媒体计算–Graph Cut Project 报告

2017011462　计 73　方言

1　实现的工作

- 块分割算法（基础）
- 三种块偏移生成算法（基础）
- 基于梯度的损失函数（附加）
- 处理 Surrounded 情况（附加）
- 考虑 Old Seam（附加）
- 基于 FFT 加速（附加）
- Error Region 及选择策略（附加）

2　块分割算法

2.1　构建图

考虑两张图片 A, B，定义其重合部分为 $overlap(A, B, t)$，其中 t 为当前的偏移量，需要寻找某种分割方式，使得重合区域的过渡最自然。根据 $GraphCut$ 算法，将重合部分的每一个像素点视为一个 node，相邻像素点之间连一条边，边权由损失函数定义，由此可以构建出一张图。另外定义超级源 src 和超级汇 dst 两个节点，重合部分边缘处，与图片 $A(B)$ 相邻的节点与节点 $src(dst)$ 连边，边权设为 $+\infty$。

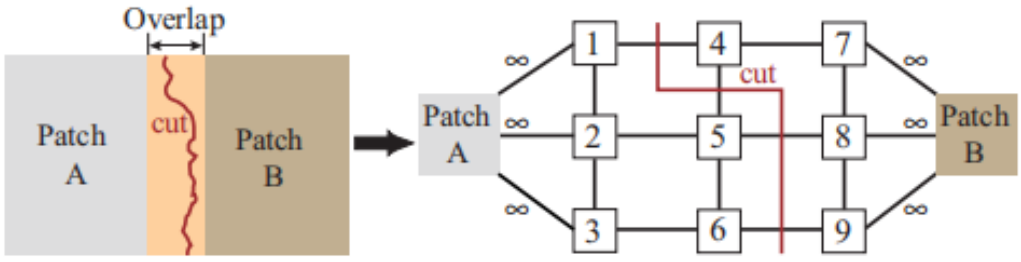

图 1: Graph Cut 算法建图过程

2.2 求解最小割

考虑损失函数：

$$M(s,t,A,B) = ||A(s) - B(s)|| + ||A(t) - B(t)||$$

其中 s,t 为两个相邻节点，$A(s), A(t)$ 表示图 A 中这两个点的像素值，对 B 同理。

这个损失函数表征的是相邻两个像素点在两张图上的相似程度，为了使得过渡自然，在上述构建出的图中，需要寻找出节点 A 到 B 的最小割，将重合区域划分为两个连通块。划分完成后，与 src 连通的像素点采用图片 A 的像素值，与 dst 连通的像素点采用图片 B 的像素值，即可完成这两张图片的拼接。

在我的实现中，采用了 $networkx$ 库用于建图以及求解最小割。

3 偏移生成算法

基于上述分割算法，进一步需要考虑每一次 new patch 放置的位置，即需要基于已有的 patch 生成合适的 offset。

3.1 Random

随机在全图范围内生成偏移。在具体实现中，为了保证每一次都能覆盖新的像素，可以在某一个未覆盖像素的附近进行随机，保证其一定会被 new patch 覆盖。

3.2 Entire Match

在全图范围内进行匹配，考虑一个 offset，设为 t，定义 SSD 损失函数为：

$$C(t) = \frac{1}{|A_t|} \sum_{p \in A_t} |I(p) - O(p+t)|^2$$

其中 $A_t = overlap(I, O, t)$，I, O 分别输入和输出的图片。之后按概率 p 选择某个 t:

$$p \propto e^{\frac{-C(t)}{k\sigma^2}}$$

其中 σ 为输入图片的标准差，k 为超参数（一般取 0.001 到 1.0，实现中取 0.01）

与随机策略类似，为了保证效率，在具体实现中，只考虑那些能够覆盖新的像素的偏移。

3.3 Subpatch Match

与上一种方法类似，首先在当前已生成的图片中选取某个完全被覆盖的 sub patch，设为 S_O，之后只需要针对这部分计算 SSD 损失，即修改损失函数为：

$$C(t) = \sum_{p \in S_O} |I(p-t) - O(p)|^2$$

之后按照类似的概率进行选择。同样的，为了保证效率，在具体实现中，将选出的 error region 作为 sub patch，并且保证 new patch 能够完全覆盖住 sub patch。

3.4 实验结果

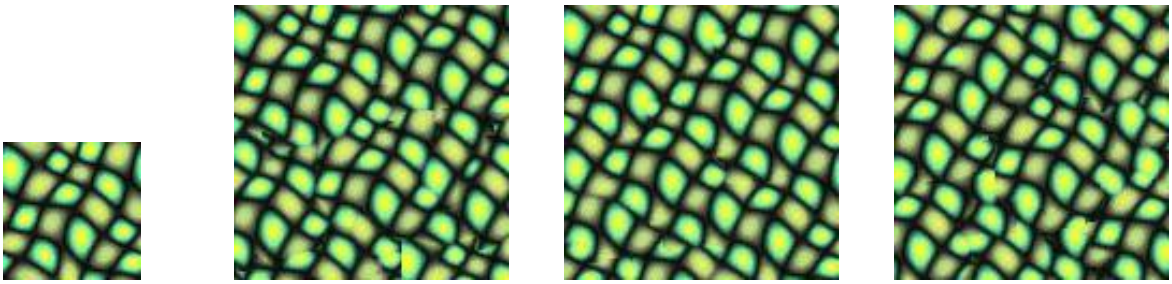

图 2: green 结果——Input，Random，Entire Matching，Subpatch Matching

图 3: akeyboard-small 结果——Input，Random，Entire Matching，Subpatch Matching

图 4: strawberries2 结果——Input，Random，Entire Matching，Subpatch Matching

4 基于梯度损失函数

改进上文提到的损失函数，在其中加入像素点附近的梯度信息，这样能够使得算法更倾向于分割梯度变化很大的区域（通常是一些边界区域）。

$$M'(s,t,A,B) = \frac{M(s,t,A,B)}{||G_A^d(s)|| + ||G_A^d(t)|| + ||G_B^d(s)|| + ||G_B^d(t)||}$$

其中 d 表示像素点 s,t 之间的方向，G_A^d, G_B^d 表示两张图片上沿方向 d 的梯度。

图 5: 使用原损失函数（左图），使用带梯度的损失函数（右图）

5 考虑 Old Seam 的分割

之前在加入 new patch 时，都是将已生成的部分当做一张完整的图片来考虑的。现在考虑重合部分中已有的分割，引入 seam node 作为新节点。最后根据最小割算法的求出的割集来判断 old seam 是否应当被 new patch 代替。

定义 A_s 为像素点 s 被拷贝到输出图片时对应的 patch。对于重合部分中每一个 old seam，和其两侧的两个节点 s,t，引入一个 seam node 节点 $seamnode(s,t)$。

- $seamnode(s,t)$ 与 new patch 对应的节点 B 连边，边权为 $M(s,t,A_s,A_t)$
- s 与 $seamnode(s,t)$ 连边，边权为 $M(s,t,A_s,B)$
- t 与 $seamnode(s,t)$ 连边，边权为 $M(s,t,B,A_t)$

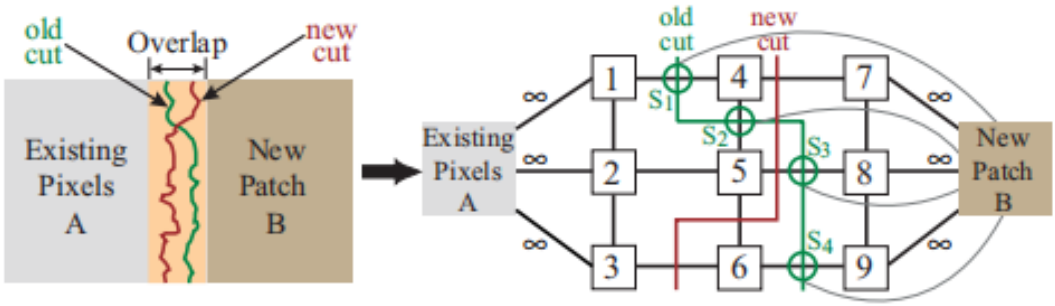

图 6: 考虑 Old Seam 构建图

对于其余节点，按照之前的方法连边。由以上方法构建图，并计算最小割。如果 seam node 与节点 B 的边被切割了，则保留原始的 old seam，否则若是 seam node 与两侧节点的边被切割了，则用被切割的边来更新此处的 seam。

具体实现时，采用额外的数据结构记录每个像素点处的 seam 信息，以及它来自的 patch 信息（通过记录对应的 offset 即可）

图 7: 不使用 old seam（左图），使用 old seam（右图）

图 8: 不使用 old seam（左图），使用 old seam（右图）

6 基于 FFT 加速

考虑块偏移生成算法中两种基于 matching 的方法，计算 SSD Cost 时，可以采用 FFT 进行加速。SSD Cost 可以写为：

$$C(t) = \sum_p I^2(p-t) + \sum_p O^2(p) - 2\sum_p I(p-t)O(p)$$

其中 $p \in S$, S 是当前需要计算的区域（对 Entire Matching 方法来说，S 为输入图片；对 Sub-patch Matching 来说，S 为当前的 sub patch）

对于第三部分，这是一个卷积的形式，使用 FFT 加速的卷积计算，复杂度为 $O(n\log(n))$，其中 n 为计算区域的像素数量。对于前两部分，需要单独考虑。

考虑计算区域 S：

- 若其为填充完全的矩形区域，则可以使用 sum table（即二维前缀和）的方法进行计算。记录 $D(i,j)$ 表示矩形区域 [1, 1, i, j] 的平方和，则对任意矩形区域 [a, b, c, d]，其区域和为 $D[c,d]-D[a,d-1]-D[c-1,b]+D[a-1,b-1]$，整体的时间复杂度为 $O(n)$。
- 若其不一定填充完全，则也可以使用类似卷积的方法计算。设 $Bitmap(O)$ 表示输出图片的 mask，已被覆盖的位置为 1，未被覆盖的位置为 0。使用 $Bitmap(O)$ 对输入图片平方进行卷积，即可得到上式中的第一部分，第二部分同理。利用 FFT 加速后，整体时间复杂度为 $O(n\log(n))$。

具体实现中，需要同时适配两种 Matching 方法，而且第三部分计算时间已经达到 $O(n\log(n))$，因此前两部分统一采用卷积方法计算。

单次时间	Original	FFT-based
Entire Matching / s	188.7	6.1
Subpatch Matching / s	62.3	4.2

表 1: 单次 Matching 用时比较

7 Error Region 及选择策略

Error Region 定义为输出图片中过渡最差的部分，而在块偏移生成算法中，可以保证 new patch 完全覆盖住这个区域。

论文中没有详细定义如何选取 error region，在我的实现中，定义一个区域的 error cost 为区域内所有 seam 的权值和。error cost 最大的区域即为 error region。类似上一部分提到的方法，这部分的计算也可以使用基于 FFT 的卷积来降低复杂度。

为了更快速填满整个输出图片，设计了选择策略。当输出图片中覆盖率小于阈值 Γ 时，error region 直接选择为同时包括已覆盖和未覆盖像素的区域，并且采用 Entire Matching 方法；覆盖率大于阈值 Γ 时，采用 Sub-patch Matching 方法。

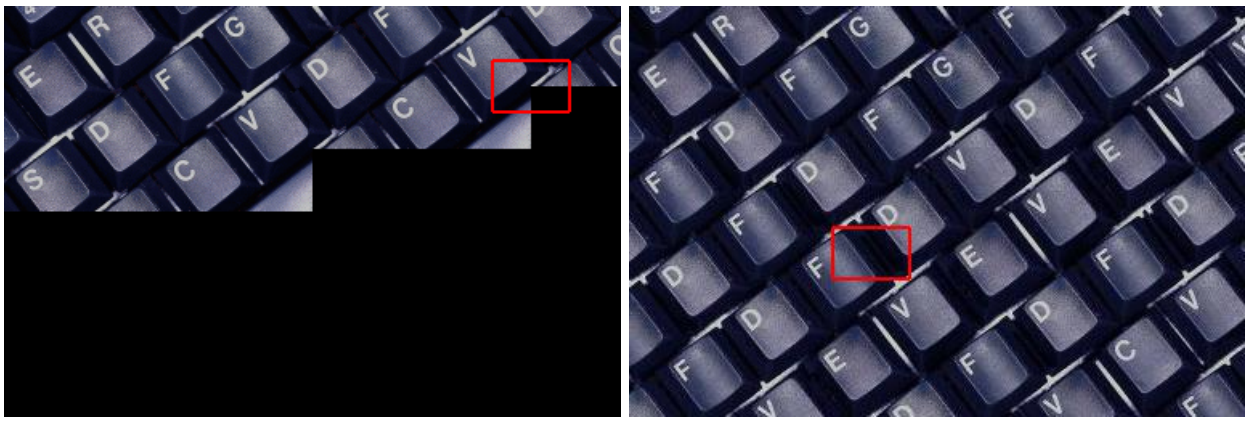

图 9: 未填充满时的 error region（左图），填充满时的 error region（右图）

8 Surrounded Region 处理

如果一个 new patch 需要覆盖在一个已经被完全覆盖的区域上，按照最初的建图方法，这个图将只有超级源，而超级汇则与其不能连通（因为重合区域没有“与 new patch 相邻的像素”）。这时可以通过强行要求重合区域的某个像素（某些像素）与 new patch 对应节点相连来处理这个情况。

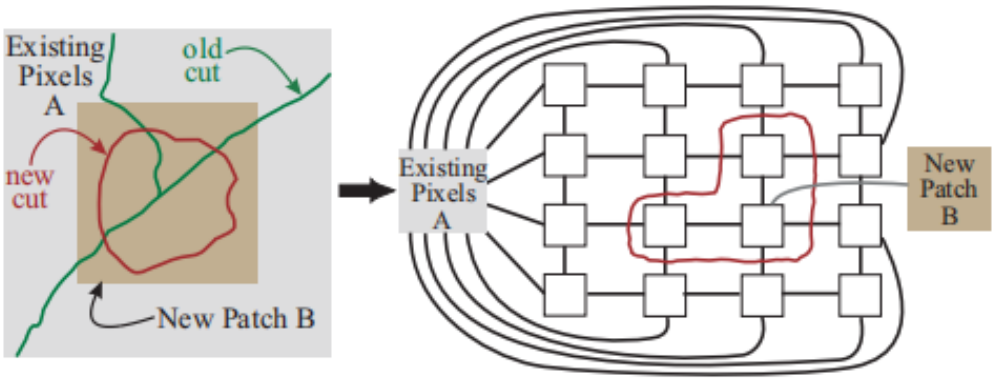

图 10: 处理 Surrounded 情况

另外，如果考虑 Old Seam，由于引入的 seam node 会与 new patch 对应节点相连，因此不需要做上述操作。

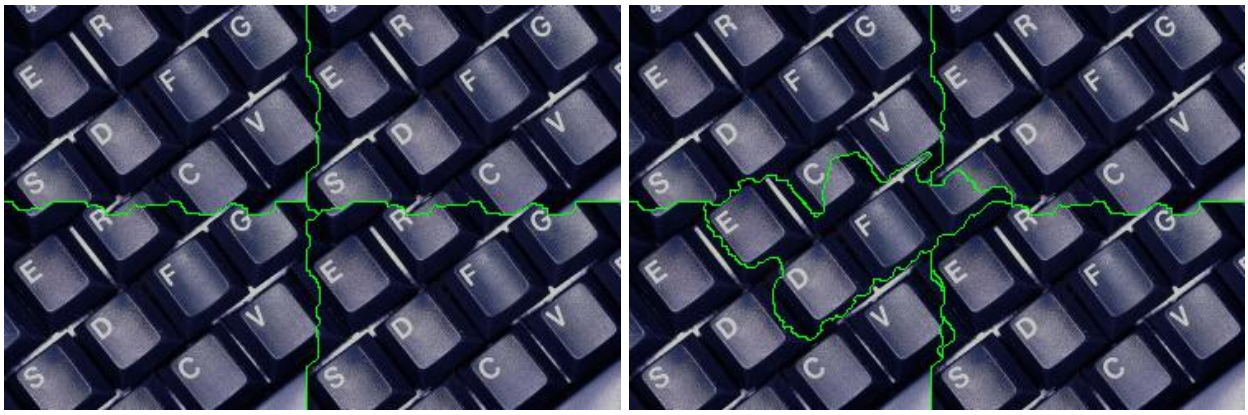

图 11: surrounded 情况处理，覆盖前（左图），覆盖后（右图）

9 总结

本项目实现了 Graph Cut 算法用于生成 Texture 及其一系列改进，最终达到比较好的效果。对于算法本质有了更深的理解，同时对于 CV，CG 相关的库使用更加熟练，受益匪浅。