

大尺寸纹理的实时合成^{*}

陈 昕^{1,2}, 王文成¹⁺

¹(中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190)

²(中国科学院 研究生院,北京 100049)

Real-Time Synthesis of Large Textures

CHEN Xin^{1,2}, WANG Wen-Cheng¹⁺

¹(State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

²(Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: whn@ios.ac.cn

Chen X, Wang WC. Real-Time synthesis of large textures. *Journal of Software*, 2009,20(Suppl.):193-201.

<http://www.jos.org.cn/1000-9825/09023.htm>

Abstract: This paper presents a novel technique for synthesizing large textures of high quality in real time. By analyzing the texture periodicity, patches in an optimized size are generated to well represent the variation of exemplar features. Then, during synthesizing, the paper first distributes patches on the output texture with a vacant region left between any pair of neighboring patches in every row and every column, where a vacant region is also in the same size as a patch. Thereafter, patches are selected to fill the vacant regions in the output texture. Obviously, both patch distributing and vacant region filling can be executed in parallel. To accelerate, for each patch, the paper constructs a set of matching patches that can be efficiently merged with the corresponding patch. The computation of patch selection for vacant regions, therefore, can be simplified into set intersection. Moreover, patch distribution and set intersection are performed on a CPU while patch stitching are executed on a GPU, to take advantage of both CPU and GPU. Experimental results show that the presented method is able to generate a high-quality texture in 1024*1024 pixels at over 45 frames per second, which is hard to achieve by existing techniques.

Key words: texture synthesis; large texture; real-time; parallel; GPU

摘 要: 提出一种纹理合成方法,可实时高质量地生成大纹理.它先基于纹理特征变化的周期性分析,得到合适的纹理块尺寸,以使所划分的纹理块能高效反映这种周期性变化,便于生成高质量的纹理;然后,它在目标纹理上均衡地分布纹理块,使得垂直方向和水平方向上相邻的纹理块之间都留有一个块尺寸大小的空白区域,再对空白区域进行填充,以完成目标纹理的生成.显然,布块操作和填充操作均可并行地进行.同时,为每个纹理块预先生成可与其邻接匹配的纹理块集合,以便在填充计算时可用简便的集合求交计算来进行邻域约束的搜索,并将这种求交计算放在CPU中进行,而将邻接纹理块在重叠区域的缝合计算放到GPU中进行,以综合利用CPU和GPU的优势.实验表明,新方法可在一般微机上以45帧/秒的速度高质量地实时合成1024*1024的大纹理,而这是已有技术难以达到的.

关键词: 纹理合成;大尺寸纹理;实时;并行;GPU

* Supported by National Natural Science Foundation of China under Grant Nos.60773026, 60873182, 60833007 (国家自然科学基金)

Received 2009-05-15; Accepted 2009-07-23

纹理合成根据小块样本纹理生成视觉上类似的大块纹理,能有效重用光照信息,以提高绘制效率.这方面的工作很多,已能很好地合成中等尺寸的纹理.但是,对于大尺寸纹理的合成,已有工作还难以实时地进行,这严重制约了纹理合成技术的实践应用,比如需要广泛应用纹理的地形生成以及显示墙的绘制计算等方面.因此,提高纹理合成速度,一直是国际上研讨的热点问题.

自 1999 年引入Markov概率模型来进行纹理合成后^[1],合成计算的效率已得到很大的提高.根据该概率模型,纹理中任一位置的色彩是由它局部有限邻域的条件决定的.因此,根据邻域情况就能逐步地扩展,以完成目标纹理的生成.在此,邻域搜索的效率是决定纹理合成效率的一个关键因素.为此,人们提出了大量的组织和搜索技术,如用树结构组织纹理的邻域空间^[2]、用K-coherence来限定每次只搜索最相关的少数数据^[3]等.早先的纹理合成技术,每次只是根据邻域生成一个新像素的色彩.这样的计算能较好地反映纹理变化的丰富性,但对于纹理结构化信息的保持以及快速计算是不利的.2001 年,人们提出了块合成方法^[4,5],每次可根据邻域的情况生成一个包含很多像素的纹理块,这能很好地加快合成速度,并有利于保持纹理块内部的结构化信息.而 2003 年提出的Wang Tiles方法^[6],则能更加快速地合成纹理,因为它事先对纹理块之间的可拼接性组合进行预处理,以使合成计算时不进行邻域搜索的计算,但该方法可处理的纹理种类不是很多,且合成质量不是很高.以上的各项技术,都是串行地进行,难以利用GPU的并行计算能力.为此,2005 年提出了并行可控纹理合成的技术^[7],将串行的约束关系转换为基于层次化树结构中的父子结点之间的约束关系,由此可逐步求精地利用GPU并行地计算,以获得更快的速度,这是目前已知的合成速度最快的方法之一.最近,文献[8]提出一种快速的纹理合成方法,能以每秒数帧的速度生成 $1024*1024$ 的大纹理,它主要是利用纹理块之间的匹配相容性,预先为每个纹理块生成可与其拼接的相容纹理块集合,以在合成计算时进行较简便的集合求交计算来加快邻域检索计算.但是该方法基本上是串行进行的,难以利用GPU进行加速.

本文提出一种新的合成方法,改变了合成时邻域约束计算的方式,由此可很好地利用 GPU 进行并行计算以加速.对于大多数纹理,它基本上可以以 45 帧/秒以上的速度实时合成 $1024*1024$ 的大纹理,并具有很高的合成质量.该方法的主要思路是:在目标纹理上均匀地分布纹理块,使得水平或垂直方向上相邻的块之间留有一个块尺寸大小的空白区域;然后,再基于邻域匹配的搜索,为每个空白区域找到合适的纹理块进行粘贴,以完成目标纹理的生成.该方法的合成计算中,布块操作与空白区域的填补操作均可并行地快速进行,因为它们各自之间的关联性很小.因此,该方法可利用 GPU 进行并行化的计算以加速.为使空白填补计算时能较方便地找到合适的纹理块,减少目标纹理中相邻块之间的相容性冲突,我们利用[9]的工作对样本纹理进行周期性分析,以使所生成的纹理块尺寸能高效地反映纹理特征变化的周期性,由此提高纹理合成的质量.与此同时,我们也如同文献[8]中那样,为每个纹理块预先建立可与其拼接的相容纹理块集合,以便在搜索合适的纹理块以填补空白区域时,可进行简便的集合求交计算,以进一步加速.

本文第 1 节对相关工作进行对比讨论,在第 2 节对新方法进行详细的介绍,然后,在第 3 节对实验结果进行分析,并在第 4 节进行总结.

1 相关工作

根据处理对象的不同,纹理合成方法可分为两类:点合成方法^[1-3,10,11]和块合成方法^[4-6,12,13].点合成方法每次生成一个像素点的色彩,而块合成方法则是每次生成一个包含许多像素的纹理块.一般而言,点合成方法便于反映纹理变化的多样性,但不利于保持纹理的结构化信息,且合成速度相对较慢;而块合成方法的速度较快,能较好地保持块内的纹理特征信息,但块之间的色彩过渡可能不很平滑,会引起合成质量的下降.根据合成计算流程的不同,合成方法也可分为两类:串行合成方法和并行合成方法.目前大多数合成方法是串行的,而并行合成方法多数是关于点合成计算的^[7,14-16].由于并行性的引入,并行合成方法相对于串行方法具有很大的速度优势,但逐点计算的方式,合成质量不是太高.本文提出的方法是一个基于块的并行合成方法,它与相关工作的比较,在下面分别讨论.

块纹理合成方法有很多.2000 年,Xu 等人最先提出了带有块合成思想的Chaos Mosaic方法^[13],以边界重叠的

方式随机布块并对重叠区域进行融合,但合成结果质量较差.2001 年,提出了两种基于邻域搜索的方法^[4,5],很好地提高了质量.其中,文献[4]的方法通过寻找相邻块在重叠区域的最小误差接缝进行块的缝合,而文献[5]的方法则对重叠区域进行“羽化”(feathering)的色彩融合计算.2003 年,文献[12]提出的基于Graph Cut的方法,将重叠区域的缝合当作最大流/最小割问题进行处理,进一步改善了质量.同年提出的Wang Tiles方法^[6],则在预计算时求出纹理块之间可组合拼接的模式,以便在合成计算时节省邻域搜索的开销,取得了很高的合成速度,但由于其所生成的可组合拼接的纹理块集合一般不大,合成结果缺乏多样性且可适用的纹理类型不多.以上这些方法都是串行计算的,不利于进一步的加速,难以实时合成高质量的大纹理.而本文提出的新方法利用GPU并行执行,能实时生成高质量的大纹理,并对大多数纹理有效.

邻域搜索是合成计算中开销最大的部分,因此,这是加速方法研究的重点内容.文献[2]提出固定邻域尺寸的大小,以便于建立规范的树结构进行管理,从而提高邻域检索的效率.K-coherence方法虽然最先在点合成方法中提出^[3],但也可在块合成方法中得到应用.该方法预先为每个纹理块计算出可拼接的K个最相似纹理块,然后的合成计算时只在该集合中寻找合适的纹理块,以降低搜索空间.Jump Map方法^[11]则是为每个纹理像素记录可与其拼接的下一个像素的可能的位置集合,以便合成计算时随机选用,而不进行邻域匹配的计算.随后,该方法又被拓展,应用于基于块的纹理合成方法中^[17].文献[8]则为每个纹理块预先建立可拼接的纹理块集合,将搜索计算简化为求交计算,能进一步提高邻域搜索的效率,但其合成方法是串行的.本文也采用文献[8]的邻域搜索方法,但却是并行地进行,具有更快的合成速度.

合成顺序方面,随着图形硬件的发展,并行化的合成方法得到越来越多的关注.2003 年,Wei等人提出了顺序无关的纹理合成方法^[16],可并行实现.2004 年,文献[18]将Wang Tiles方法并行化以在图形硬件上高速实现,但其适用的纹理种类不多且合成质量有限.2005 年,Lefebvre等人提出并行可控纹理合成方法^[7],建立层次多分辨结构对纹理进行逐步求精的计算,在此,合成计算主要是根据父子结点的约束进行的,能很好地利用GPU进行并行计算,这是目前所知的最快的纹理合成方法之一.为提高其合成质量,文献[19]将邻域信息也记录并参与合成计算中,但其速度较文献[7]中的方法略慢.2008 年提出的多尺度纹理合成方法^[14],以文献[7]的工作为基础,可实现多尺度纹理的合成,但其合成质量受文献[7]方法的影响,并不是很高.本文的新方法也是并行化的合成,但不建立层次结构,可比方法^[7]具有更高的合成速度和更高的合成质量.

除加速技术外,还有许多工作注重于提高纹理合成的质量,如改善纹理结构化信息合成的方法^[20,21],基于全局优化的方法^[15]等.其中,文献[20,21]是串行工作的,而全局优化的方法^[15],通过全局的相似性计算,降低全局性误差,可生成质量很高的纹理,它亦可并行实现,但速度很慢.文献[22]中的方法通过引入K-coherence以及PCA等技术使得文献[15]的方法可在GPU上高速实现.虽然全局优化方法能很好地保持纹理的全局性特征,但对于局部的细节特征难以很好地处理,因此其合成质量还不是很理想.新方法通过考虑纹理的周期性特征,能很好地保持纹理的特征,以生成高质量的纹理,同时,它能实时地生成大纹理,而这是已有方法难以达到的.

2 大尺寸纹理的并行合成

本文提出的并行合成方法,是先在目标纹理上分布一些纹理块,使得水平和垂直方向上相邻的纹理块之间都是间隔一个空白区域,其尺寸大小就是一个纹理块的大小;然后,根据周围已分布的纹理情况,选用合适的纹理块填补空白区域,以完成目标纹理的生成.其工作流程,如图 1 中所示.在这工作流程中,布块计算时,各个纹理块之间没有约束关系,而在填充计算时,各个空白区域也是各自独立地进行,因此,这两步操作均可并行计算.

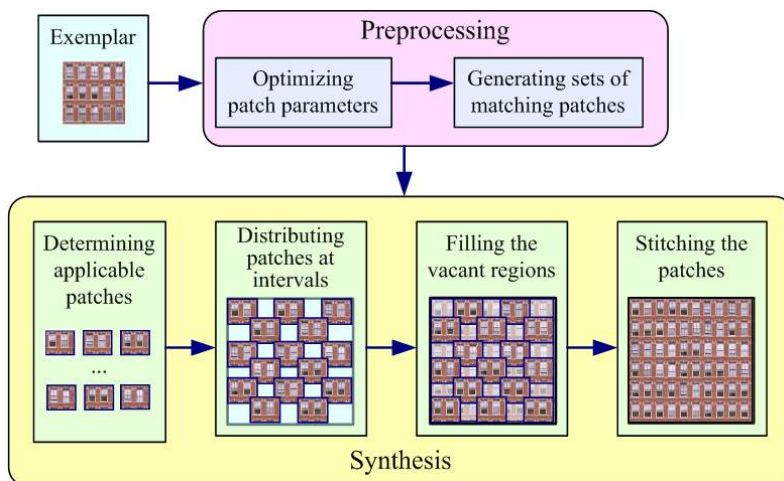


Fig.1 The pipeline of the new method

图1 新方法的工作流程

每个纹理都有其固有的特征变化趋势.如果这些趋势在布块过程中得不到有效的处理,就会在空白区域填充时引起纹理合成计算的匹配冲突,严重影响合成的质量.为此,我们采取以下两个措施,使得合成计算时能有效保持纹理特征变化的趋势,使得空白区域填充时能高效缝合邻接的纹理块,以生成高质量的纹理.

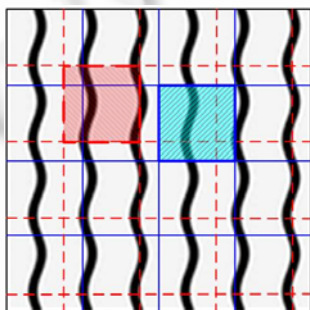


Fig.2 Texture phases. The red lines and the blue lines represent two divisions of the sample texture into patches. The grid cells formed by the red lines represent the patches in a different phase from the grid cells of the blue lines

图2 纹理的相位.图中的实线和虚线是按照块大小对样本纹理所做的两种不同的块划分,同色框内的块具有相同的相位,实线块和虚线块为两种不同相位的块

找到能高效反映纹理特征周期性变化的块尺寸.因为纹理的特征变化是有周期性的,所以,我们找到这样的块尺寸,就可使邻接块之间的缝合也能很好地符合这种周期性变化,以降低空白区域填充时的匹配冲突.

每次合成计算时,只选用相位相近的纹理块.在此,仿照周期函数的描述,对样本纹理进行一次按块大小的均匀划分所得纹理块,就是同一相位的纹理块;不同位置的划分,就产生不同相位的纹理块.如图2所示,实线框和虚线框所表示的纹理块,就是两种不同相位的纹理块,同种线框表示同一相位的块.这样的处理,使得同相位的纹理块反映的纹理特征变化趋势比较相近,有利于减少邻接块之间的匹配冲突.

在填充空白区域时,要根据周围已有纹理进行邻域搜索,以找到合适的纹理块,并进行块之间的缝合计算.在此,我们采纳文献[8]中的方法,为每个纹理块先生成可与它邻接匹配的相容纹理块集合,然后在合成计算时,根据周围已有的纹理块,找到它们各自对应的相容纹理块集合,从这些集合的交集中随机选用一个纹理块,并在该纹理块与周围纹理重叠的区域进行“羽化”计算^[5]以缝合纹理块,即对重叠位置的像素进行色彩的线性插值计算, $C = C_A * w_A + C_B * w_B$, 这里, C 表示融合后所得的色彩, C_A 和 C_B 分别表示来自 A 纹理块和 B 纹理块在这同一像素位置的色彩, w_A 和 w_B 则表示插值计算的权值,与该像素到重叠区域边界的距离相关.如果这些相容性集合的交集为空,则选一个在这些集合中出现次数最多的纹理块进行填充计算.“羽化”计算对于大多数纹理的合成是有效的,

但不利于处理特征比较强烈的结构化信息,因为它是一种平滑算子.对此,我们可在预计算时,为每个纹理块与它的相容纹理块的邻接匹配进行逐个的最佳缝合计算,并将结果保存,以在合成计算时调用.但我们目前没有进行这样的实现,因为这不是本文讨论的重点.

综上,新方法分为两个部分:预处理和合成计算.

预处理时,所进行的工作有以下几项:

- 1) 计算优化的块尺寸大小和用于约束搜索的邻域宽度;
- 2) 根据所得的块尺寸对样本纹理进行划分,并将所得到纹理块按它们各自的相位进行组织;
- 3) 为每个纹理块,生成可分别在其上、下、左、右四边进行拼接的相容纹理块集合.根据文献[8]的讨论,每个这样的集合中不必包含很多的纹理块,就可生成高质量的纹理.我们的实现中只是使每个这样的集合中包含 8 个纹理块,就能满足要求.生成相容性纹理集合时,我们采用 L^2 -Norm 的误差度量方式,并用 ANN 算法^[23]选取匹配误差最小的前 8 个纹理块构成一个相容纹理块集合.

合成计算时,先选定相位相近的纹理块,构成本次合成计算适用的纹理块集合,然后从中随机选取纹理块进行布块操作;之后,根据布块的情况,对空白区域进行填充.这 2 个步骤都可并行计算.但我们发现,填补空白区域时要进行的相容纹理块集合的求交计算,在 GPU 中进行开销很大,这可能是因为并行处理的单元数较小且涉及许多分支操作;而在 CPU 中进行则有很高的效率.为此,我们先在 CPU 中进行布块操作,并为每个空白区域选定待粘贴的纹理块;然后,将这些结果输入 GPU 中,以进行邻接纹理块在重叠区域的缝合计算.这样,就能综合利用 CPU 和 GPU 的优势,使新方法的工作效率达到很高的水平.

2.1 纹理块参数的优化计算

文献[9]深入探讨了纹理块尺寸对合成效率的影响,并设计了算法,以找到能很好反映纹理特征周期性变化的块尺寸大小和用于约束搜索的重叠区域的尺寸.本文在此就利用它的方法进行纹理块尺寸的优化计算.该方法主要是探测不同尺寸和矩形形状的纹理块划分对样本纹理周期性全局特征的反映程度,以此来决定纹理块的划分.具体地,要计算两种度量参数,纹理块的信息包容性度量和纹理块的周期性度量,并以这两种度量参数都比较好的纹理块大小作为优化的纹理块尺寸.

2.1.1 纹理块的信息包容性度量

纹理块的信息包容性是指纹理块对样本纹理信息的包容反映程度.其计算步骤如下:

- (1) 计算样本纹理的灰度直方图;
- (2) 依次取出每一个纹理块,计算其灰度直方图;
- (3) 将纹理块和样本纹理的灰度直方图归一化,并计算其欧式距离.距离越近,则纹理块对样本纹理的信息全局性特征反映越好;
- (4) 度量所有同尺寸的纹理块的信息包容性,如果大部分(比如 90%以上的)纹理块都与样本纹理的全局性特征相近,那么该尺寸下的纹理块就具有好的信息包容性.

2.1.2 纹理块的周期性度量

我们将样本纹理均匀地划分成比较大的网格,然后对每一种可能尺寸的块,在各个网格中随机选取一个这样大小的参考块;随后,对每一个这样的参考块,在样本纹理中搜索与之具有相似结构特征的纹理块的个数;最后,对这些参考块所能找到的各自类似的纹理块个数进行平均.如果一种尺寸下的参考块都能找到相似的块,并且这些块的个数具有较高的平均值,则这种尺寸的块对纹理信息周期性变化的反映就比较好.

一般地,我们可以找到多个符合条件的块尺寸.从合成速度方面考虑,文献[9]倾向于选择较大的块尺寸.但是,过大的块尺寸,会减少纹理块的数目,降低合成纹理的丰富性,并降低邻接块之间缝合的效率.因此,实现新方法时,我们通常选择大小在样本纹理尺寸 $1/4$ 至 $1/2$ 之间的块尺寸,作为纹理块划分的尺寸.
















关于重叠区域宽度的尺寸,文献[9]的方法是考察每种宽度下的重叠区域对纹理块进行区分的效率,并选用高效的宽度作为重叠区域的宽度.如果某个宽度下的重叠区域相似,则它们对应的纹理块也相似,反之,则不是,那么这样的宽度就是高效的.文献[9]对纹理块的重叠区域分为左边界、上边界和左上边界分别计算.但为简化

缝合计算的复杂性,本文则进行统一的一种宽度的计算,并经实验表明,对合成质量影响不大.

3 实验结果

我们用 Visual C++2005 实现了本文的新方法,并在一台 Dell Optiplex 755 的 PC 上进行了实验.这台微机配有一个 Intel® Core™2 Duo E6550 2.33GHz 的 CPU,2G 内存,和一个 NVIDIA Geforce 8600GTS 的 GPU.新方法的预计算在 CPU 中进行,并在表 1 中列出了本文测试的样本纹理,及相关的预处理时间(没有包括纹理块参数的优化计算时间).从表 1 中可知,新方法的预处理时间并不多.

Table 1 The preprocessing time for the tested textures (s)
表 1 所测试的纹理的预处理时间(秒)

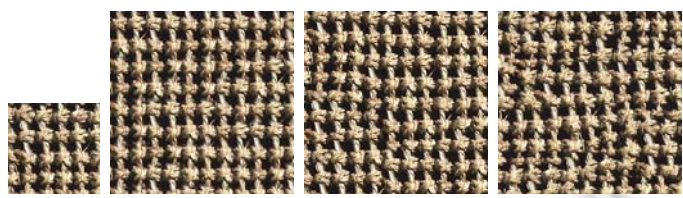
Sample textures (64*64)					
Time(s)	1.75	6.36	6.15	4.88	0.53
Sample textures (128*128)					
Time(s)	11.85	31.40	12.89	28.39	25.86
Sample textures (other sizes)					
Time(s)	32.37	9.42	9.86	18.67	8.99

通过测试这些纹理合成不同大小纹理的情况,表 2 列出了新方法合成不同尺寸纹理的时间开销,及各尺寸下的每秒平均能处理的像素数.根据SIGGRAPH'2008 上发表的论文^[14],作为目前合成速度最快的一种方法,并行可控纹理合成方法^[7]在配置有一个GeForce 8800 GTX的GPU的机器上的处理效率是 15.3M像素/秒.而新方法在较低GPU配置的情况下,合成 512*512 像素的纹理时的处理效率就达 20.15M像素/秒,并且合成的纹理越大,其处理效率越高.因此,新方法的合成效率是要超过并行可控纹理合成方法的,很适合生成大纹理.文献[8]的方法在生成 1024*1024 这样的大纹理时,只有每秒数帧的效率,而新方法可以以 45 帧/秒以上的速度进行计算,因此,新方法的工作效率也是要好于文献[8]的方法的.

Table 2 The efficiency of the new method
表 2 新方法的合成计算效率

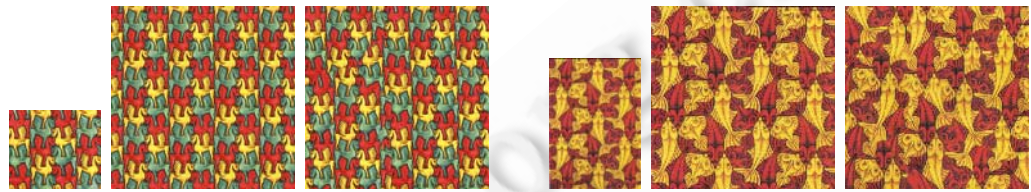
Texture size (pixels)	Synthesis time (ms)	Synthesized pixels per second (M)
256*256	10.23	6.11
512*512	12.41	20.15
1024*1024	21.78	45.91
2048*2048	58.93	67.88

由于新方法是基于纹理特征的周期性分析来工作的,所划分的纹理块对纹理特征的周期性变化有很好的反映,因此,它能高效地处理纹理合成过程中的特征变化,生成高质量的纹理.图 3 中列出了新方法、全局优化合成方法^[15]和并行可控纹理合成方法^[7]生成的一些对比纹理,其中,对比的纹理是从这些文献相关网页中下载的.从中可见,新方法生成的纹理质量要优于并行可控纹理合成方法,与全局优化方法相当,甚至某些结果还要优于全局优化方法.图 4 中还列出了新方法生成的其它许多纹理,它们都是 256*256 像素的,图 5 中列出了 1024*1024 像素的大纹理的合成结果,这些结果都有很高的质量且合成速度很快.从这些实验结果可知,新方法具有很高的工作效率,能生成高质量的纹理.但由于新方法采用“羽化”的方法进行邻接纹理块的缝合,所以,它还难以高质量地处理结构化信息比较强烈的纹理,比如图 3(c)中右边的纹理.



(a) From left to right: the sample texture, the results by the new method, texture optimization^[15], and the parallel controllable method^[7]

(a) 从左至右:样本纹理、新方法的结果、全局优化方法的结果^[15]、并行可控方法的结果^[7]



(b) From left to right: sample textures, the results by the new method and texture optimization^[15]

(b) 从左至右:样本纹理、新方法的结果、全局优化方法的结果^[15]



(c) From left to right: sample textures, the results by the new method and the parallel controllable method^[7]

(c) 从左至右:样本纹理、新方法的结果、并行可控方法的结果^[7]

Fig.3 Comparison between the new method, texture optimization^[15] and the parallel controllable method^[7]

图 3 新方法、全局优化方法^[15]和并行可控纹理合成方法^[7]所生成纹理的对比

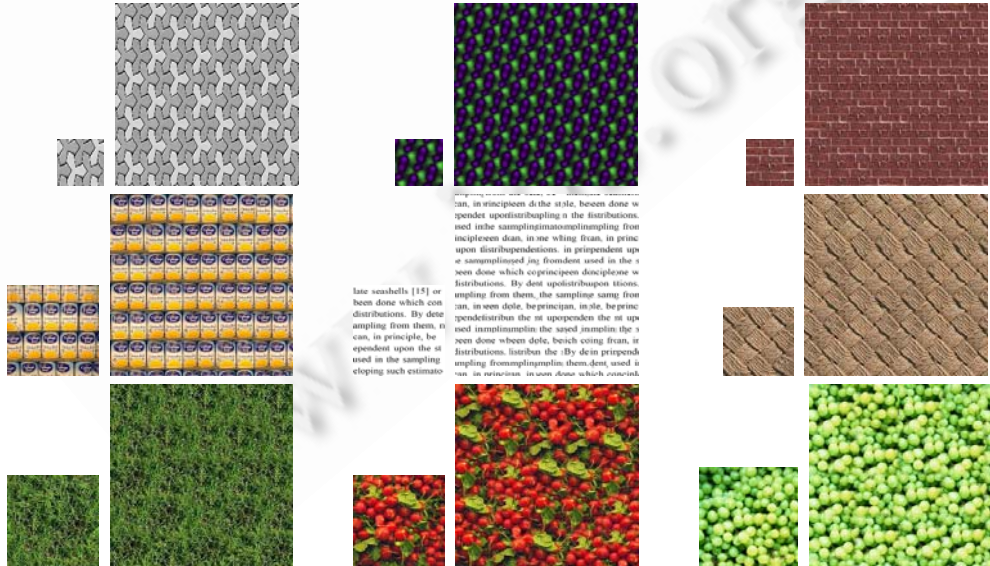


Fig. 4 Synthesis results in 256*256 pixels by the new method, where the sample textures are to the left, and the synthesized results to the right

图 4 新方法所生成的多种纹理(256*256),这里,样本纹理位于左侧,合成结果在右侧

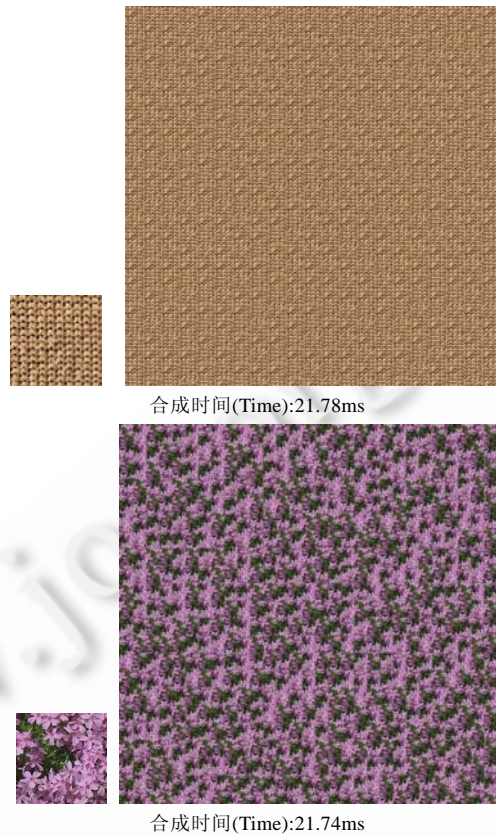


Fig. 5 Synthesis results in 1024*1024 pixels by the new method, where the sample textures are to the left, the synthesized textures to the right, and the synthesis time is in the below

图5 新方法所生成的大尺寸纹理(1024*1024),这里,样本纹理位于左侧,合成纹理在右边,下面是合成时间

4 结 语

本文提出一种并行计算的纹理合成方法,可以实时地生成大尺寸纹理.首先,它对样本纹理进行特征的周期性分析,使所生成的纹理块能高效地反映纹理特征变化的周期性,以利于提高纹理合成的质量.然后,它在合成计算时先进行间隔性的布块操作,使得垂直和水平方向上相邻的纹理块之间留有一个块尺寸大小的空白区域,再根据布块情况对空白区域进行填补.显然,布块操作和填补操作均可并行进行.为进一步加速计算,新方法为每个纹理块生成可与其邻接匹配的相容性纹理块集合,以便在填补空白区域时可用便捷的集合求交计算来选择合适的纹理块,并将此难以在 GPU 中进行的集合求交计算放在 CPU 中进行,由此综合利用 CPU 和 GPU 的优势,使得纹理合成能高效地进行.实验表明,新方法可高质量地生成大尺度纹理,能以 45 帧/秒以上的速度合成 1024*1024 的大纹理,而这是已有技术难以达到的.

因为邻接纹理块之间的高效缝合依然是个很困难的问题,因此,新方法在处理结构化信息比较强的纹理时质量不是太好,这是我们将要进一步研究的问题.

References:

- [1] Efros AA, Leung TK. Texture synthesis by non-parametric sampling. In: Proc. of the Int'l Conf. on Computer Vision-Volume 2. Washington: IEEE Computer Society, 1999. 1033-1038.
- [2] Wei L, Levoy M. Fast texture synthesis using tree-structured vector quantization. In: Proc. of the ACM SIGGRAPH 2000, ACM Press/ACM SIGGRAPH. New Orleans. Annual Conference Series, ACM, 2000. 479-488.

- [3] Tong X, Zhang J, Liu L, Wang X, Guo B, Shum HY. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics*, 2002,21(3):665–672.
- [4] Efros AA, Freeman WT. Image quilting for texture synthesis and transfer. In: *Proc. of the ACM SIGGRAPH 2001*. ACM SIGGRAPH, New York. E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series. ACM, 2001. 341–347.
- [5] Liang L, Liu C, Xu Y, Guo B, Shum H. Real-Time texture synthesis by patch-based sampling. *ACM Trans. on Graphics*, 2001,20(3):127–150.
- [6] Cohen MF, Shade J, Hiller S, Deussen O. Wang tiles for image and texture generation. *ACM Trans. on Graphics*, 2003,22(3):287–294.
- [7] Lefebvre S, Hoppe H. Parallel controllable texture synthesis. *ACM Trans. on Graphics*, 2005,24(3):777–786.
- [8] Wang W, Liu F, Huang P, Wu E. Texture synthesis via the matching compatibility between patches. *Science in China Series F: Information Sciences*, 2009,52(3):512–522.
- [9] Wang Y, Wang W, Wu E. Optimizing implementation of patch-based texture synthesis. *Journal of Computer-Aided Design & Computer Graphics*, 2006,18(10):1502–1507 (in Chinese with English abstract).
- [10] Ashikhmin M. Synthesizing natural texture. In: *Proc. of the 2001 Symp. on Interactive 3D Graphics*. New York: ACM, 2001. 217–226.
- [11] Zelinka S, Garland M. Towards real-time texture synthesis with Jump Map. In: *Proc. of the 13th Eurographics Workshop on Rendering*. Airela-Villa, Switzerland, Switzerland: Eurographics Association, 2002. 99–104.
- [12] Kwatra V, Schodl A, Essa I, Turk G, Bobick A. Graphcut textures: Image and video synthesis using Graph Cuts. *ACM Trans. on Graphics*, 2003,22(3):277–286.
- [13] Xu Y, Guo B, Shum H Y. Chaos mosaic: Fast and memory efficient texture synthesis. Technical Report, MSR-TR-2000-32, Microsoft Research, 2000.
- [14] Han C, Risser E, Ramamoorthi R, Grinspun E. Multiscale texture synthesis. *ACM Trans. on Graphics*, 2008,27(3),Article No.51.
- [15] Kwatra V, Essa I, Bobick A, Kwatra N. Texture optimization for example-based synthesis. *ACM Trans. on Graphics*, 2005,24(3):795–802.
- [16] Wei L, Levoy M. Order-Independent texture synthesis. <http://graphics.stanford.edu/papers/texture-synthesis-sig03>
- [17] Zelinka S, Garland M. Jump map-based interactive texture synthesis. *ACM Trans. on Graphics*, 2004,23(4):930–962.
- [18] Wei L. Tile-Based texture mapping on graphics hardware. In: *Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conf. on Graphics Hardware*. 2004. 55–63.
- [19] Lefebvre S, Hoppe H. Appearance-Space texture synthesis. In: *Proc. of the SIGGRAPH*. New York: ACM, 2006. 541–548.
- [20] Wu Q, Yu Y. Feature matching and deformation for texture synthesis. *ACM Trans. on Graphics*, 2004,21(3):364–367.
- [21] Tonietto L, Walter M, Jung C. A randomized approach for patch-based texture synthesis using wavelets. *Computer Graphics Forum*, 2006,25(4):675–684.
- [22] Huang H, Tong X, Wang W. Accelerated parallel texture optimization. *Journals of Computer Science and Technology*, 2007,22(5):761–769.
- [23] Mount DM. ANN programming manual. Department of Computer Science, University of Maryland, College Park, 2006.

附中文参考文献

- [9] 王一平,王文成,吴恩华.块纹理合成的优化计算.计算机辅助设计与图形学学报,2006,18(10):1502–1507.



陈昕(1982—),女,辽宁大连人,博士生,主要研究领域为纹理生成,真实感绘制。



王文成(1967—),男,博士,研究员,博士生导师,主要研究领域为虚拟现实,可视化,大规模场景绘制。