Homework 3 Poisson Image Editing

华南理工大学 曾亚军

一　实验目的

- 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 库求解大型稀疏方程组。
- 学习使用 OpenCV。

二　功能描述

本作业实现的泊松图像编辑程序界面如下

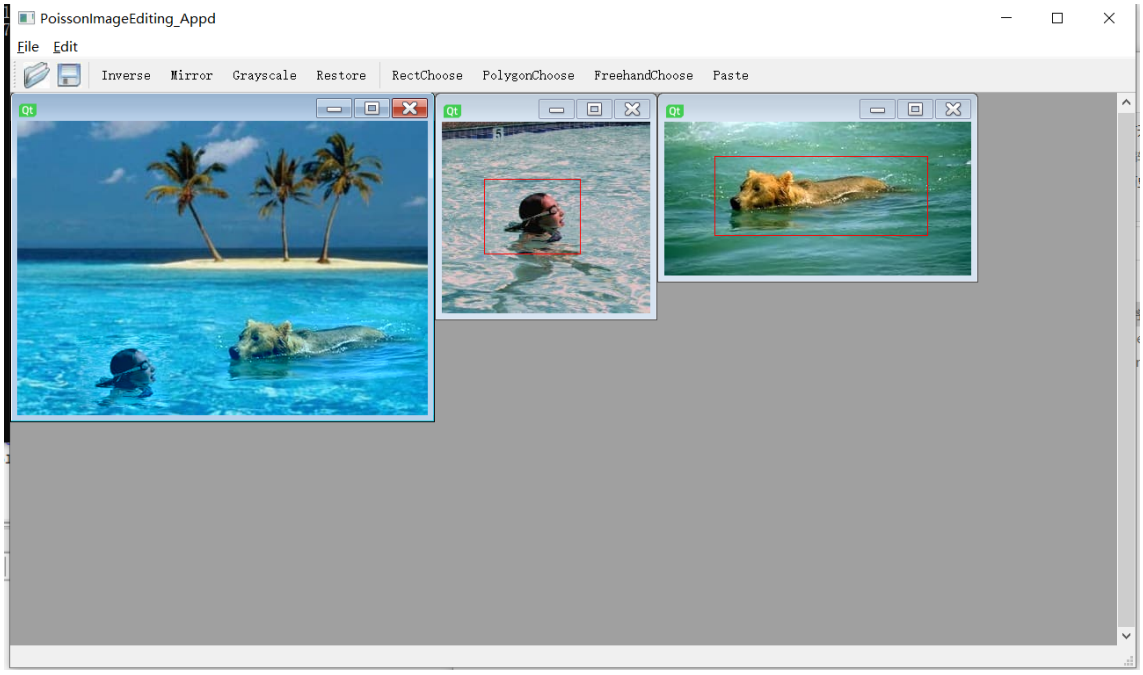


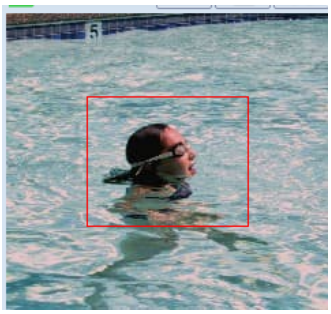
图 1: 界面

除了框架自带的功能，程序还实现了如下功能：

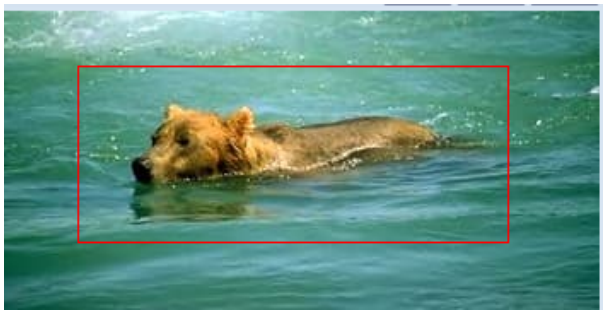
- 矩形、椭圆、多边形和自由绘制工具绘制选择框。
- 多边形扫描识别和图像融合。

- 实时拖动区域显示结果。

一些实验结果如下：



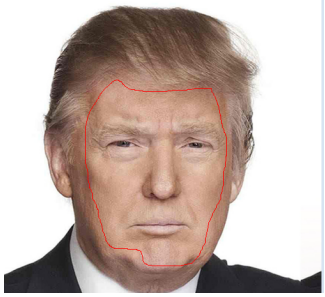
(a)



(b)

(c)

(d)



(e)

(f)

三　类设计

在类图里面我这里只展示了新建的几个类：PoissonEdit, Edge, ScanLine, 以及几个枚举类 ShapeType, PointType, DrawStatus。其中枚举类 ShapeType 表示选择工具的形式，矩形选择框，多边形选择框，自由绘制选择框。PointType 表示点的类型，标志了选择了图元后，在图像中哪些像素点是内点，哪些是外点，哪些是边界点。边结构 Edge 服务于

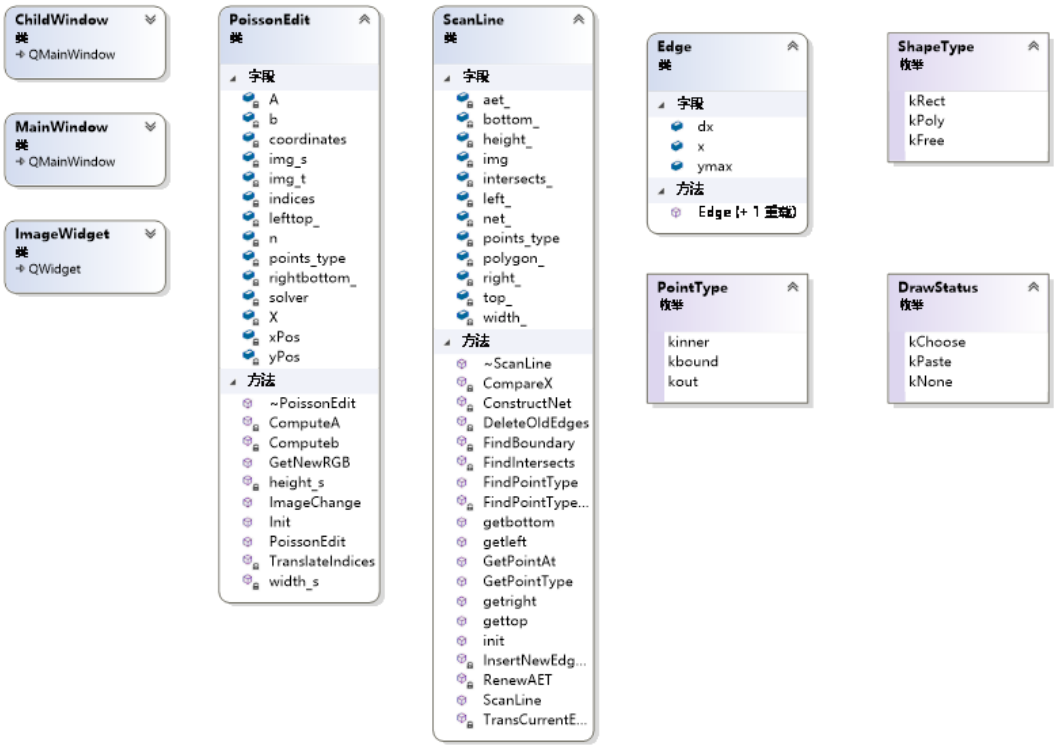


图 2: 类图

多边形扫描算法，是该算法的基本运算结构。ScanLine 用于输出图像点的类型（是否选择工具的内点或边界）。类 PoissonEdit 用于实现泊松图像编辑。

四　算法具体实现

这次的图像融合作业主要包含两个算法，多边形扫描算法和泊松图像编辑算法。在编写代码的过程中，首要的是确定多边形扫描算法会不会发生问题，是否能正确地识别内点、边界点和外电。其次再将对应的点转化成方程组系数，求解大型稀疏方程组。

4.1　多边形扫描算法

多边形扫描算法是 X 扫描线算法的改进算法。X 扫描线算法的基本流程如下：

- 确定多边形所占有的最大扫描线书，得到多边形顶点的最小和最大 Y 值（ymin、ymax）；
- 从 y = ymin 到 y = ymax，每次用一条扫描线进行填充；
- 对一条扫描线填充的过程可分为四个步骤：

  a. 求交：计算扫描线与多边形各边的交点；

  b. 排序：把所有交点按递增顺序进行排序；

  c. 交点配对：第一个与第二个，第三个与第四个；

  d. 判断内点；

其中很多细节需要注意，如：

- 交点的个数应保证为偶数个；
- 若共享顶点的两条边分别落在扫描线的两边，交点只算一个；
- 若共享顶点的两条边在扫描线的同一边，这时交点作为零个或两个；

然后，频繁的求交是非常耗时的。因而需要对 X 扫描线算法进行改进，维持一些特殊的数据结构，用于快速求交。在数学上，多边形的求交，实际上是当前位置的每一条边跟平行线进行求交。平行线 y 每次增加 1，而边的斜率是保持不变的，因而可以通过边的斜率来反应下一个交点的情况。如上图，当我们已知这条边的斜率，和上一次平行线的交点

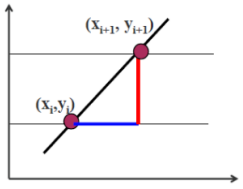


图 3: line

x_i，那么下一条平行线跟这条边的交点为：

$$x_{i+1} = x_i + \frac{1}{k},$$

其中 k 表示斜率。

因而对于每条边，我们可以维持一个特殊的数据结构 Edge, 包含如下内容：

- x：当前扫描线与边的交点坐标；
- Δx：从当前扫描线到下一条扫描线间 x 的增量；
- y_max：该边所交的最高扫描线的坐标值 ymax；

其中 $\Delta x = 1/k$

为了加快算法进度，应当维持一个新边表 (NET) 和活边表 (AET)。其中新边表是一个纵向链表，链表的长度为多边形所占有的最大扫描线数，链表的每个结点，成为一个吊桶，对应多边形覆盖的每一条扫描线；新边表 NET 的每个节点（即对应每条扫描线）存放着第一次出现的边。并按如下方法进行构造：

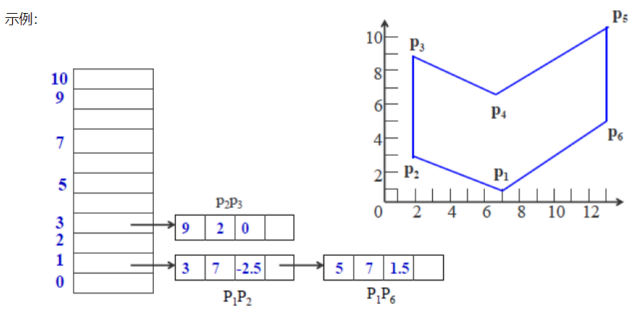


图 4: NET

而活边表 (AET) 则保存了与当前平行线相交的所有边。每一次更新（下一条扫描线），要进行如下几个操作：

- 判断 AET 中有没有需要丢弃的边（通过 ymax 进行判断）
- 更新 AET 中的边 (交点坐标通过增量进行更新)
- 从 NET 中获取当前扫描线需要加入的新的边。

当得到内点之后，边界点很好判断。遍历所有的像素点，如果不是内点，且周围有内点存在，则该像素点认为是边界点。

4.2 泊松图像编辑

泊松图像编辑算法理解原理之后就很好实现，本质上就只是求解大型稀疏方程组。泊松图像编辑的核心思想是，尽可能地保留图像的特征（即尽可能保留图像的二阶导不变），同时边界要插值对应边界（偶目标图像的对应）像素。因而可以表达为一个优化问题：

$$\min_{f} \iint_{\Omega} \|\nabla f - \boldsymbol{v}\|^2, \ s.t. \ f|_{\partial\Omega} = f^*|_{\partial\Omega}. \tag{1}$$

该变分问题可以转化为：

$$\begin{cases} \Delta f = div(\boldsymbol{v}) \\ f|_{\partial\Omega} = f^*|_{\partial\Omega} \end{cases} \tag{2}$$

将上述微分方程离散化，即可建立大型稀疏方程组。

五 实现难点

算法过程中遇到的几个难点主要在于：

1. 多边形扫描算法的细节处理

多边形扫描算法核心的步骤和思想是不难的，难的地方就在于各种特殊规则和特殊情况的判别。在多边形扫描算法的实现过程中，我为了保证算法的正确，一开始让输出的所有内点像素颜色为黑色。但会发现，对一些简单情况可以正确判断内点，但对于一些复杂情况，如五角星，凹多边形等，会发生某一条平行线判断错误等可能。因而在程序中我花了较多的时间，不断测试多边形扫描算法，改边自己算法内部的细节处理。

2. 将不规则的点转化成方程组的未知数

本作业中设计了几种选择图元，如矩形，椭圆、多边形和自由选择等方式。这以为了像素点的边界不再是规则的。而我们离散化二阶导时，计算一个点的二阶导，需要周围四个点的信息，而我在不规则的选择框下，周围的四个点的序号，未必就挨着这个点的序号。这个地方迷惑了我一段时间，后来发现，我给选择图元的内点编号后，将这个编号和它在图像中的像素坐标建立一个双射，就可以这个像素点的序号是多少，它周围的四个像素点的序号对应多少，求解得到的这个结果对应哪个像素点。

六　注意事项

- 多边形扫描算法的细节处理，需要非常多的测试和很好的耐心。

七　收获和感悟

这次作业和作业 2 都是计算机图形学在数字图像处理上的应用。而且图像变形和图像融合其实都是很贴近我们日常生活的，就会让我觉得，我现在在学习的东西，确实是以见很容易触摸的东西。事实上我对计算机图形学抱有比较强的好奇心和热情，也是因为当我第一次接触科研项目的时候，经过了漫长的理论推导和代码的 Debug，重构出来 bunny 兔子的网格模型。当时的欣喜和成就感一直留存在心中。

在这次作业中，事实上泊松图像编辑算法不难实现，归根结底，在代码中就是一个解方程的问题。整个流程理解起来也不难。但我会注意到我有一些数学知识的欠缺。泊松图像编辑，实际上是将一个变分问题转化成了偏微分方程求解的问题，而这部分我在学习泛函的时候其实没有接触到。因而也特地去重温了一下这一点。不得不说的是，数学在计算机图形学里面确实是非常重要的。