基于运行时纹理合成的纹理画刷*)

邹 昆 韩国强 李 闻 张潇元

(华南理工大学计算机科学与工程学院 广州 510641)

摘 要 纹理画刷是一种交互式纹理生成工具,能够在用户控制下生成所需要的纹理。本文提出了一种基于运行时纹理合成的纹理画刷实现算法。纹理合成采用逐块合成的方式,每次将整个样本图以选定的位移放置到合成图中,然后用 Graph Cut 来决定最终的输出区域。在预处理过程中计算得到两个相同样本图间的最佳切合位移集合,在合成过程中,贴块位移搜索范围限定在由此集合及用户控制所决定的一个很小的范围内,使合成速度达到实时。另外通过对"孤立块"采取"虚拟贴块法"选取贴块位移,较好地保持了画刷所生成纹理的一致性。实验结果表明该纹理画刷的纹理生成速度满足与用户交互的需求,且生成纹理的质量高。

关键词 纹理合成,纹理画刷,用户控制,Graph Cut

Texture Brush Based on Run-time Texture Synthesis

ZOU Kun HAN Guo-qiang LI Wen ZHANG Xiao-yuan

(School of Computer Science and Engineering, South China University of Technology, Guangzhou 510641, China)

Abstract This paper proposes a texture brush based on run-time texture synthesis. In each step of synthesis, the sample texture is placed into the output with a selected displacement, and graph cut technique is used to determine the optimal patch boundaries in the overlap region. As a preprocessing process, the matching errors for all relative displacements between two samples are calculated and an optimal set containing displacements with the smallest matching errors is obtained. In the process of run-time synthesis under user control, the displacements for patch pasting are searched within a very small scope determined by the optimal set and user control, so the speed is very fast. By using "virtual patch pasting" method to select displacements for "isolated patches", the consistencies of the generated textures are guaranteed. Experimental results show that the texture brush can generate textures with high quality in real-time.

Keywords Texture synthesis, Texture brush, User control, Graph cut

1 引言

基于样图的纹理合成是计算机图形学、计算机视觉和图像处理领域的一个研究热点,它被广泛地应用于物体的真实感显示、图像修复等方面。纹理合成的目标是根据给定的样本纹理,合成出与样本相似的任意大的纹理图像。本文提出了一种能够实时贴块的纹理合成算法,并将其用于实现基于运行时纹理合成的纹理画刷。

画刷是很重要的一种图像编辑工具,但目前常用图像编辑软件中的画刷都是单色画刷或带有固定几种纹理的画刷。如果用户能够自己选择任意纹理样本来产生相应的纹理画刷,那么对于图像编辑和图像创作是非常有意义的。纹理画刷可以通过带用户控制的基于样图的纹理合成来实现,在用户选定样本纹理和画刷属性(如画刷的形状、尺寸等)后,随着画刷在用户控制下的移动,在画刷轨迹上合成出与样本相似的纹理。

纹理画刷与用户交互特性决定了所使用的纹理合成方法必须是实时的,这种实时性包括两个方面:一个是预处理的实时性,用户在选定样本图像后能够立刻使用该画刷,而不需要较长的等待,这样需要较长预处理时间的纹理合成方法是不适用的;另一个是合成过程的实时性,即随着画刷的移动,能实时地合成出所需要的纹理,让用户及时看到效果。

目前大多纹理合成算法都按照某一固定顺序(如扫描线、螺旋线顺序等)进行合成,后面合成的纹理对之前合成的纹理有依赖性,这样在只需求一部分纹理时,得先按顺序合成出前面所有的纹理,这样时间上不能满足纹理画刷这种需要与用户交互的应用。运行时的纹理合成很好地解决了这个问题,即在运行时根据应用需求合成出相应部分的纹理。这类纹理合成方法可以按任意顺序进行合成,且合成出的纹理是一致的。这一点对于纹理画刷也非常重要,因为我们希望无论用户怎么控制画刷,在画刷轨迹交汇处所生成的纹理都是光滑一致的。

本文在前期工作[1]的基础上提出一种能够实时贴块的纹理合成方法,在加入用户控制并做适应性改变后,将其用于纹理画刷的实现——基于运行时纹理合成的纹理画刷;另外通过采用"虚拟贴块法"控制"孤立块"(周围区域没有已贴块)的贴块位移,使得画刷生成纹理的一致性得到了较好的保持。这种方法在时间上和效果上很好地满足了纹理画刷与用户交互的需求,且操作简便,用户无需设置任何合成参数。

2 相关研究

在众多基于样图的纹理合成方法中,使用最广泛的是基于最相似邻域搜索的区域增长的纹理合成方法。此类方法一般基于 Markov 随机场(Markov random field, MRF)模型,在合成一个像素或一块像素区域时,先获取该像素或块的已合成邻域,然后在样本图像中搜索与该邻域最相似的邻域,将对应像素点或块输出到合成图中。根据每次合成区域的大小可分为逐点合成方法[2-4]和逐块合成方法[5-7]。一般来说,逐点合成方法在控制上更加灵活,但合成速度相对较慢,而逐块合成方法速度较快,且能完好保持块内的结构特征,但块间容易

*)本课题得到国家自然科学基金项目(60573019)、广东省自然科学基金项目(05103541,05300198,07300561)的资助。邹 昆 博士生,主要研究方向为数字图像处理技术等;韩国强 教授,博导,主要研究方向为多媒体数据压缩与图像重建技术等;李 闻 博士生,主要研究方向为数字图像处理技术;张潇元 博士生,主要研究方向为图像处理及信息安全。

出现不一致。

Kwatra 等人[7]提出了一种基于 Graph Cut 的逐块合成方法，将整个样本每次以不同的位移放置到合成图中，然后用 Graph Cut 来决定最终的输出区域，这样输出块的大小、形状是不规则的。该方法的优点是无需设置块大小参数，另外通过 Graph Cut 技术能较精确地最小化块间重叠区域的误差。但由于没有考虑纹理的结构特征，在块间结构连续性保持上效果不好，另外在 Graph Cut 时有可能出现误切割情况。文献[1]在其基础上提出一种新的基于 Graph Cut 的纹理合成方法，通过引入边界图使得块间结构连续性保持得更好，另外改进了原建图方法，使得切割更优，但算法速度还达不到实时。

本文在前期工作[1]的基础上提出一种新的贴块位移搜索和选取方法。在预处理过程中确定两个样本图像间的最佳切合位移集合，合成过程中的贴块位移搜索范围限定在由此集合及用户控制所决定的一个很小的候选位移集合内，使贴块速度达到实时。由于采用了快速傅里叶变换(FFT)进行加速，预处理过程耗时非常少。

Ashikhmin[4]在纹理合成中引入用户控制，使用用户绘制的目标图来控制纹理合成。Hertzmann 等人[8]也提出一种交互式纹理合成方法，使合成按照用户绘制的草图来进行。这两种方法都是基于点的纹理合成方法。Ramanarayanan 和 Bala[9]提出一种基于块的带约束的纹理合成方法，用能量最小化来使得合成图与目标图匹配。以上三种方法虽然引入了用户控制，但都是在合成前引入的控制，而纹理画刷需要的是合成过程中的用户控制，对算法速度要求更高。

运行时的纹理合成算法能较好地满足纹理画刷的应用需求。Wei 和 Levoy[10]，以及 Lefebvre 和 Hoppe[11,12]先后提出了几种运行时的纹理合成算法，这几种算法的合成结果都是与顺序无关(Order-Independent)的，即无论按什么顺序合成，合成的结果都是相同的，这样合成纹理的一致性得到了保证，但它们都需要较长的预处理时间。本文通过加入对“孤立块”的贴块位移的控制来解决这个问题，虽然这样合成的纹理是与合成顺序有关的，但是光滑一致的。

3 纹理合成算法

由于本文算法基于前期工作[1]，本节将首先简要介绍文献[1]中的纹理合成算法。本文算法保留了其中边界图生成和 Graph Cut 方法，提出了基于最佳切合位移集合的实时贴块算法。

3.1 文献[1]的纹理合成算法

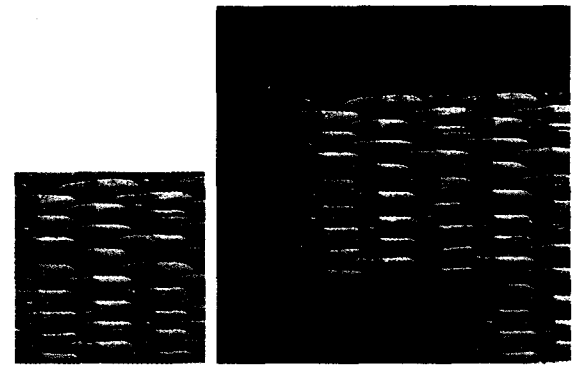
该算法是一种逐块合成方法，将整个样本块每次以不同的位移放置到合成图中，然后用 Graph Cut 方法决定最终的输出区域，如图 1 所示。为了保持块间的结构连续性，引入了边界图(Edge Map)，在计算最佳贴块位移时，同时考虑重叠区域的 RGB 值误差和边界图误差，在拷贝样本区域到合成图 O 中时，同时拷贝对应的边界图到 O 的边界图 $Edge_O$ 中。算法步骤如下。

Step 1　用边缘检测方法生成样本图像 I 的边界图 $Edge_I$。边界图是一幅二值图像，边界处像素值为 0，非边界处为 255。

Step 2　将样本图 I 以随机位移拷贝到合成图 O 中。

Step 3　寻找匹配误差最小的贴块位移。计算公式如下：

$$MatchError(t)=\frac{1}{|A_t|}\sum_{p\in A_t}\{|I(p)-O(p+t)|^2+K|Edges_I(p)-Edge_o(p+t)|^2\} \quad (1)$$



(a) 纹理样本　　(b) 合成三块后的图像

图 1　合成过程示意图

其中 t 为贴块位移，$|A_t|$ 为重叠区域 A_t 的面积，K 为权重系数，决定边界匹配误差的权重。在计算时使用 FFT 和平方和查找表(Summed Squared Table，SST)进行加速[7]。

Step 4　贴块位移选定后，用改进的 Graph Cut 方法决定重叠区域的像素取值。

Step 5　如果存在未合成区域，则转 Step 3。

在合成完毕后，原算法还提供了可选的改进步骤，用新的贴块覆盖误差较大的区域来改善合成结果。文献[1]算法在搜索贴块位移时虽然使用了 FFT 和 SST 进行加速，但由于每次贴块都要计算所有位移所对应的误差，且每次贴块速度与合成图大小有关，仍达不到纹理画刷的实时合成要求。

3.2 实时贴块的纹理合成算法

我们称两个有重叠的块所对应的位移为相关位移。实验中发现，合成过程中所确定的一系列贴块位移有一定规律：相关位移之差，即相对位移，往往在一个小的固定集合中选取。因为两个样本块之间的理想切合位置是非常有限的，所以可以提前计算出理想切合位移集合，在选取贴块位移时缩小搜索范围，从而提高算法速度。

具体做法是加入一步预处理操作，计算两个样本块在所有合理相对位移下的误差：

$$MatchError(t)=\frac{1}{|A_t|}\sum_{p\in A_t}\{|I(p)-I(p+t)|^2+K|Edges_I(p)-Edge_I(p+t)|^2\} \quad (2)$$

t 为样本间的相对位移，其他符号意义与(1)式相同。计算过程和文献[1]一样使用 FFT 和 SST 进行加速，由于只需计算一次，且样本图像一般较小，因而速度非常快。

合理的相对位移是指对重叠区域面积 $|A_t|$ 的限制：

$$\frac{1}{10}|A_I|\leqslant|A_t|\leqslant\frac{9}{10}|A_I| \quad (3)$$

$|A_I|$ 是样本图像大小。过小的 $|A_t|$ 对扩充区域的限制太小，不能保证平滑过渡；而过大的 $|A_t|$ 会导致扩充区域太小，使总贴块数目增加，降低合成速度。

在得到所有合理相对位移所对应的误差后，将其保存在一个列表中，并按照误差大小排序，取前 n 个误差最小的相对位移组成最佳切合位移集合 T_{opt}。基于多样性的考虑，这 n 个位移之间还须保持一定距离。

在合成时，将使用过的贴块位移保存下来，构成集合 T_{hist}。在搜索下一个贴块位移时，仍使用式(1)计算误差，但搜索范围缩小为：

$$t\in\{t_1+t_2\,|\,t_1\in T_{hist},t_2\in T_{opt}\} \quad (4)$$

$$\text{且}\ |A_I|-|A_t|>\min\left(\frac{|A_t|}{10},\frac{ext_{\max}}{2}\right) \quad (5)$$

ext_{max}是当前情况下对合成区域可能的最大扩充面积，这样式(5)是要求位移 t 所对应的贴块须对合成区域有一定量的扩充。

称由满足式(4)和(5)的位移 t 所构成的集合为候选位移集合 T_{cand}。这样在计算得到每个候选位移所对应的误差后，在误差最小的 k 个位移中随机选取一个作为下一个贴块位移。但这样随着贴块的增多，T_{hist} 逐渐增大，T_{cand} 相应增大，搜索速度会逐渐变慢。

我们发现，每两次相邻位移搜索过程中很多计算是重复的，因为上一次贴块仅影响到与它有重叠的候选位移的误差。这样保存 T_{hist} 中每个位移在上次计算过程中所得到的 n 个匹配误差(对应 n 个优选位移)，在每次搜索过程中，对于没有受到上次贴块影响的候选位移，则直接通过查表得到匹配误差。由于 n 非常小(通常 10 以下)，内存的占用是非常少的。

当未合成区域很小时，可能出现 T_{cand} 为空的情况，即所有满足(4)的位移都不能满足(5)，这时我们根据预处理过程中的相对位移排序，依次选取 T_{opt} 之外的相对位移 t_2 加到 T_{hist} 中的位移 t_1 上，直至(5)式得到满足，然后取误差最小的位移进行贴块。

4 纹理画刷

纹理画刷工具的运作流程为：首先用户选择一幅纹理样本图像 I，然后选择画刷的属性(如画刷的形状、尺寸等)，接下来就可以使用纹理画刷了，随着纹理画刷在用户控制下在画布 C 上的移动，在移动轨迹上合成出与样本相似的纹理。

实现方式为：

1)在用户选定纹理样本后，进行预处理操作，即边界图和最佳切合位移集合的生成，并在内存中生成一幅和画布同样大小的合成图 O(初始化为空)。

2)当画刷在画布上移动时，将轨迹颜色显示为样本纹理的平均色，并将待合成点坐标加入队列 L。

3)当 L 不为空时，启动另一线程进行纹理合成。设待合成点为 p，如果 $O(p)$已经合成，则直接输出到 $C(p)$，否则先在 O 上进行一次贴块，要求该贴块必须覆盖 p 点。

在将上一节所述的实时贴块的纹理合成算法用于实现纹理画刷之前还须做一定修改，转变为基于用户控制的运行时纹理合成算法：

1)搜寻贴块位移时搜索范围进一步缩小，因为当前贴块必须覆盖待合成点 p，即加入限制条件：

$$p \in I_t \tag{6}$$

I_t 为位移为 t 的样本块。

2)(5)式中的 ext_{max} 目前意义为在满足(4)和(6)的条件下对合成区域可能的最大扩充面积。

3)满足(4)(5)(6)式的位移 t 构成候选位移集合 T_{cand}，为了保持合成过程的一定的随机性，T_{cand} 不能太小，因此可以适量增加优选位移集合的元素数目 n 来达到这个目的。因为有式(6)的约束，计算量也是很小的。

4)由于式(6)的约束，贴块位移搜索时的计算量不会随着贴块数目的增加而增大，因此不用再保存 T_{hist} 中每个位移在上次计算过程中所得到的匹配误差。

5)“孤立块”的贴块位移的选取。当待合成点 p 周围没有已合成点时，包含 p 点的贴块不可能与已合成区域重叠，此时的贴块称为“孤立块”。

显然，按照前面的处理方法，T_{cand} 必为空。虽然孤立块周围没有已合成区域，但如果其贴块位移选择不当，仍可能会在后续的合成中出现不一致的情况，尤其是当样本为规则纹理或近规则纹理时，如图 2 (a)所示。图中显示了两笔画刷产生的纹理，第一笔从上到下，第二笔从左到右。显然，在第二笔起始时的第一次贴块为孤立块，图 2 (a)显示的是对孤立块采用随机位移选取(当然要覆盖待合成点)的方式所产生的效果，当第二笔刷到右边与第一笔所产生的纹理相遇时，不一致情况发生了。这是因为对于孤立块采用随机选取位移的方式可能破坏纹理的规则性。

(a) 随机贴块法

(b) 虚拟贴块法

图 2 对孤立块的两种贴块方式的效果对比
(画刷宽 60 像素，样本纹理见图 1 (a))

我们采用了“虚拟贴块”的方法来选取孤立块的贴块位移，从与待合成点 p 距离最近的历史贴块开始，向 p 点方向进行虚拟贴块，直至能覆盖到 p 点。具体步骤如下：

Step 1 首先找到距离待合成点 p 最近的历史贴块 I_{ta}。

Step 2 对于所有

$$t \in \{t_a + t_2 \mid t_2 \in T_{opt}\} \tag{7}$$

计算 I_t 到 p 点的距离，取距离最近的贴块 I_{tb}。

Step 3 如果 I_{tb} 没有覆盖 p 点，则 $t_a = t_b$，转 Step2，否则 t_b 为最终选取的贴块位移。

通过虚拟贴块方法进行规则性传递，能较好地保持画刷轨迹相交处纹理的一致性，如图 2 (b)所示，由于是虚拟贴块，没有误差计算，速度是非常快的。

综上所述，当需要合成点 p 时，采取的具体步骤如下：

Step 1 判断 p 点是否为孤立点。是则采用最优虚拟贴块法选取贴块位移，转 Step 5；否则继续。

Step 2 根据(4)(5)(6)式得到候选位移集合 T_{cand}。如果 T_{cand} 为空则转 Step 4，否则继续。

Step 3 计算 T_{cand} 中所有位移所对应的误差，取误差最小的 k 个位移，随机选取一个为贴块位移，转 Step 5。

Step 4 根据预处理中得到的相对位移按误差排序的列表，依次选取 T_{opt} 以外的相对位移 t_2 代入式(4)中，直至式(5)和(6)得到满足，然后取误差最小的位移为贴块位移。

Step 5 将样本图以选定的贴块位移放置到合成图 O 中，对于孤立块则直接输出，否则用文献[1]中的 Graph Cut 方法决定最终输出区域。返回 $O(p)$的值。

5 实验结果

所有实验都在配置为 P4 3.4G CPU，1G 内存的机器上

进行，使用 VC 6.0 进行编程。边界匹配误差权重 K 取 1，最佳切合位移集合大小 n 取 15，贴块位移在误差最小的 3 个位移中选取(k 取 3)。

图 3 给出了用纹理画刷所生成的纹理图像，小图为纹理样本，输出图大小均为 360×360，绘制时全部采取随机涂抹的方式，直至输出图全部被填满。从图中可以看出，从随机纹理到规则纹理，纹理的绘制质量都是比较理想的。图 4 给出了两个纹理画刷在其他图片上绘制的实例，样本图片为上方的两幅小图。

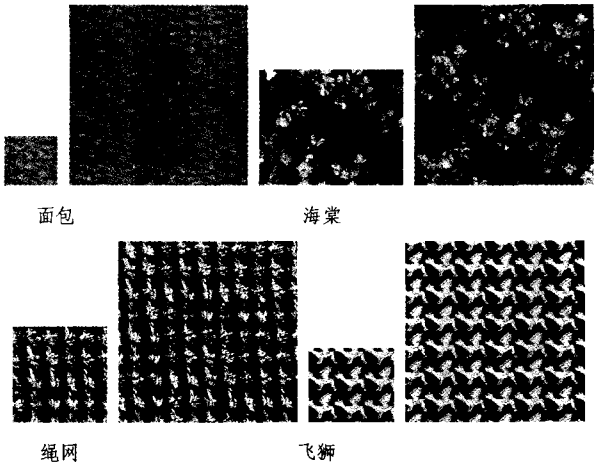


图 3 部分纹理样本及由纹理画刷生成的纹理

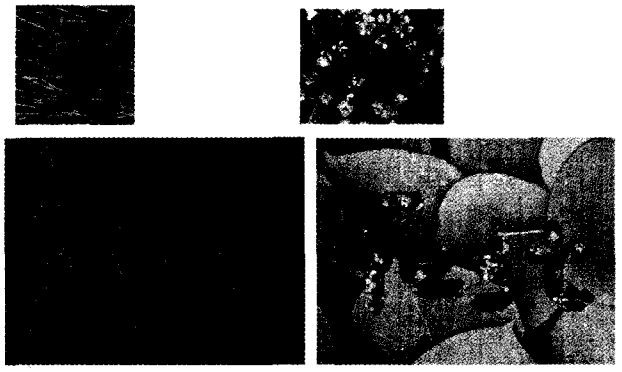


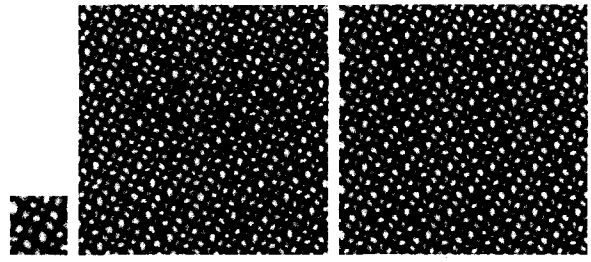
图 4 两个纹理画刷绘制实例

表 1 纹理画刷的时间参数

纹理样本	样本尺寸	预处理时间	平均贴块时间
面包	108×99	0.132s	0.016s
海棠	288×231	0.757s	0.107s
绳网	192×192	0.571s	0.096s
飞狮	172×145	0.356s	0.044s

表 1 列出了图 3 中的四种样本所对应的纹理画刷在绘制中的时间参数。预处理主要包括边界图和最佳切合位移集合生成，由于采用了 FFT 和 SST 进行加速，预处理时间均在 1 秒以内，一般较大的纹理样本的预处理时间也相对较长。平均贴块时间包括了贴块位移搜索时间和 Graph Cut 时间，由于位移搜索范围非常小，因而速度非常快，一般对于较大的纹理样本，所用的平均贴块时间要长，因为其可能的重叠区域面积更大，计算匹配误差和进行 Graph Cut 时计算量更大。

虽然对于大多数纹理样本，纹理画刷都能生成令人满意的纹理图像，但与没有用户控制的算法(3.2 节算法)相比，在某些纹理的合成质量上相对要差一些，图 5 给了一个例子。因为加入用户控制后，可选择的贴块位移更少，如果相应方向上的最佳切合位移对应误差较大，则会出现一定的瑕疵。



(a) 样本　　(b) 画刷生成纹理　　(c) 自动生成的纹理

图 5 画刷生成纹理与自动生成的纹理的对比

结束语 本文在前期工作[1]的基础上提出了一种实时贴块的纹理合成方法，并将其用于图像编辑工具——纹理画刷的实现。合成时将样本图像每次以选定的位移放置到合成图中，然后用 Graph Cut 方法决定最终的输出区域。在选取贴块位移时，搜索范围限定在由事先计算好的最佳切合位移以及用户控制所决定的一个很小的范围内，大大提高了贴块速度，使得用户控制下的实时纹理合成成为可能。另外通过对孤立块位移选取采取虚拟贴块的方式，使得合成出的纹理的一致性得到较好的保持，且用户在使用纹理画刷时，只用设置画刷属性，无需设置任何合成参数，非常简便。实验结果显示，本文纹理画刷生成纹理的质量和速度都是令人满意的。本文下一步研究工作主要包括更多的用户交互方式以及合成质量的进一步提高。

参 考 文 献

[1] Zou Kun, Han Guoqiang, Li Wen, et al. Edge map-based texture synthesis via graph cuts // International Conference on Convergence Information Technology. 2007

[2] Efros A A, Lung T K. Texture synthesis by non-parametric sampling // Proceedings of IEEE International Conference on Computer Vision. 1999:1033-1038

[3] Wei L Y, Levoy M. Fast texture synthesis using tree-structured vector quantization // Proceedings of ACM SIGGRAPH. 2000: 479-488

[4] Ashikhmin M. Synthesizing natural textures // Proceedings of the ACM Symposium on Interactive 3D Graphics. 2001:217-226

[5] Efros A A, Freeman W T. Image quilting for texture synthesis and transfer // Proceedings of ACM SIGGRAPH. 2001:341-347

[6] Liang L, Liu C, Xu Y Q, et al. Real-time texture synthesis using patch-based sampling. ACM Transactions on Graphics, 2001, 20(3):127-150

[7] Kwatra V, Schodl A, Essa I, et al. Graphcut textures: image and video synthesis using graph cuts. ACM Transactions on Graphics, 2003, 22(3):277-286

[8] Hertzmann A, Jacobs C E, Oliver N, et al. Image analogies // Proceedings of ACM SIGGRAPH. 2001:327-340

[9] Ramanarayanan G, Bala K. Constrained texture synthesis via energy minization. IEEE Transactions on Visualzation and Computer Graphics, 2007, 13(1):167-178

[10] Wei L Y, Levoy M. Order-independent texture synthesis. http://graphics.stanford.edu/papers/texture-synthesis-sig03/. 2003

[11] Lefebvre S, Hoppe H. Parallel controllable texture synthesis // Proceedings of ACM SIGGRAPH. 2005:777-786

[12] Lefebvre S, Hoppe H. Appearance-space texture synthesis // ACM Transactions on Graphics, 2006, 25(3):541-548