Journal of System Simulation

Volume 27 | Issue 6 Article 19

1-15-2021

Research of Liver Solid Texture Synthesis and Mapping Method with CUDA Acceleration

Guodong Chen
Institute of Physics and Information Engineering, Fuzbou Univrsity, Fuzhou 350002, China;

Hanxin He
Institute of Physics and Information Engineering, Fuzbou Univrsity, Fuzhou 350002, China;

Research of Liver Solid Texture Synthesis and Mapping Method with CUDA Acceleration

Abstract

Abstract: A liver solid texture synthesis and mapping method based on Computer Unified Device Architecture acceleration (CUDA) was proposed to solve the problem of the overlong time consuming within the period of synthesizing liver solid texture in traditional way. *The relevance in traditional serial texture synthesis was elim inated in the new method. The work of selecting and distributing blocks of space synthesis of the liver solid texture was processed by using the parallel processing of multiple threads based on CUDA. Both the procedures of tinting the surface grid nodes of liver model and internal point set traversal in mapping calculation were parallelized and these work were done by GPU.* The experimental results show that the new method possesses a higher resultant velocity, simultaneously guaranteeing the realistic of liver solid texture.

Keywords

solid texture, synthesis, mapping, CUDA, parallel computing

第 27 卷第 6 期 系统仿真学报© Vol. 27 No. 6
2015 年 6 月 Journal of System Simulation Jun., 2015

CUDA 加速的肝脏体纹理合成与映射方法研究

陈国栋，何汉鑫
（福州大学物理与信息工程学院，福州 350002）

摘要：提出一种基于计算机统一设备架构加速(CUDA)的肝脏体纹理合成与映射方法，用于解决传统方法中合成肝脏体纹理耗时过长的问题。*新方法消除了传统串行纹理合成中的关联性，采用基于 CUDA 构架的多线程并行处理来进行肝脏体纹理空间合成中的选块和布块工作，将映射计算中肝脏体模型表面网格节点着色、三角面片内部点集遍历的方法并行化，转移到 GPU 上进行计算。*实验结果表明，该方法在保证肝脏体纹理真实感的同时，具有更高的合成速度。

关键词：体纹理；合成；映射；CUDA；并行计算
中图分类号：TP317.4 文献标识码：A 文章编号：1004-731X (2015) 06-1280-08
DOI:10.16182/j.cnki.joss.2015.06.019

Research of Liver Solid Texture Synthesis and Mapping Method with CUDA Acceleration

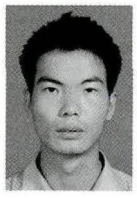
Chen Guodong, He Hanxin
(Institute of Physics and Information Engineering, Fuzhou University, Fuzhou 350002, China)

Abstract: A liver solid texture synthesis and mapping method based on Computer Unified Device Architecture acceleration (CUDA) was proposed to solve the problem of the overlong time-consuming within the period of synthesizing liver solid texture in traditional way. *The relevance in traditional serial texture synthesis was eliminated in the new method. The work of selecting and distributing blocks of space synthesis of the liver solid texture was processed by using the parallel processing of multiple threads based on CUDA. Both the procedures of tinting the surface grid nodes of liver model and internal point set traversal in mapping calculation were parallelized and these work were done by GPU.* The experimental results show that the new method possesses a higher resultant velocity, simultaneously guaranteeing the realistic of liver solid texture.

Keywords: solid texture; synthesis; mapping; CUDA; parallel computing

引言

真实感和实时性是当前虚拟手术领域广泛讨论的热点问题。在虚拟肝脏手术的研究中，由于肝脏体包含了大量复杂的纹理信息，因此很难同时兼顾合成的虚拟肝脏模型的真实感与合成速度。国内外不少学者对纹理合成的方法进行过深入的研究，提出了许多相对高效的合成方法，但大多都是通过改进算法，减少纹理合成过程中的计算量来实现加速，但这种方法受到了 CPU 等硬件设备处理能力的限制，存在很大的局限性。近年来，由于制造工艺技术的制约，CPU 时钟频率的提升渐渐碰到了壁垒，相比较之下 GPU 因其硬件架构的不同，性能仍能够遵循摩尔定律不断提升，在一些高度并行化的大规模数据场的计算中，GPU 开始崭露头角。在此基础上，NVIDIA 公司于 2007 年发布了 CUDA(Computer Unified Device Architecture:计算机统一设备架构)，使 GPU 能够解决复杂的计算问



收稿日期：2014-05-14 修回日期：2014-07-13；
基金项目：福建省科技计划重点项目(2011H0027)；
福建省自然科学基金(2013J05090)；
作者简介：陈国栋(1979-)，男，福建永春，博士，副研究员，研究方向为虚拟现实；何汉鑫(1990-)，男，福建漳州，硕士生，研究方向为图像处理与通信。

题，有力地推动了计算机软硬件行业的发展。本文的研究基于 NVIDIA 提出的 CUDA 架构，以具有高度并行计算能力的 GPU 为硬件基础，对体纹理合成的方法进行改进，使其与 GPU 硬件特性相结合，以此来提高体纹理的合成速度。

在虚拟肝脏手术的研究中，肝脏体纹理空间的合成和肝脏体模型的纹理映射是合成肝脏体纹理过程中密不可分的 2 个部分，不同的方法所合成的肝脏体纹理在真实感与合成速度上往往有着很大的差别[1-2]。文献[3]提出了一种基于优化的二维纹理来合成三维体纹理的方法，并采用直方图匹配的方式，使生成的体纹理能够包含较丰富的全局特征，该方法能够在模型表面映射出质量较高的纹理，但在内部纹理细节的显示方面存在一定不足。文献[4]提出一种基于空间各向异性的体纹理合成方法，根据用户在四面体网格模型上定义的张量场粘贴体纹理块，该方法能够较好地模拟树木、水果、蔬菜等的表面及内部纹理细节，但在合成高精度模型时存在速度偏慢的问题，无法满足虚拟手术对于交互性的要求。文献[5]提出一种基于二维纹理贴图的体纹理合成方法，该方法对候选块进行一系列的预处理选择工作，以减少搜索空间，弱化相邻纹理匹配的关联性，并利用 GPU 的并行特性进行计算，优化合成速度。文献[5]所述方法在一些纹理细节较少且周期性特征较强的体纹理合成中具有良好的合成效果及合成速度，但由于肝脏体具有丰富的纹理细节，且无明显的结构性特征，因此基于二维贴图的体纹理合成无法合成具有精细纹理的肝脏体模型。

文献[2]提出了一种基于复用计算的体纹理合成方法，它将文献[6-7]中复用合成二维纹理的方法拓展到了三维空间，该方法首先根据样本体纹理合成一个初始体纹理区域，以合成的初始体纹理区域为新的样本体纹理获取更大的体纹理空间，进而迭代地进行目标体纹理空间余下区域的合成，采用复用计算的肝脏体纹理合成速度要明显高于非复用计算，这也是目前虚拟手术中合成肝脏体纹理空间较快的方法。但随着肝脏体模型表面三角面片数目的增大，将肝脏体纹理从纹理空间映射到肝脏体模型的过程的耗时仍是尚未解决的一大难题。

本文针对文献[2]中仍然存在的问题，引入并行计算的方法，将体纹理空间的合成过程中选块、布块操作与肝脏体纹理的映射过程中模型表面着色计算等部分搬移到 CUDA 上执行，利用 GPU 多线程与高浮点运算能力的特性，对肝脏体纹理合成的过程进行加速。实验结果表明，在对表面三角面片数目较多的肝脏体模型进行纹理合成时，新方法在保证体纹理真实感的前提下，依然能够拥有很高的合成速度。

1 准备工作

1.1 样本体纹理生成

我们采用文献[2]中的方法，在 VHP 数据集提供的人体肝脏连续横断面图像中选取 128 张感兴趣区域(ROI)大小为 128×128 的肝脏横断面图，通过顺序累加的方法来生成样本体纹理，得到的肝脏体纹理样本如图 1 所示。

图 1 肝脏体纹理样本

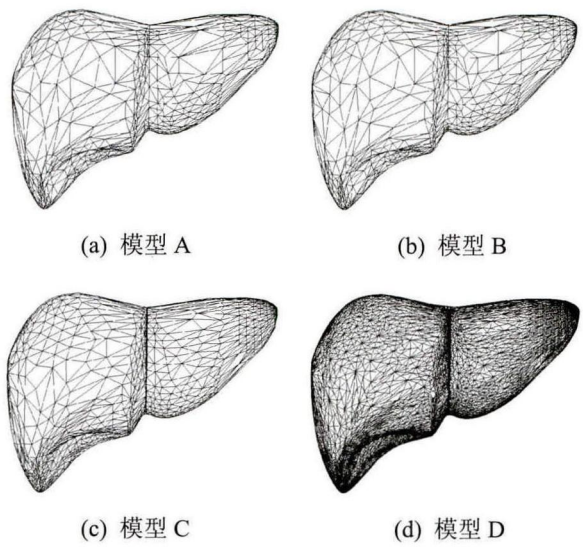
1.2 肝脏模型构建

为了对比研究新方法下表面三角面片数目不同的肝脏体模型的合成速度和映射效果，我们构建了 4 组大小相同，但面数不同的肝脏体表面网格模型。由于模型表面肝脏纹理的映射及后续工作中内部纹理结构显示的需要，我们首先对模型进行四面体化，使模型内部由一定数量的四面体无缝拼接而成，得到如图 2 所示的 4 组肝脏体模型。由于四面体的 4 个面均为三角形，因此肝脏

DOI: 10.16182/j.cnki.joss.2015.06.019

第 27 卷第 6 期 系统仿真学报 Vol. 27 No. 6
2015 年 6 月 Journal of System Simulation Jun., 2015

体模型的表面由三角面片覆盖而成。可以看出，随着肝脏体模型表面三角面片数目的增加，其表面的凹凸细节越来越丰富，边缘过渡也更加平滑，从而在合成过程中能够获得更具真实感的虚拟肝脏体模型。表 1 为 4 组肝脏体模型的基本信息。



(a) 模型 A (b) 模型 B

(c) 模型 C (d) 模型 D

图 2 肝脏体表面网格模型

表 1 肝脏体模型信息

	模型 A	模型 B	模型 C	模型 D
顶点数	1 323	1 537	2 687	14 663
面数	2 642	3 070	5 370	18 676
四面体数	4 830	5 578	9 853	61 014

2 体纹理空间合成

为了将体纹理样本映射到肝脏体模型上形成纹理细节丰富且真实感强的肝脏体纹理，需要建立精细的肝脏体纹理空间。文献[2]提出 1 种基于复用计算的方法来合成体纹理空间，基本方法是：首先计算肝脏体模型在空间直角坐标系中 XYZ 各方向上的最小值点，以该点为起始点合成 1 个初始区域，然后以合成的区域为新的样本体纹理迭代地合成更大的纹理区域直至包含整个肝脏体模型空间。在该方法中，后续的体纹理合成工作重复使用了上一次复用合成的结果，因此能够获得较高的合成效率。但对于 GPU 架构来说，空间区域内所有的布块工作都必须独立地进行，才能够高效地利用 GPU 的硬件性能。有别于文献[2]，新方法采用多线程进行随机布块，对肝脏体纹理空间进行并行填充。由于肝脏体纹理的分布不具有明显的结构性与周期性特征，因此只要纹理块大小选取合适[8]，就能够保证合成的体纹理空间包含细节丰富且真实感强的肝脏体纹理。具体方法如下：

(1) 确定体纹理块大小：首先，需要从肝脏体纹理样本中选取大小合适的体纹理块，以便进行空间区域的布块。考虑到后续映射工作中节点着色时索引方便的需要，本文采用固定大小的体纹理块进行填充。为了能够既方便地提取纹理块，又保证纹理细节足够丰富，我们将用于填充的纹理块尺寸确定为原样本纹理块的1/8，即为64×64×64。

(2) 确定体纹理空间大小：遍历肝脏体模型表面网格的节点坐标值，计算XYZ各方向上最大坐标值 x_{max}，y_{max}，z_{max} 及最小坐标值 x_{min}，y_{min}，z_{min}，分别相减得到方向最大差值 $x_{max}-x_{min}$，$y_{max}-y_{min}$，$z_{max}-z_{min}$，将各方向最大差值除以64，向上取整得到的结果即为各方向上填充的纹理块数目。通过获得各方向上的纹理块排列数目，可以直观地建立GPU多线程模型。这里不妨假设XYZ 3个方向上排列的纹理块数目分别为a, b, c，则在进行排列布块工作时，我们需要同时启动 $a\times b\times c$ 个thread同时且独立地进行纹理块的提取和排列工作。

(3) 纹理块的提取：首先，以图1所示的样本体纹理块中红蓝绿3条线为XYZ坐标轴，交点为原点建立空间直角坐标系。在样本体纹理块空间中随机地取一点 $A(x_a, y_a, z_a)$，其中 $0 \leqslant x_a, y_a, z_a \leqslant 64$，如图3所示，以A点为起始点，XYZ 3个方向上的空间长度为64，提取出1个新的纹理块。新方法不仅可以从样本纹理空间中随机地提取出纹理块，也能够保证合成的体纹理空间能够充分地包含样本纹理块中的信息。由于肝脏体纹理空间可以容纳 $a\times b\times c$ 个体纹理块，纹理块的提取与排列工作都是随机且独立进行的，因此采用并行处理的GPU相对于传统CPU架构来说有着明

显的速度优势。

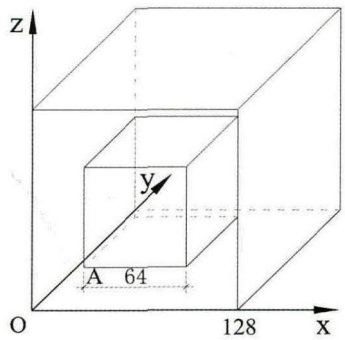


图 3 纹理块提取示意图

(4) 接缝处理：为了避免对体纹理空间进行填充后各纹理块的拼接出现不自然的情况，需要对纹理块的接缝进行缝合处理。缝合采用泊松图像编辑的方法[9-10]，首先利用图像的梯度场对待拼合的区域进行引导插值，这里梯度可以写作：

$$\nabla I_i = \frac{I_{i+1,j} - I_{i,j}}{(i+1) - i} = I_{i+1,j} - I_{i,j} \quad (1)$$

$$\nabla I_j = \frac{I_{i,j+1} - I_{i,j}}{(j+1) - j} = I_{i,j+1} - I_{i,j} \quad (2)$$

其中，i, j分别为纹理图像平面I的行坐标和列坐标。然后建立泊松方程(式(3))，由于体纹理图像是RGB颜色模型，因此3个颜色通道之间是相互独立的，只需对每一个颜色通道分别求解泊松方程，将解向量赋值给拼合图像边界即可。

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \varphi(x, y) \quad (3)$$

对于离散的数字图像来说，空间中每一层的纹理图像拼合工作都可以是独立进行的，如图4所示，我们以阴影方块作为待处理的区域，白色方块为目标区域，直接对体纹理空间中每一层图像进行拼合处理。在CUDA中，为了加快拼合速度，我们取 $x_{max} - x_{min}$，$y_{max} - y_{min}$，$z_{max} - z_{min}$ 中最大值L_{max}为纹理层数，启动L_{max}个thread。建立线程索引为threadIdx.x，使每个thread处理一层纹理图像，然后将拼合计算后的结果重新写入相应的纹理块中。由于拼合计算在CUDA中实现的线程结构较为简单，限于篇幅，此处不作过多阐述。图5为经过泊松编辑后拼接而成的肝脏体纹理空间。

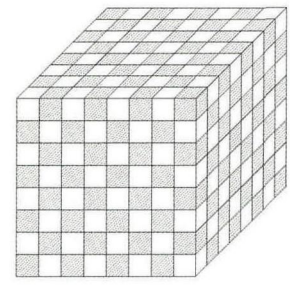


图 4 泊松图像编辑区域

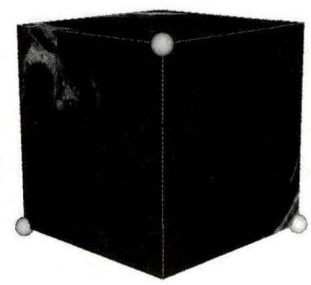


图 5 8×8×8 排列的泊松拼接效果图

3 体纹理映射

在完成体纹理空间的合成工作后，需要将纹理空间中的像素点映射到相应位置的肝脏体模型上。在不进行切割操作的情况下，只有模型表面的纹理是可见的，因此我们只对肝脏体模型表面进行纹理映射计算。在体纹理映射计算中，由于四面体化后的肝脏体模型的表面是由三角面片组成，为了获得三角面片内部点集的纹理值，需要根据网格节点的空间坐标值计算出内部点集的空间坐标，再将各点的空间坐标转换为所在纹理块的纹理坐标，最后进行三角面片内部点集的着色。由于遍历计算的步进值参数无法确定，内部点集的遍历无法覆盖网格节点，因此我们首先需要对肝脏体模型表面网格的节点进行着色计算。映射计算的具体工作如下：

(1) 表面网格节点着色：首先计算体纹理空间中排列的各体纹理块中心点的空间坐标值，计算出该体纹理块所包含的空间区域范围，然后根据肝脏体模型表面网格节点的空间位置，判断其所在的体纹理块，再通过该节点的空间坐标及纹理块中心点

DOI: 10.16182/j.cnki.joss.2015.06.019

坐标，计算出其在相应纹理块中的纹理坐标，最后根据纹理坐标获得相应位置的纹理值，将该纹理值赋给相应的网格节点，即可完成节点的着色工作。在肝脏体模型表面网格节点的着色计算中，节点的坐标需要经过一系列的计算才能获得该点在相应位置的纹理值，这样单个节点的实际着色时间为 4 ms 左右，在顶点数目较多的肝脏体模型中，节点的着色工作需要耗费很长的时间。由于各网格节点的着色工作实际是相互独立的，因此利用 GPU 的并行特性进行着色计算是一个更好的选择。

本文在CUDA上实现了节点着色的并行计算，步骤如下：

(a) 分配显存空间：在 GPU 上分配显存空间，用于存放 float 3 类型的网格节点空间坐标值，int 3 类型的纹理块中心点坐标值，byte 类型的纹理块的纹理数据，int 3 类型的节点纹理值，以及 int 类型的顶点数和 int 类型的纹理块个数。

(b) 主机数据传输至设备：调用 cudaMemcpy() 函数，利用 cudaMemcpyHostToDevice 将主机内存中节点的坐标值、纹理块中心点坐标值、纹理块的纹理数据及顶点数和纹理块个数拷贝到设备(Device)中。

(c) Kernel 函数参数设定：我们假设肝脏体模型有 N 个顶点，由于体纹理空间中共有 $a\times b\times c$ 个纹理块，需要对每个节点遍历 $a\times b\times c$ 个纹理块来判断其属于哪个纹理块，因此需要启动 $N\times a\times b\times c$ 个 thread，每 $a\times b\times c$ 个 thread 处理 1 个节点的着色计算。由于单个 block 最多可以同时启动 1 024 个 thread，在 $N\times a\times b\times c$ 远大于 1 024 的情况下，我们通过定义 1 个 dim 3 类型的二维线程块 threadsPerBlock(1 024,1)以及 1 个 dim 3 类型的二维网格 blocksPerGrid((N+1 023)/1 024,48)来实现线程的覆盖，即在 CUDA 中，启动((N+1 023)/1 024)×48 个 block，每个 block 中启动 1 024 个 thread，其中 x 方向上线程索引为 blockIdx.x × blockDim.x + threadIdx.x，y 方向上的线程索引为 blockIdx.y。

(d) Kernel 函数：在 Kernel 函数中，我们通过计算网格节点在纹理块中的偏移量来获得相应位置的纹理值，具体做法为，首先根据各纹理块的中心点坐标在 XYZ 正方向上加上体纹理尺寸的一半，负方向上减去体纹理尺寸的一半来确定纹理块的空间大小，然后通过 x 方向线程取网格节点的坐标在 y 方向上遍历，判断其属于哪个体纹理块，再计算当前节点与所在体纹理块中心点在 XYZ 3 个方向上的距离 d，然后用体纹理尺寸的一半减去 d 即可算出当前节点在相应纹理块中的纹理坐标，最后根据新的坐标获取该点的纹理颜色值。在着色计算完成后，存储所有网格节点的 RGB 颜色值。

(e) 计算结果传送回主机：调用 cudaMemcpy() 函数，利用 cudaMemcpyDeviceToHost 将结果拷贝回主机(Host)，并释放显存空间。

(2) 三角面片内部点集计算：为了将体纹理空间中的纹理信息映射到肝脏体模型上，形成有真实感的纹理，还需要计算出三角面片内部点集的空间坐标，再转换为所在纹理块的纹理坐标，最后进行纹理着色。

在内部点遍历时，不妨假设三角面片的三个顶点分别为 $A(x_0,y_0,z_0)$，$B(x_1,y_1,z_1)$，$C(x_2,y_2,z_2)$，取三角形 ABC 内部任意一点 M，如图 6 所示。根据数学向量的性质可知：

$$\overrightarrow{AM}=a\times\overrightarrow{AB}+b\times\overrightarrow{AC} \tag{4}$$

其中 $a=AB'/AB$，$b=AC'/AC$，且 a,b 满足条件 $0\leqslant a\leqslant 1$，$0\leqslant b\leqslant 1$，$a+b\leqslant 1$，因此只要 a,b 的步进值取值足够小，就可以遍历三角形中所有的点，从而计算出这些点的空间坐标值。虽然步进值越小能够映射出越精细的表面纹理，但超过一定的临界值后，效果提升的显著性将大幅度降低且将造成大量的冗余计算。经过比较测试，我们取步进值 a,b 分别为 AB，AC 长度值的倒数，即：

$$\begin{aligned}L_a&=1/\sqrt{(x_1-x_0)^2+(y_1-y_0)^2+(z_1-z_0)^2}\\L_b&=1/\sqrt{(x_2-x_0)^2+(y_2-y_0)^2+(z_2-z_0)^2}\end{aligned} \tag{5}$$

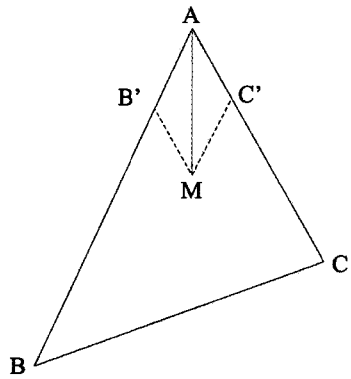


图 6 三角面片内部点计算

在三角面片内部点集的着色计算中，我们通过遍历肝脏体模型中所有四面体的 4 个表面，判断每个表面是否存在邻接面来确定表面三角面片的数目。由于表面判断不需要进行结构复杂的计算，因此我们将统计表面三角面片数目的工作放在 CPU 中进行。

在利用 CUDA 架构进行计算时，由于肝脏体模型表面三角面片的边长各不相同，每个三角面片内部点的数目也不一样，为了简化线程索引的结构，我们让每个 thread 完成 1 个三角面片的所有计算，虽然计算量较大，但就单个三角形内部点集的遍历来说，所花费的时间较短，因此该方法具有较强的可行性。同节点着色计算一样，首先需要在 GPU 中分配显存空间，用于存放 float 3 类型的网格节点坐标值，byte 类型的纹理块的纹理数据，int 3 类型的纹理块中心点坐标值，int 3 类型的三角面片内部点坐标值，int 3 类型的三角面片内部点纹理值以及 int 类型的三角面片数。然后调用 cudaMemcpy 函数，利用 cudaMemcpyHostToDevice 将主机内存中节点坐标值，纹理块的纹理数据，纹理块中心点坐标值和三角面片数拷贝到设备(Device)中。启动能够覆盖三角面片数目的 thread 以便进行着色的计算。在 Kernel 函数中，取 a, b 的初始值为 0，首先在 $a+b$ 不大于 1 的情况下，以步进值为 L_b 对 b 进行累加，再取 a 的值为 L_a，在 $a+b$ 的值不大于 1 的情况下，重新以 $b=0$ 为初始值对 b 进行循环累加，以此类推，直到 a 的值无限接近 1 为止，即可完成对三角面片内部点集的遍历。在获得三角面片内部点的空间坐标后，我们依照节点着色中的方法，将各点的空间坐标转换为所在纹理块的纹理坐标，再根据纹理坐标从相应体纹理块中提取出纹理颜色值，最后将各三角面片中点集的坐标值和纹理值拷贝回主机(Host)，完成三角形内部点集的着色计算。

三角面片内部点集并行计算着色流程如图 7。

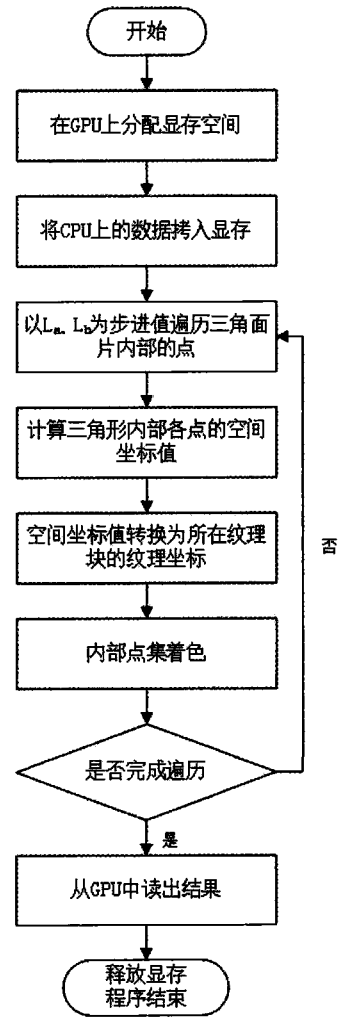


图 7 三角面片内部点集着色并行计算流程

4 实验结果与分析

4.1 合成速度分析

我们用 Visual Studio 2012 和 CUDA 5.5 实现了本文的新方法，并在 1 台 Dell XPS L502X 的 PC 上进行了实验，该机配置了 Intel® Core™ i7-2670QM 2.2 GHz 的 CPU, 8 GB 内存，NVIDIA

DOI: 10.16182/j.cnki.joss.2015.06.019

GeForce GT 540 M 的显卡及 2 GB 显存。由于文献[2]的实现使用的机器配置与本文不同，为了不失公平性，我们在本机上对文献[2]中方法的着色时间进行了重新记录。表 2 列出了新方法下顶点数不同的肝脏体模型表面网格节点着色的平均时间与文献[2]中方法着色时间的对比。表 3 列出了新方法下三角面片数不同的肝脏体模型三角面片内部点集的着色时间与文献[2]中方法着色时间的对比。由于 4 组肝脏模型的空间大小相同，在体纹理空间合成的时间上并无差别，且耗时占总时间的比例较小，因此这里不做详述。

表 2 新方法与文献[2]方法节点着色时间对比

	模型 A	模型 B	模型 C	模型 D
顶点数	1 323	1 537	2 687	14 663
新方法着色时间	0.236 s	0.254 s	0.285 s	0.302 s
文献[2]着色时间	5.169 s	5.957 s	10.677 s	56.659 s
加速比	21.90	23.45	37.46	187.61

表 3 新方法与文献[2]方法三角面片内部点着色时间对比

	模型 A	模型 B	模型 C	模型 D
三角面片数	2 642	3 070	5 370	18 676
新方法着色时间	1.33 s	1.362 s	1.425 s	2.687 s
文献[2]着色时间	36.286 s	37.183 s	42.22 s	115.37 s
加速比	27.28	27.30	29.63	42.94

另外，在对三角面片内部点集进行着色的工作中，由于模型表面三角面片内部点集的数据量过于庞大，显存中无法分配可以同时容纳三角面片内部各点空间坐标值与纹理值的空间，因此，我们将计算各点空间坐标值与着色的工作拆分在 2 个 Kernel 函数中进行，首先在 setInsidePointKernel 函数中进行三角面片内部各点坐标的计算，在各点坐标值从显存拷贝回内存后，释放对应的显存空间，再为内部点集分配存储纹理值的空间，然后在 setInsideColorKernel 函数中重新遍历三角面片内部的点，同时通过添加一维索引的方法来使各点与其纹理值一一对应，从而完成内部点集的着色。由于实验设备硬件条件的限制，实验时不得已做出一些妥协，因此产生了一些重复计算造成的延迟，影响了实验的结果。在显存容量更高的硬件设备中，我们可以得到比表 2 更短的三角面片内部点集的着色时间。

4.2 合成效果分析

图 8 所示为采用新方法合成肝脏模型体纹理的结果。通过表面三角面片数目不同的肝脏模型合成效果的对比，可以看出在图 8(a)所示的模型 A 中，由于表面三角面片数目较少，三角形的面积相对更大，存在一部分三角形的 3 个顶点不在同 1 个体纹理块内的情况，造成了在映射计算中表面纹理的过渡不够连续。随着表面三角形数目的增多，面积随之变小，这种情况得到了很大的改善，如图 8(b)。当肝脏体模型表面的三角面片数目达到一定数量时，纹理不连续的情况就得到了很好的解决，如图 8(c)。当肝脏体表面的三角面片数目达到 18 676 时，已经能够合成出具有较强真实感的肝脏体纹理，可以看出该模型表面纹理细节丰富，过渡自然，如图 8(d)。

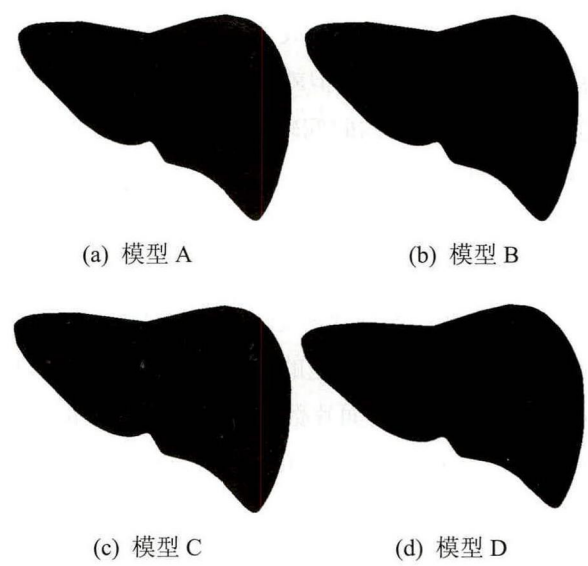


图 8 肝脏体模型体纹理合成效果

5 结论

本文在 CUDA 架构的基础上，研究了基于并行处理的肝脏体纹理合成与映射方法，提出了利用多线程处理的方法来进行肝脏体纹理空间合成

中选块、布块工作，将映射计算中表面网格节点着色、三角面片内部点集遍历与着色等工作并行化，同时转移到 GPU 上进行计算，使体纹理合成的速度能够满足虚拟手术的交互性的需求，且具有较强的真实感，为后续切割显示的工作奠定了基础。开发了基于 OpenGL 的三维显示系统，该系统能对肝脏体模型进行旋转、平移、缩放，利用 CUDA 的并行特性，我们对显示系统的性能进行了优化，提高的系统的响应速度。但仍然存在一些问题，如在三角面片数目过高时将导致显存不足以分配而使系统性能降低，我们相信在具有高配置的医疗设备中进行新方法的合成时，这一问题将迎刃而解；另外，由于尚未进行肝脏体模型切割计算的研究，因此无法看到肝脏体模型内部的纹理细节，在后续工作中，我们将继续对研究肝脏体的切割操作及内部纹理显示的方法进行研究，并在 CUDA 上进行计算，使其能够在虚拟手术中实现交互速度的切割操作。

本文实现了肝脏体纹理的快速合成，对于虚拟肝脏手术的研究有着重要的促进作用。新方法同样适用于纹理结构性较弱且周期性特征不明显的体纹理的合成，因此在合成某些特定的体纹理时，新方法也可以获得非常高的合成速度和真实感，对体纹理合成领域的发展有一定的指导意义。

参考文献:

[1] 翟朝亮, 陈国栋, 王娜. 基于体纹理的肝脏可视化仿真方法研究 [J]. 电视技术, 2012, 36(17): 169-172.

[2] 潘翔. 基于复用计算的肝脏软组织体纹理合成方法研究 [D]. 福州: 福州大学, 2014.

[3] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, *et al*. Solid texture synthesis from 2D exemplars [J]. ACM Transactions on Graphics (S0730-0301), 2007, 26(3): 21-29.

[4] Kenshi Takayama, Makoto Okabe, Takashi Ijiri, *et al*. Lapped Solid Textures: Filling a Model with Anisotropic Textures [J]. ACM Transactions on Graphics (S0730-0301), 2008, 27(3): 531-539.

[5] Yue Dong, Sylvain Lefebvre, Xin Tong, *et al*. Lazy Solid Texture Synthesis [J]. Computer Graphics Forum, (S0167-7055), 2008, 27(4): 1165-1174.

[6] 陈昕, 王文成. 大尺寸纹理的实时合成 [J]. 软件学报, 2009, 20(S1): 193-201.

[7] 陈昕, 王文成. 基于复用计算的大纹理实时合成 [J]. 计算机学报, 2010, 33(4): 768-775.

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

[9] Perez P, Gangnet M, Blake A. Poisson image editing [J]. ACM Transaction on Graphics (S0730-0301), 2003, 22(3): 313-318.

[10] 张建桥, 王长元. 基于泊松方程的数字图像无缝拼合 [J]. 现代电子技术, 2010 (17): 139-141.

(上接第 1279 页)

[8] Yinghao Feng, Riwei Wang, Fan Wang, *et al*. Real-time Rendering Techniques of Lunar Massive Terrain Data Based on Wavelet [J]. Journal of Computational Information Systems (S1553-9105), 2012, 8(14): 6079-6085.

[9] Albert Cervin. Adaptive Hardware-accelerated Terrain Tessellation [M]. SWE: Linköping University, 2012.

[10] David Eberly. 3D游戏引擎设计: 实时计算机图形学的应用方法 [M]. 徐明亮, 译. 北京: 清华大学出版社, 2013.

[11] Rui Wang, Xuelei Qian. OpenSceneGraph 3 Cookbook [M]. UK: Packt Publishing, 2012.

[12] David Wolff. OpenGL 4 Shading Language Cookbook [M]. UK: Packt Publishing, 2013.

[13] Sellers G, Obert J. Rendering massive virtual worlds [C]// ACM SIGGRAPH 2013 Courses. Anaheim, California, USA: ACM, 2013: 1-88,

[14] Olsson O, Billeter M, *et al*. Clustered deferred and forward shading [C]// HPG '12: Proc. of the Conf. on High Performance Graphics, Paris, France, 2012. 2012: 87-96.

[15] Karras T, Aila T. Fast parallel construction of high-quality bounding volume hierarchies [C]// HPG '13: Proc. of the Conf. on High Performance Graphics, Anaheim, California, USA, 2013. 2013: 89-99.

DOI: 10.16182/j.cnki.joss.2015.06.019