CodingProject2-2 Report

张景浩 PB20010399

2023.6.8

1 问题描述

参考文章 Graphcut Textures: Image and Video Synthesis Using Graph Cuts [1] 中的算法，实现基于 graphcut 的图像纹理合成。

2 算法原理

文章中提出的纹理合成算法是通过对输入的纹理补丁进行不断地变换、复制，然后沿着一条最佳接缝拼接到一起，最后得到我们想要的输出纹理。首先，我们要寻找合适的位置来放置补丁，然后使用 graphcut 技术找到纹理补丁中的最适合复制到输出纹理中的区域。我们将整个算法大致的分为两个部分，对于每一个纹理补丁而言：

1. 比较纹理补丁和已有的输出纹理，确定纹理补丁相对于输出纹理的位置
2. 从纹理补丁中确定一个最适合复制到输出纹理中的区域（形状不规则），并将其转移到输出纹理中

为了找到纹理补丁中最适合复制到输出纹理中的区域，我们需要这个区域中的纹理与输出纹理在感知上具有相似性，这种相似性被形式化为一个 Markov 随机场 (MRF)。我们将输出纹理表示为节点的网格，其中每个节点表示输入纹理中的一个像素或像素邻域。一对节点的边缘概率取决于其像素邻域的相似性，因此输入纹理中来自相似邻域的像素最终成为生成纹理中的邻域，从而保持了输入的感知质量。所以在找到一个适合的子补丁及其相对输出纹理的位置后，我们要计算此时所有可能接缝中产生最高 MRF 可能性的接缝，文章算法通过将这个问题重新构造为一个最小代价图切割问题来找到这条最佳接缝：将一些特殊的节点添加到 MRF 网格中，并计算两个特殊的终端节点之间的最小切割。

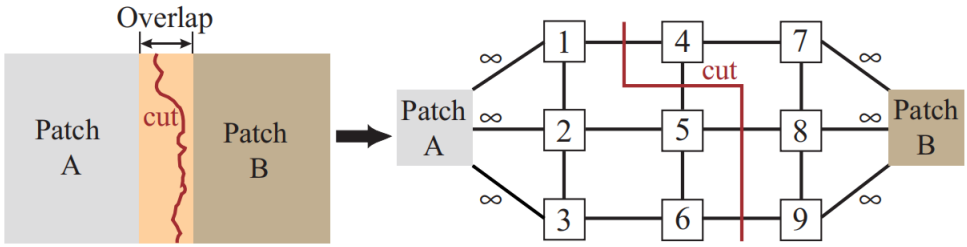


图 1: 将最佳接缝问题转换为最小图切割问题

如图(1)所示，我们将新的纹理补丁 (B) 放置到已有的输出纹理 (A) 上，并且二者有一定的重合区域，为了让二者之间的缝隙尽可能不明显，我们要找到一条接缝，在接缝的左边，纹理像素由 A 提供，在接缝的右边，纹理像素由 B 提供。接下来我们将重合区域看作是一个图，并且有两个端点 A 和 B，图中每一个内部节点都是重合区域的一个像素，节点之间的连接都是具有代价的。假设重合区域中的两个相邻的节点分别为 s 和 t，并且用 A(s),B(s) 来表示节点 s 在 A,B 中的像素值，用 A(t),B(t) 来表示节点 t 在 A,B 中的像素值，我们将节点 s 和 t 之间的连接的代价定义为:

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

现在我们就是要找到一种分割方式，使得被分割的连接的代价之和最小，这样就将最佳接缝问题转换成了一个图切割问题。

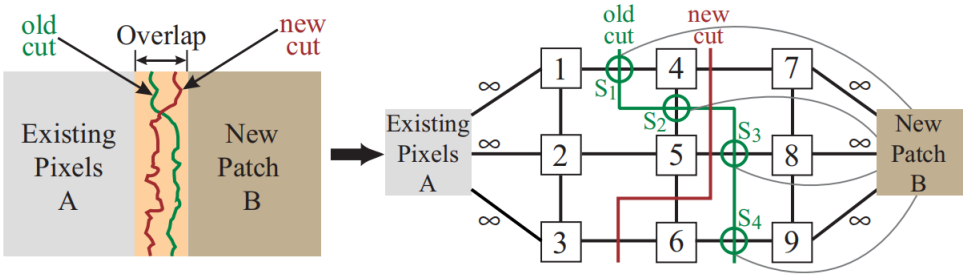


图 2: 最佳接缝的更新

通过上面的方法我们可以将两个纹理补丁以一种相似性最高的方式拼接起来，但是对于已经生成的部分输出纹理，其中存在着多个纹理补丁，并且有着隐藏的纹理接缝。那么当新的纹理补丁加入时，这些隐藏的接缝可能会被改变。将新的纹理补丁加入到上一轮已经生成的部分输出纹理时，需要保存上一轮接缝的分割代价，并将新计算的分割代价加入到老的分割代价

中，形成总体代价，并以这个代价来决定新的分割位置。如图(2)所示，我们给出一个最佳接缝的更新方式。首先，我们在旧的分割上加入缝节点 s_1, s_2, s_3, s_4，然后在这些缝节点与端点 B 之间构造新的连接，并对新生成的连接赋予代价。如表(1)所示，我们记 A_i 为节点 i 的原始纹理补丁。在得到新的图之后，我们可以使用图切割算法来更新最佳接缝，方法如下：

连接	代价
$1 \rightarrow S_1$	M(1,4,A_1,B)
$S_1 \rightarrow 4$	M(1,4,B,A_4)
$S_1 \rightarrow B$	M(1,4,A_1,A_4)

表 1: 缝节点生成的连接的代价

1. 若接缝节点与端点 B 之间的连接被分割，说明旧的接缝将保留在输出图像中
2. 若接缝节点与端点 B 之间的连接未被分割，说明旧的接缝已经被新的像素覆盖，旧接缝的成本将不计入最终成本
3. 若接缝节点与其相邻像素节点被分割，则说明在相同位置有一个新的接缝，并将一个新的接缝成本（取决于已切割的弧）添加到最终成本中

通过这个方式不断加入新的纹理补丁，用新的纹理补丁去代替旧的接缝，会使得输出纹理的接缝越来越不明显，直到代价收敛。

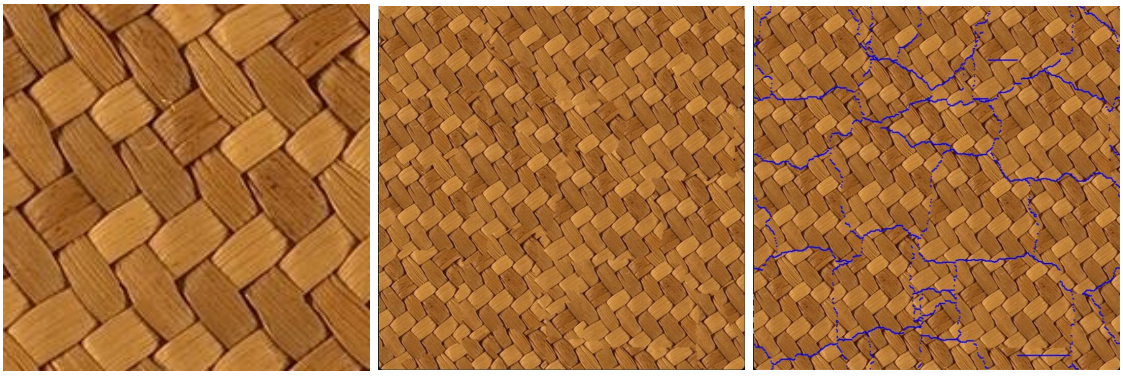
为了选取选取合适的纹理补丁并将其放到合适的相对与输出纹理的位置上，文章中提到了三种方法：

1. 随机放置：将新的纹理补丁随机放置在原有的输出纹理上，再将二者进行拼接
2. 整体匹配：遍历所有可能的平移量，并从中找出与已有的输出纹理最匹配的位置，这个过程可以通过计算二者重合区域的平方和差 (SSD) 来完成
3. 子区块匹配：先从输出纹理中找到一个子区域，然后再输入纹理中找到与这个子区域最匹配的纹理补丁

3 代码实现

本次实验代码使用了 Kuva 作为基本框架，在 Visual Studio 上编译运行。使用时在 Kuva 的解决方案属性界面的命令参数中，输入框架使用说明.txt 中的相应命令后，运行程序即可。

4 测试结果



源图像　　纹理合成　　纹理接缝

图 3

源图像　　纹理合成　　纹理接缝

图 4

源图像　　纹理合成　　纹理接缝

图 5

源图像　　纹理合成　　纹理接缝

图 6

源图像　　纹理合成　　纹理接缝

图 7

5　总结

从实验结果的纹理合成结果可以看出，文章提出的基于 graphcut 的图像纹理合成算法通过对源纹理的复制拼接，合成了一个效果非常好的新纹理，从视觉上几乎看不出纹理拼接的痕迹。通过将纹理接缝表现在合成的纹理上我们发现，合成的纹理其实是由源纹理的大小、形状不一的子纹理进行拼接得到，这些子纹理在接缝处的差异非常小，所以合成的纹理很难看到这些接缝。

参考文献

[1] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," *ACM Trans. Graph.*, p. 277–286, jul.