基于GPU加速的保结构纹理合成

汤 颖 林琦峰 肖廷哲 范 菁
(浙江工业大学计算机科学与技术学院 杭州 310023)

摘 要 提出一种基于Chamfer距离的保结构纹理合成方法。使用Chamfer距离度量纹理结构特征的相似度,在查找匹配块的同时计算纹理在颜色空间和结构特征空间的匹配度,从而解决以往纹理合成中有显著结构特征的纹理容易出现不连续的问题。但是Chamfer距离的计算量很大,而且随着纹理合成图分辨率的提高,计算成本会变得相当高昂以至于难以负担。因此,提出了基于GPU加速的保结构纹理合成方法,通过并行查找匹配块提高合成效率。实验证明本方法既提高了结构纹理的合成质量,又使得保结构纹理方法的合成速度大大提高,且与纹理合成图的分辨率无关。

关键词 纹理合成,Chamfer距离,结构特征,GPU
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.4.061

GPU-based Texture Synthesis with Preserved Structures

TANG Ying LIN Qi-feng XIAO Ting-zhe FAN Jing
(College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract We proposed a texture synthesis method based on Chamfer distance, which preserves the texture structures. We measured the similarity between texture structures by Chamfer distance and computed the distances of textures in both color space and structure feature space during the search of the matching texture patches. In this way, we solved the problem of discontinuity in synthesized textures with obvious structural patterns. However, the computational cost of Chamfer distance is very expensive, and it will become unaffordable as the resolution of the synthesized textures increases. To address this problem, we further presented a GPU-based algorithm to accelerate the computation of Chamfer distance during texture synthesis. The matching patches are searched in parallel on GPU to greatly improve the computational efficiency. Experimental results show that our method improves the quality of structured texture synthesis and accelerates the synthesis speed, which is irrelevant to the resolution of synthesized textures.

Keywords Texture synthesis, Chamfer distance, Structure feature, GPU

1 简介

基于样本的纹理合成技术根据输入的纹理样图合成任意大小的视觉上与样图类似的纹理图案,它是计算机图形学领域的研究热点之一。该技术广泛应用于纹理映射、图像编辑、图像类比等方面。

基于样本的纹理合成技术主要分为基于像素的合成和基于块的合成两种方法。基于像素的合成方法每次从样图中选取一个像素点复制到目标纹理图案中,基于块的合成方法则是每次从样图中选取一块连续的纹理合成到目标纹理图案中。通常,基于块的合成方法可以更好地保持纹理结构,并且由于不再对每个像素进行复杂的计算操作,它的执行效率将会优于基于像素的合成方法[23]。但是基于块的合成方法容易导致相邻块重叠区的不连续。以往的方法,例如图像缝合算法(Image quilting)[1],通过寻找最小误差路径较好地解决了此问题。但人们的眼睛不仅对颜色的变化敏感,还对边缘、边角等纹理的结构特征非常敏感。因此,仅仅根据边缘区域颜色值的误差查找匹配块并不准确,当选取的块在边缘等结构特征上不匹配时,图像缝合算法就会出现结构不连续的问题。

为了解决这个问题,本文提出在查找匹配块时不仅要考虑纹理的颜色信息,还要考虑纹理的结构特征。为了方便计算,我们需要定量地表示颜色相似度和纹理结构相似度。颜色相似度可以使用欧氏距离表示,而纹理结构相似度的计算则复杂许多,目前没有一个统一的方法。Chamfer距离[19]是由BARROW H. G.等人提出的计算距离变换的一种方法,该方法主要用来描述图像两个轮廓之间的相似度,而且它对旋转、噪声、尺度都有很好的鲁棒性,因此非常适合用来度量轮

到稿日期:2015-02-08 返修日期:2015-05-21 本文受国家自然科学基金(61003265,61173097),浙江省自然科学基金(LY14F020021)资助。
汤 颖(1977—),女,博士,副教授,硕士生导师,CCF会员,主要研究方向为计算机图形图像、虚拟现实和信息可视化,E-mail:ytang@zjut.edu.cn;林琦峰(1990—),男,硕士生,CCF学生会员,主要研究方向为计算机图形图像和高性能计算;肖廷哲(1988—),男,硕士,CCF学生会员,主要研究方向为计算机图形图像和高性能计算;范 菁(1969—),女,博士,教授,硕士生导师,CCF理事,主要研究方向为虚拟现实、软件工程和人工智能。

廓片段之间的相似度[20]。本文提出用 Chamfer 距离来度量纹理结构的相似度，通过更好地捕获纹理的轮廓和形状等结构特征来提高纹理合成时结构特征的匹配质量。

但是 Chamfer 距离的计算量很大，而且随着纹理样图分辨率的提高，计算量会急剧增加以至于难以负担。因此，本文进一步提出采用 GPU 并行计算来提高合成的效率，针对耗时最大的匹配块查找进行并行加速计算，从而大大加快纹理合成速度，并且使得合成速度与纹理样图分辨率无关。

2 相关工作

基于样图的纹理合成技术是计算机图形学领域的热点之一。在该技术发展的初期，研究者们一般使用基于像素的纹理合成方法，即每次合成一个像素。Efros 等人[3]首次提出了通过采样小块纹理样图来合成大块纹理图案的方法(Image quilting)。随后 Wei 等人[4]提出采用 L 形邻域和行扫描的合成顺序，并且采用多分辨率合成方法和树形结构的矢量量化方法进行加速。更进一步，Ashikhmin[5]提出根据图像局部相关性缩小领域匹配的搜索范围；Zelinka 等人[6]将纹理合成分为分析和合成两个阶段，在合成阶段基本达到了实时。随着 GPU 的发展，Lefebvre 等人[7]提出了并行可控的纹理合成算法，使用 GPU 加速达到了实时的合成。不同于基于像素的合成方法，基于块的纹理合成每次合成一个纹理块，可以更好地保持纹理的结构，并且有着更好的合成效率。Guo 等人[8]提出一种混沌拼接算法来快速合成纹理，但是这种算法的图像拼接过于随机，对结构性纹理的合成效果比较差。为了解决上述问题，Efros 等人[1]在 2001 年提出了一种图像缝合的算法，块的合成过程类似于 Wei 的方法[4]，采用了行扫描的合成顺序，使用块的边界区域来约束块的选取；他还提出通过寻找最小误差路径来缝合相邻块，提高合成质量。朱等人[8]认为这个图像缝合的算法是纹理合成研究的一个里程碑。对于基于块的纹理合成来说，相邻块的拼接是一个主要问题，Efros 等人[1]提出了图像缝合的算法，除此之外，Liang 等人[10]提出采用羽化技术处理相邻块的重叠区域，Nealen 等人[11]提出利用基于像素的重新合成消除相邻块拼接的瑕疵，Kwatra 等人[12]提出图形剪切等。

基于样图的纹理合成技术大多数是基于马可夫随机场模型的，即通过邻域匹配从样图中找到合适的点或块，然后将其拷贝到目标纹理图中。但是在邻域匹配时仅仅考虑颜色信息会造成纹理结构的不匹配，使得合成结果不连续，特别是结构明显的纹理。针对这样的问题，也有研究者提出了改进的方法。Wu 等人[13]提出一种基于特征匹配和变形的纹理合成方法，该方法在块的选取和拼接时考虑了纹理的颜色和特征曲线的切线方向。Lee 等人[14]在块的搜索时提高了边界点和结构特征点的权重。Lefebvre 等人[15]在 2006 年提出使用降维技术将样图转换到外观空间，使得转换后样图的每个像素都包含纹理结构的信息，提高了合成的质量。沈等人[16]提出使用 Harris 算法提取纹理图像的结构信息，然后使用 Hausdorff 距离度量结构的相似度以引导块的搜索。张等人[17]提出一种结构自适应的纹理合成算法，其可以根据纹理的大小和对比度自适应地选择块匹配：当纹理的结构不明显时，只使用颜色作为匹配准则；当纹理的结构明显时，则采用基于结构相似性的匹配准则。Gui 等人[18]提出在选取匹配块时考虑像素间的关联性并参考样图构成的方向场，同时考虑颜色信息和纹理结构信息要符合人对纹理的感知，在一定程度上提高了纹理合成的质量。

3 基于 Chamfer 距离的纹理合成

本文采用图像拼接算法[1]的合成框架进行纹理合成。算法思想如图 1 所示，按照扫描线顺序逐行逐块合成目标纹理图案。对于当前待合成的块，计算它与样图中各个块的边缘区域的相似度，将满足阈值的相似块作为候选块。然后从这组候选块中随机选择一个块放到待合成的位置，并与已合成区域形成一定的边缘重叠区。在重叠区中找到一条最小误差缝合路径，沿此路径将新选择的块缝合到输出纹理中。如图 1 所示，基于块的纹理合成，输出部分的阴影代表了已经合成的部分；待合成的块边缘区域与已合成的部分有重叠。

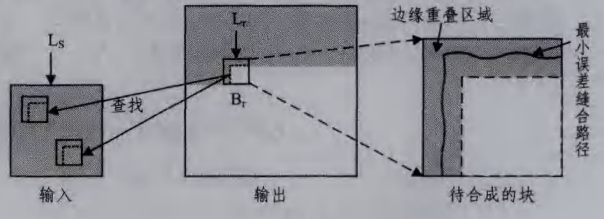

图 1

不同于以往仅考虑颜色信息的匹配块查找，本文在查找候选匹配块的同时考虑了纹理颜色和结构的相似度，从而提高了纹理合成的质量。与以往大多数纹理合成方法类似，颜色空间的相似度采用 RGB 颜色空间的欧氏距离进行计算，如式(1)所示：

$$d_e(B_s,B_r)=\sum_{\substack{p\in L_s,q\in L_r\\ p与q对应}}\sqrt{(R(p)-R(q))^2+(G(p)-G(q))^2+(B(p)-B(q))^2} \tag{1}$$

式中，B_s 表示样图中的块，B_r 表示待合成块，L_s 和 L_r 分别表示 B_s 和 B_r 中边缘区域，p 和 q 分别表示 L_s 和 L_r 中的像素(见图 1)，$R(\cdot)$、$G(\cdot)$、$B(\cdot)$分别表示颜色的 3 个通道。$d_e(B_s,B_r)$越小，B_s 与 B_r 在 RGB 颜色空间越相似。

如何衡量纹理块在结构空间的相似度是一个很难确定的问题。纹理的结构空间是一个复杂的高维空间，很难对纹理结构空间直接进行建模和表示。另一方面，Chamfer 距离适用于对简单的二值图或只有几种标签值的分割图进行距离计算。因此，为了方便地计算出纹理块的结构相似度，我们对纹理样图进行预处理，得到对应其轮廓或边缘的二值特征图(如图 2 所示，样图 S 生成对应的特征图，表现为二值图，标签值为 0 和 1)，把对纹理结构相似度的计算转换到二值特征图中进行，即在特征图 F_s 中计算对应纹理块的 Chamfer 距离。本文所述的 Chamfer 距离定义为边缘区域 L_r 中所有点与边缘区域 L_s 中标签值相同的点之间最小的 L_∞ 距离的总和[19]。如果在 L_s 中找不到相同标签值的点，这个点的距离则设为纹理块大小的两倍。根据这个定义，可以计算两个纹理块在对应的特征图上的 Chamfer 距离，图 3 显示了上下两组纹理块分别按照式(1)给出的颜色空间欧氏距离和 Chamfer 距离定义计算出来的距离值(即分别计算上下两组纹理块的欧氏距离和 Chamfer 距离，根据欧氏距离计算结果判定下层的两个区域更相似，根据 Chamfer 距离计算结果判定上层的两个区域更相似)。从计算结果可见，尽管上一组纹理比下一组纹理

视觉上更相近，但是其计算出来的欧氏距离更大，而 Chamfer 距离的计算结果却符合人眼的视觉观察。可见，对于形状或轮廓匹配等涉及纹理结构相似性的计算，Chamfer 距离比欧氏距离更加合适。

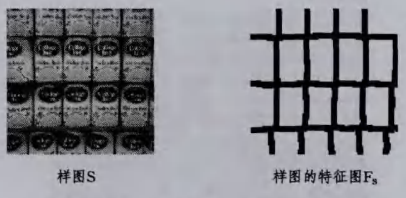

图 2　纹理样图 S 和其对应的特征图 F_s

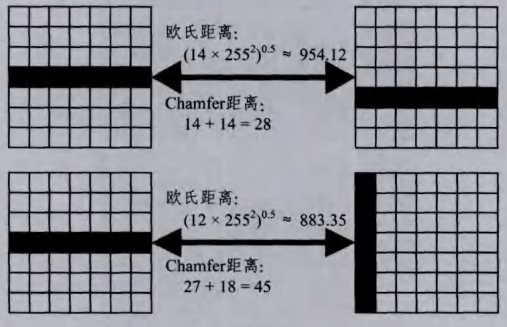

图 3

根据上述定义，Chamfer 距离是不对称的，即 L_r 到 L_s 的距离不一定等于 L_s 到 L_r 的距离。我们将 L_r 到 L_s 的 Chamfer 距离加上 L_s 到 L_r 的 Chamfer 距离得到对称的 Chamfer 距离。将样本和待合成块的边缘区域向量表示为：

$$L_s=\{p_{00},p_{01},\cdots,p_{xy},\cdots,p_{mn}\}$$

$$L_r=\{q_{00},q_{01},\cdots,q_{ij},\cdots,q_{mn}\}$$

其中，x,y 和 i,j 分别表示样图和待合成块中的点在边缘区域中的横纵坐标，p_{xy} 和 q_{ij} 表示对应点的标签值，那么 L_r 到 L_s 的单向 Chamfer 距离如式(2)所示，其中 $L_\infty d(q_{ij},p_{xy})$ 表示的是两点的 L_∞ 距离，如式(3)所示。B_s 与 B_r 的纹理结构的相似度由对称的 Chamfer 距离 $d_c(B_s,B_r)$ 表示，如式(4)所示。

$$d_c'(L_r,L_s)=\sum_{i=0}^{m}\sum_{j=0}^{n}\min_{0\leqslant x\leqslant m,0\leqslant y\leqslant n}\{d_{xy}\mid[d_{xy}=L_\infty d(q_{ij},p_{xy})\cap q_{ij}=p_{xy}]\cup[d_{xy}=2\times\max(m,n)\cap q_{ij}\neq p_{xy}]\} \tag{2}$$

$$L_\infty d(q_{ij},p_{xy})=\max(|i-x|,|j-y|) \tag{3}$$

$$d_c(B_s,B_r)=d_c'(L_r,L_s)+d_c'(L_s,L_r) \tag{4}$$

本文方法在查找相似块时，同时考虑了颜色的相似度和纹理结构的相似度，所以块的相似度函数公式为 $d(B_s,B_r)=d_e(B_s,B_r)+w\times d_c(B_s,B_r)$，其中 w 表示在计算总的相似度时 Chamfer 距离所占的权重。图 4 显示了样图 S(图 2 左)的合成结果，在得到合成目标纹理图 R 的同时会得到一张对应的特征合成图 F_r。

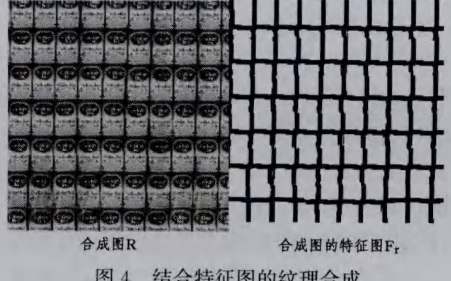

图 4　结合特征图的纹理合成

4　基于 GPU 加速的 Chamfer 距离的计算

随着计算机图形技术的发展，可编程图形处理器(GPU)提供了越来越强大的并行计算能力。NVIDIA 推出的通用并行计算架构 CUDA 使 GPU 能够解决复杂的计算问题。基于 CUDA 的 GPU 编程语言 CUDA C 使程序员能在 C 语言环境下完成对 GPU 并行处理架构的调用，减轻了使用 GPU 加速的难度。大部分在 GPU 上取得良好加速效果的程序都有一个或少数几个计算子程序因包含大量重复的可独立计算而需要花费大量的时间。将这些程序转化为 CUDA 内核函数，利用 GPU 来并行计算，是一种提高程序性能的好方法[21]。在本文的纹理合成的算法中，计算待合成块与样图中所有块的相似度是最为耗时的。但幸运的是，这个计算步骤可以并行计算，因此，为了提高效率，本文使用 GPU 来进行加速计算。

要想进行相似度的计算，首先需要收集当前待合成块的边缘区域 L_r 和样图中所有可能的块的边缘区域$\{L_s\mid L_s\in S\}$；然后计算边缘区域间的距离，包括欧氏距离和 Chamfer 距离。这里假设待合成块的边缘区域 L_r 由 n 个点组成 $L_r=\{q_1,q_2,\cdots,q_n\}$；样图中有 m 个可能的块的边缘区域$\{L_s^1,L_s^2,\cdots,L_s^m\}$，样图中的边缘区域同样由 n 个点组成 $L_s^i=\{p_1^i,p_2^i,\cdots,p_n^i\}$，$i=1,2,\cdots,m$，进行相似度计算得到 m 个距离值$\{d_1,d_2,\cdots,d_m\}$。相似度计算的串行伪代码如表 1 所列。

表 1　串行的相似度计算的伪代码

```
for i from 1 to m then
    d_i=0;
  for j from 1 to n then
        //计算欧氏距离
d_i+=
√((R(p_j^i)−R(q_j))^2+(G(p_j^i)−G(q_j))^2+(B(p_j^i)−B(q_j))^2)
//计算 Chamfer 距离
        min_L∞d1=区域大小的两倍
        min_L∞d2=区域大小的两倍
        //查找最小的 L∞距离,L∞d(·)表示计算 L∞距离
      for t from 1 to n then
          If p_j^i 的标签值等于 q_t 的标签值且
          L∞d(p_j^i,q_t)<min_L∞d1 then
              min_L∞d1=L∞d(p_j^i,q_t)
          If q_j 的标签值等于 p_t^i 的标签值且
          L∞d(q_j,p_t^i)<min_L∞d2 then
              min_L∞d2=L∞d(q_j,p_t^i)
      end
      d_i+=min_L∞d1×w
      d_i+=min_L∞d2×w
  end
end
```

将表 1 内的代码转化为 CUDA 的并行代码需要进行拆循环处理。拆循环是将 CPU 串行的代码转化为 CUDA 内核函数的一种有效方法，其前提是循环的前后步骤没有关联。拆循环就是为循环的每一步计算分配一条线程，这些线程在 GPU 中被并行处理。由表 1 中的伪代码可知相似度计算有三重循环，由于待合成块与样本中各个可能的块之间的相似度计算相互没有联系，因此最外层循环的前后步骤是没有关联的，可以进行拆循环。同样，计算块间两个点之间的欧氏距离和 L_∞ 距离也是相互独立的，所以中间层循环也可以拆。而拆了中间层循环也就意味着每个线程计算块与块之间两个点的距离，而欧氏距离是块之间所有对应点的距离之和，所以这里还需要一个累加计算，因此，我们在 GPU 中使用归约算

法进行累加计算。最内层的循环用于计算 Chamfer 距离，做一些修改后也是可以并行化的。

随着拆循环的深入，虽然线程的使用数量会随之增多，可以更充分地利用 GPU 的并行计算能力，但同时也会产生更多的中间值，需要进行显存的读写，而频繁读写显存会影响 GPU 的计算效率，同时也会造成显存溢出。所以，在本文的 CUDA 内核函数中，Chamfer 距离的计算相对比较复杂；而欧氏距离的计算比较简单，拆两层循环所带来的并行计算的优势反而比不上因此产生的读写的消耗，所以欧氏距离的计算只拆一层循环。本文 GPU 并行计算的伪代码如表 2 所列，使用 3 个 CUDA 内核函数实现。

表 2　并行的相似度计算的伪代码(使用 3 个 CUDA 内核函数实现)

```
计算 Chamfer 距离的内核函数：
for i from 1 to m then do in parallel
  for j from 1 to n then do in parallel
      min_L∞d1＝区域大小的两倍
      min_L∞d2＝区域大小的两倍
      //查找最小的 L∞距离，L∞d(·)表示计算 L∞距离
    for t from 1 to n then
          If p_j^i 的标签值等于 q_t 的标签值且
          L∞d(p_j^i, q_t)<min_L∞d1 then
            min_L∞d1＝L∞d(p_j^i, q_t)
          If q_j 的标签值等于 p_t^i 的标签值且 L∞d(q_j, p_t^i)<min_L∞d2
          then
            min_L∞d2＝L∞d(q_j, p_t^i)
    end
    d_ij＝(min_L∞d1＋min_L∞d2)×w
  end
end
累加内核函数：
for i from 1 to m then do in parallel
    //并行的归约累加
    for j from 1 to n then do in parallel
        d_i＋＝d_ij
    end
end
计算欧氏距离的内核函数：
for i from 1 to m then do in parallel
    for j from 1 to n then
      d_i＋＝√((R(p_j^i)−R(q_j))^2＋(G(p_j^i)−G(q_j))^2＋(B(p_j^i)−B(q_j))^2)
    end
end
```

5　实验结果

我们将本文方法的合成结果与图像缝合算法(Image quilting)的合成结果在合成质量和时间方面进行比较和说明。图 5 展示的是本文方法合成效果和图像缝合算法纹理合成效果的比较。图像缝合算法仅仅考虑了纹理的颜色信息，在合成结构明显的纹理图案时会出现结构不连续的问题，这一点可以从图 5 第 2 列的结果图(Image quilting 算法合成结果)中很容易地观察出来；而本文的方法在查找匹配块时不仅考虑到纹理的颜色信息，同时也考虑到纹理的结构特征，比较好地解决了纹理结构不连续的问题，其效果如图 5 第 3 列所示。

图 6 显示的是将本文合成的地形纹理映射到三维地形的绘制结果，可以看出在三维图形绘制中本文方法合成的二维纹理很好地保持了地形纹理的结构特征。从图 5 和图 6 的实验结果来看，本文使用 Chamfer 距离度量纹理结构的相似度能够有效提高纹理合成质量。

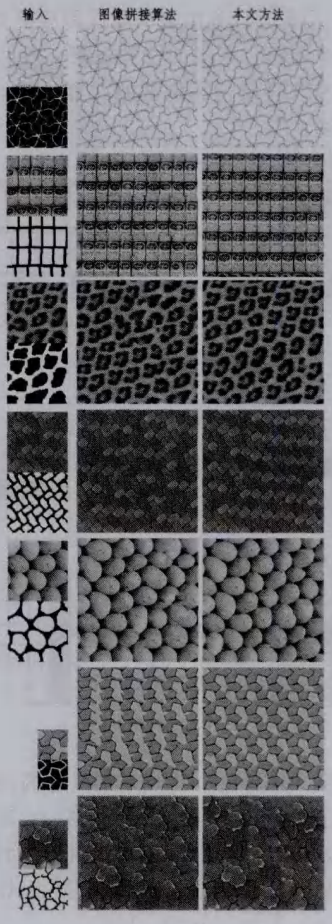

图 5　Image quilting 和本文方法合成效果的比较

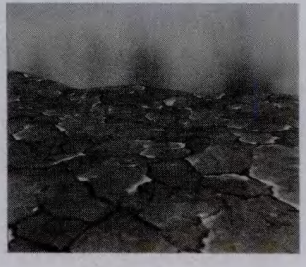

图 6　本文方法在合成地形纹理中的应用

使用 Chamfer 距离度量纹理结构的相似度大大增加了纹理合成的计算量，所以本文使用 GPU 并行计算来加速合成，将合成纹理所需要的时间控制在一个可接受的范围内。通过实验比较本文方法分别在 CPU 和 GPU 上合成纹理的效率。测试机器的 CPU 为 Intel(R) Xeon(R) E5506，GPU 为 Nvidia Quadro 600。

在块的大小和重叠边缘区域的宽度不变的情况下，分别使用 CPU 和 GPU 合成多种分辨率的纹理，并比较两种方式的合成效率。实验参数如下：样图大小为 128 * 128，块的大小是 20 * 20，边缘区域的宽度为 4，目标纹理合成图的大小分别为 200 * 200、400 * 400、600 * 600。测试结果的比较如图 7 所示，当目标纹理合成图的分辨率为 200 * 200 时，GPU 耗时 5.8s，CPU 耗时 233.1s，GPU 的计算效率快了 40 倍。随着目标纹理合成图的分辨率增大，GPU 加速效果越发明显。从曲线变化情况可见，CPU 合成时间随着合成图的分辨率提高而急剧增加，而 GPU 的合成时间基本与合成图的分辨率无关。

(下转第 312 页)

shop on Advanced Research and Technology in Industry Applications (WARTIA). 2014

[18] Vanne J, Viitanen M, Hamalainen T. Efficient Mode Decision Schemes for HEVC Inter Prediction[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2014, 24(9): 1579-1593

[19] Chen Y, Mukherjee D, Han J. Joint Inter-Intra Prediction Based on Mode-Variant and Edge-Directed Weighting Approaches in Video Coding [C] // International Conference on Acoustics Speech and Signal Processing. 2014

[20] Shen L, Liu Z, Zhang X, et al. Zhang. An Effective CU Size Decision Method for HEVC Encoders [J]. IEEE Transactions on Multimedia, 2013, 15(2): 465-470

[21] Bossen F, Bross B, Suhring K, et al. HEVC Complexity and Implementation Analysis [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2012, 22(12): 1685-1696

[22] Bjontegaard G. Calculation of average PSNR differences between RD-curves: VCEG-M33[S]. Austin, March 2001

(上接第 302 页)

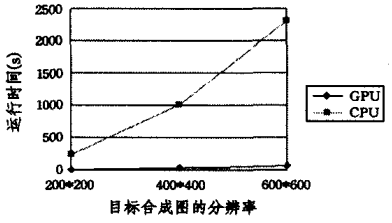

图 7 GPU 与 CPU 合成不同分辨率的纹理图的耗时比较

结束语 本文提出了一种改进的 Image quilting 方法，在样本中查找待合成的块时增加了纹理结构信息的引导，使得从样图中选取的块更为合理，改善了纹理合成的效果。但是 Chamfer 距离计算复杂，为了解决该方法带来的计算效率问题，本文使用了 GPU 并行加速，将合成时间控制在可接受的范围内，提高了纹理合成的质量，同时也保证了纹理合成效率。未来将进一步考虑如何在合成时根据样图纹理的特点更好地结合结构和颜色的信息，比如对于结构性较强的纹理，则在匹配时更多地考虑结构相似性；而对于结构性不那么强的纹理，则适当地添加结构信息。

参 考 文 献

[1] Wei L Y, Lefebvre S, Kwatra V, et al. State of the Art in Example-based Texture Synthesis [R]. Eurographics 2009 State of the Art Reports(STARs), 2009

[2] Efros A A, Freeman W T. Image quilting for texture synthesis and transfer[C] // Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. ACM, 2001: 341-346

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

[4] Wei L Y, Levoy M. Fast texture synthesis using tree-structured vector quantization[C] // Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., 2000: 479-488

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

[6] Zelinka S, Garland M. Towards real-time texture synthesis with the jump map[C] // Proceedings of the 13th Eurographics Workshop on Rendering. Eurographics Association, 2002: 99-104

[7] Lefebvre S, Hoppe H. Parallel controllable texture synthesis [J]. ACM Transactions on Graphics (TOG), 2005, 24(3): 777-786

[8] Guo B, Shum H, Xu Y Q. Chaos mosaic: Fast and memory efficient texture synthesis[R]. Microsoft Research Paper MSR-TR-2000-32, 2000

[9] Zhu Wen-hao, Wei Bao-gang. The Technology of Sampled-Based Texture Synthesis[J]. Journal of Image and Graphics, 2008, 13(11): 2063-2069(in Chinese)
朱文浩，魏宝刚. 基于样本的纹理合成技术综述[J]. 中国图象图形学报, 2008, 13(11): 2063-2069

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

[11] Nealen A, Alexa M. Hybrid texture synthesis[C] // Proceedings of the 14th Eurographics Workshop on Rendering. Eurographics Association, 2003: 97-105

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

[13] Wu Q, Yu Y. Feature matching and deformation for texture synthesis[J]. ACM Transactions on Graphics (TOG), 2004, 23(3): 364-367

[14] Lee T Y, Yan C R. Feature-based texture synthesis[M] // Computational Science and Its Applications-ICCSA 2005. Springer Berlin Heidelberg, 2005: 1043-1049

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

[16] Shen Jian-wei, Wang Shuo-zhong. Structural Matching in Texture Synthesis Based on Image Quilting[J]. Journal of Shanghai University(Natural Science Edition), 2010, 16(1)(in Chinese)
沈剑伟，王朔中. 拼接法纹理合成中的结构特征匹配[J]. 上海大学学报 (自然科学版), 2010, 16(1)

[17] Zhang Jun, Zhu Wei, Huang Wei-qiang. Novel Structure Adaptive Algorithem for Texture Synthesis [J]. Journal of Chinese Computer Systems, 2011, 32(2): 351-355(in Chinese)
张军，朱为，黄伟强. 一种新的结构自适应纹理合成算法[J]. 小型微型计算机系统, 2011, 32(2): 351-355

[18] Gui Y, Chen M, Xie Z, et al. Texture Synthesis based on Feature Description[J]. Journal of Advanced Mechanical Design, Systems, and Manufacturing, 2012, 6(3): 376-388

[19] Barrow H G, Tenenbaum J M, Bolles R C, et al. Parametric correspondence and Chamfer matching: two new techniques for image matching[C] // Proceedings of the 5th international joint conference on Artificial intelligence-Volume 2. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1977: 659-663

[20] Borgefors G. Hierarchical Chamfer matching: a parametric edge matching algorithm[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1988, 10(6): 849-865

[21] Farber R. CUDA application design and development[M]. Morgan Kaufmann, 2011