Homework3 Program Report

张景浩 PB20010399

2023.3.26

1 问题描述

按照参考文献 [1] 中给出的方法实现两个图像之间对于给定区域的泊松融合，并尝试实现实时拖动多边形区域得到结果。

2 算法原理

2.1 图像的泊松融合

- 源图像梯度：$v = \nabla I = (\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y})$
- 目标图像颜色：f^*
- 重叠区域：Ω

我们希望区域融合后能有 $\nabla f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}) = v$，但这显然是不现实的。所以我们退一步对 f 做以下的最优化：

$$arg \min_f \iint_\Omega |\nabla f - v|^2, \quad s.t.\ f\ |_{\partial\Omega} = f^*\ |_{\partial\Omega}$$

$\Longrightarrow \triangle f = div\ v \quad s.t.\ f\ |_{\partial\Omega} = f^*\ |_{\partial\Omega}$

$\Longrightarrow f_{x,y-1} + f_{x,y+1} + f_{x-1,y} + f_{x+1,y} - 4f_{x,y} = div\ v_{x,y}$

这样一来，目标区域中的色素值就可以通过求解大型线性方程组得到。

需要注意的是，在构造线性方程组时，等式的右端要先进行边界条件的处理，并且因为泊松方程对应的系数矩阵是一个大型的稀疏矩阵，所以考虑使用 matlab 中的 sparse 来降低求解稀疏矩阵的时间。

为了方便对于边界条件的处理，我们可以构造如下结构的稀疏矩阵：

$$\begin{bmatrix} -1 & \cdots & -1 & 4 & -1 & \cdots & -1 & \cdots & 0 & & \\ & \ddots & & \ddots & \ddots & \ddots & & \ddots & \cdots & & \ddots \\ & & -1 & \cdots & -1 & 4 & -1 & \cdots & -1 & \cdots & 0 \end{bmatrix}$$

这样一来，被 4 作用到的位置就是像素矩阵中位于多边形 Ω 内部的点，而只被-1 作用到的位置就是位于多边形 Ω 内部或者边界的点，只被 0 作用到的位置则是剩余的点。

2.2 实时泊松融合

使用函数 addlistener 即可，但需要注意的是，实现泊松融合的函数运行时间长会导致拖动多边形的反馈变得卡顿，这也是为什么在 2.1 中使用 sparse 来构造稀疏矩阵。

3 代码实现

本次实验使用 matlab 进行编译，代码框架由课堂上以及 assignment3 中给出，算法具体的实现详见 blendImagePossion.m 文件。

4 测试结果

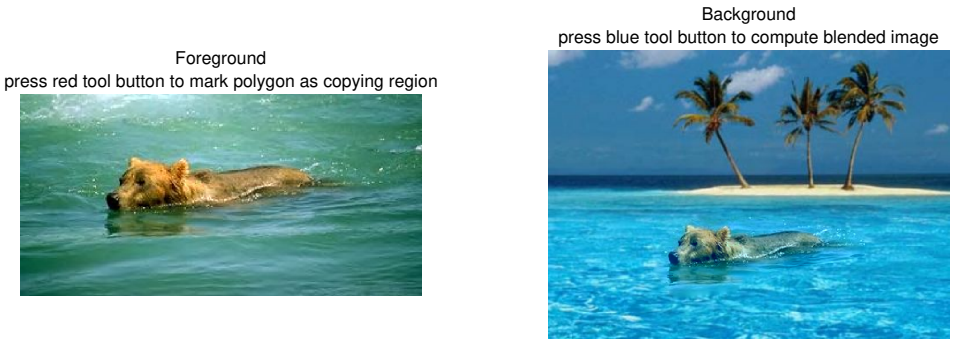

图 1: Possion Editing

5 总结

通过反复的实验操作后可以发现，在泊松融合算法下，给定多边形区域中的图像与目标背景几乎可以完全融合，并没有出现突兀的边界。但是如果仔细观察的话还是能区分出二者之间的差异，这是因为泊松融合是在源图像的色素信息上，基于目标区域边界的色素信息做了一个平滑的过渡处理。所以如果两个图像对应区域之间的色彩、纹路等信息差异过大的话，还是会出现较为明显的边界。而处理这种情况的一个可行的方法就是在选取多边形区域时尽可能地贴近目标物体，尽可能地减少源图像中背景的信息。另外还会发现，受目标多边形边界条件的影响，所融合的图像自身的色彩风格也会受到影响，如图 1 中原本的棕熊在融合近新的背景之

后，颜色多了一层水蓝色，使得真实性有所下降。

同时，为了做到移动多边形的同时显现出泊松融合的结果，需要用到 addlistener 函数及时进行功能函数的回调。在实验中发现，拖动多边形与显示泊松融合结果之间的连续性与实现函数中解泊松方程的快慢相关。

参考文献

[1] P.Perez,M.Gangnet,and A.Blake.Possion Image Editing.SIDDRAPH2003.