PossionImageEdit

SA21010060 周俊亦

Abstract

图像融合是一种十分有趣的图像处理技术应用。本报告复现了经典的图像融合算法: 泊松图像融合 (Poisson Image Editing).

1 算法原理

1.1 概述

融合两幅图像的指定区域是十分热门的应用。当然，我们希望图像融合越“自然”越好。如图 1 所示，如果生硬的剪切图像，这样的融合图像很难达到我们的预期。泊松算法将图像融合看成曲面拟合问题，如图 2 所示，将剪切图的边缘硬置成背景图像素值，通过调和函数拟合计算生成融合图像。

图 1: 图像移植

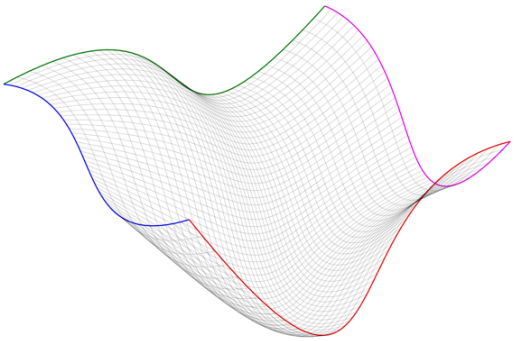

图 2: 曲面拟合

1.2 原理

从字面意思上来看，无缝融合 (SeamlessCloneing) 强调 “自然”，尽量让前景 (剪切) 图与背景图看起来像是原生一起的。自然而然的想法便是让前景图 “融入背景图”，这也是前面提到的为什么将前景边缘硬置成背景像素而不是背景边缘硬置成前景像素的原因。

由于融合后必须保留前景的语义，因此，在边缘重置后，需要拟合内部像素，使拟合后的图看起来与原图一致。一般的，图像梯度很大程度上影响了图像的语义，因此，需要让拟合后的图像与原前景图梯度相似。

稍后可知，拟合建模问题是典型的狄利克雷条件下的泊松方程，求解方法可以利用典型的最小二乘。

算法描述为:

给定: 区域 $\Omega \in \mathbb{R}^2, \Omega$上的向量场$\mathbf{v}$, 以及函数$f^* \in \mathbb{R}^2$, 求函数 f 使得:

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^* \mid \partial\Omega$$

等价于求解狄利克雷条件下的泊松方程:

$$\Delta f = \operatorname{div} \mathbf{v} \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

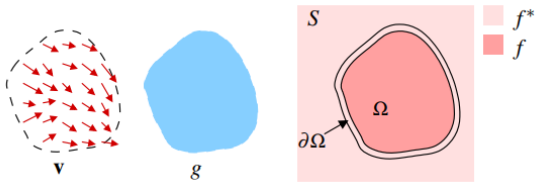

图 3: 拟合插值

1.3 Seamless Cloning

连续条件下方程为:

$$\min_{f|_\Omega} \sum_{\langle p,q\rangle \cap \Omega \neq \emptyset} (f_p - f_q - v_{pq})^2, \text{ with } f_p = f_p^*, \text{ for all } p \in \partial\Omega$$

其中:

N_p 为 p 的邻点集，$v_{pq} = \mathbf{v}\left(\frac{p+q}{2}\right) * \overrightarrow{pq}$，当 $\mathbf{v}$ 是 g 的梯度时，$v_{pq} = g(p) - g(q)$

由于图像不是连续函数，需要将上述问题转化为离散条件下的问题:

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

当 $N_p \subset \Omega$ 时:

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

向量场 $\mathbf{v}$ 的选取方式有多种，这里选取 Mixing gradients:

$$\text{for all } \mathbf{x} \in \Omega, \mathbf{v}(\mathbf{x}) = \begin{cases} \nabla f^*(\mathbf{x}) & \text{if } |\nabla f^*(\mathbf{x})| > |\nabla g(\mathbf{x})| \\ \nabla g(\mathbf{x}) & \text{otherwise} \end{cases}$$

理论上，该梯度可以自适应前后景梯度。实验验证效果良好。

2 实验分析

实验使用 Qt5 与 Eigen 库 进行算法验证，使用 MSVC2017 构建生成。

2.1 程序界面

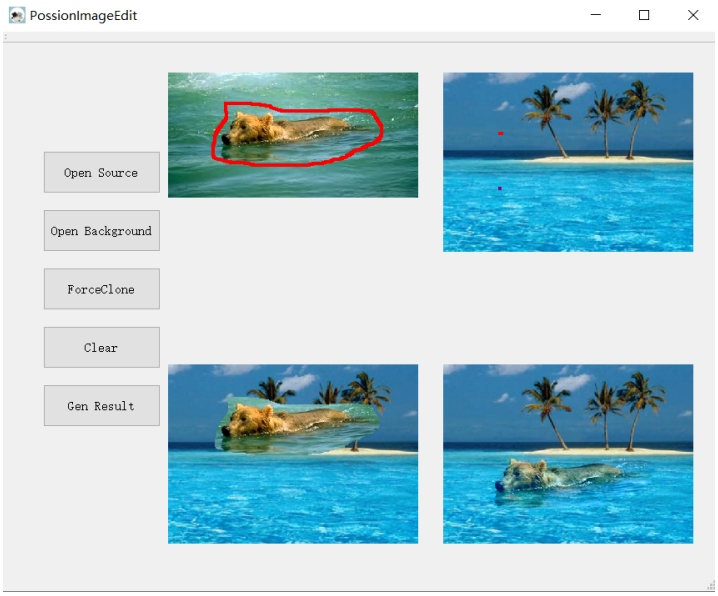

图 4: 程序界面

如图，程序可以实现任意曲线的图像融合，点击暴力融合按键可以查看暴力融合的预览，在右上角目标图像设置位移矢量的起点 (左键描点) 与终点 (右键描点) 可以改变前景与背景融合的位置。

2.2 Examples

(a) source (b) target (c) result

图 5: Ex1.

(a) source (b) target (c) result

图 6: Ex2.

(a) source (b) target (c) result

图 7: Ex3. So beautiful !!!

实验中求解大型稀疏方阵时使用 Cholesky 分解，极大的提高了方程的解的速度，比 BiCGSTAB 约快两个数量级。

实验中发现程序运行耗时主要在生成稀疏矩阵的过程里，推测可能是 Eigen 的稀疏矩阵的插入耗时。图 8 显示了 Example2 的运行时长截图。可以发现，方程求解是很快的。

```
Gen Mask take: 0.0590691seconds
Circle area contains 17803points.
Displacement vector:  0   0

Gen Sparse Matrix take: 23.8836seconds
Solve...

Solve Sparse Matrix take: 0.230418seconds
Gen result picture take: 0.0166133seconds
Total take: 24.1897seconds
```

图 8: running time

3 实验总结

又是大开眼界的一次应用实践。本次实验复现了图像融合算法，在应用中进一步体会到了数学的奇妙。

附睿客网代码链接：https://rec.ustc.edu.cn/share/8ab70680-3a56-11ec-b63a-414d96e82c31

参考文献

[1] Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. ACM Trans. Graph. 22, 3 (July 2003), 313–318.