取之后放缩）
- 扫描线算法没有针对多边形自相交进行改进
- 更方便的图形化界面

8.2 future work

- 上述遇到的问题的改进
- 文章中提到的图像整平，光线处理，更改颜色等操作
- 可以对图像选取加入自动选取（自动确定边界）功能