第一次大作业实验报告

实现的功能

1. 使用计图实现卷积；
2. 自主完成建图，但可使用开源库如maxflow完成GraphCut；
3. 自主构建泊松融合矩阵、通过开源算法进行矩阵求解。

具体实现

1. 位置匹配。

我们需要提取待补全图像距离缺失区域在K个像素内的区域B，通过opencv dilate函数来实现。论文中这个数取了80，我们在opencv dilate中取核的大小为60\*60，interation=1。还尝试了另外一种核，取半径为1的核，iteration=40次，效果和之前接近。

采用jittor的元算子计算误差，然后取误差最小的区域。

2. 计算融合边界

在上一步得到了候选区域后，采用graph cut算法得到融合边界。边权重为重叠部分的差，最大流用pymaxflow计算得到，进而得到最小割，即为融合边界。

3. 自然融合

采用泊松融合算法计算。我们先构造稀疏矩阵A和向量b，然后用scipy中的共轭梯度法解方程得到融合的结果。

运行方法

|  |
| --- |
| 1. python main.py --help 2. usage: main.py [-h] original\_image\_path mask\_path patch\_image\_path output\_directory 4. positional arguments: 5. original\_image\_path  Path to the original image 6. mask\_path            Path to the mask image 7. patch\_image\_path     Path to the patch image 8. output\_directory     Directory to save the result 10. optional arguments: 11. -h, --help           show this help message **and** exit |

例如：

|  |
| --- |
| python main.py ./data/completion/input4.jpg ./data/completion/input4\_mask.jpg ./data/completion/input4\_patch.jpg ./result |

结果将保存在result文件夹中，命名方式为input{i}\_result.jpg。

耗时

生成一张图片耗时约为1分钟，大部分时间在求解方程上面。

结果

最左边是原图，旁边是未融合图像，再旁边是80\*80的核，最右边是1\*1的核。









参考的实现：  
https://github.com/parosky/poissonblending/tree/master

https://github.com/Cydiater/scene-completion/tree/main