

基于图形硬件的纹理图像编码与实时绘制算法

汤 颖¹⁾ 张宏鑫²⁾ 张美玉¹⁾

¹⁾ (浙江工业大学软件学院 杭州 310014)

²⁾ (浙江大学 CAD&CG 国家重点实验室 杭州 310027)

摘 要 真实感绘制对于细节的要求越来越高,应用程序通常采用多幅或大幅分辨率很高的纹理图像,有限的内存空间就成了一个制约的瓶颈.针对纹理图像的特点和可编程图形硬件的特殊要求,该文提出了一种新的面向绘制的编码算法——增量式纹理编码算法及相应的解压绘制算法,有效地解决了纹理存储容量和真实感之间的矛盾,并利用可编程图形硬件实现了实时解压绘制.该算法在图像压缩编码过程中,动态添加码表内容,只有当已有码表内容不能表示当前图像区域时,才增加码表内容.这种方法不仅能够对于自相似性较强的纹理图像取得很高的压缩比,而且由于码表的动态更新特性,可以对图像序列进行流式编码.在绘制纹理时,该算法充分利用了现有可编程图像硬件的特性,实现了实时解压绘制.文中分别对于静态图像和动态图像序列进行了实验,结果显示,此方法能灵活有效地对各类纹理图像进行编码.

关键词 纹理图像;图像编码;可编程图形硬件
中图法分类号 TP391

GPU-Based Texture Encoding and Real-Time Rendering

TANG Ying¹⁾ ZHANG Hong-Xin²⁾ ZHANG Mei-Yu¹⁾

¹⁾ (Software School, Zhejiang University of Technology, Hangzhou 310014)

²⁾ (State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310027)

Abstract In photorealistic rendering, multiple high-resolution texture images are needed to add details to the scene, which makes the limited texture memory exhausted soon. This paper proposes a novel encoding algorithm for texture image—incremental texture encoding algorithm, for which the compressed textures are decompressed and rendered in real-time by programmable graphics hardware. This method gradually fills in the codebook by dynamically adding in the elements when the current codebook can not encode the region of the image being processed. In this way, the method can not only produces high compression ratio for texture images with self-similarity, but also encode the image sequences in a stream-like manner due to the property of dynamically updating the codebook. Experimental results show that the method is effective and efficient in encoding and rendering still/animated textures.

Keywords texture; image encoding; programmable graphics hardware

1 引 言

纹理映射作为真实感绘制的一种重要技术,自

从 Catmull^[1]提出后,得到了广泛的应用,目前多数硬件绘制平台支持实时纹理映射.要实现实时纹理映射,硬件需要专门的内存空间来存储纹理图像以支持对像素的实时读写.当应用程序需要多幅或大

幅分辨率很高的纹理图像时,就会占据大量的内存空间.随着真实感绘制对于细节的要求越来越高,有限的内存空间就成了一个制约的瓶颈.有两个方法可以解决这个问题:增加内存的容量和降低存储量、提高内存利用率.增加内存的容量,将提高硬件的成本,同时面对日益增加的要求,被动地增加内存容量不是一个有效的解决方法.另一方面现有硬件处理能力快速提高,特别是可编程图形硬件(GPU)的快速发展,使得第二个方法变得可能.本文提出了一种新的利用 GPU 直接解压绘制的纹理编码算法,利用较少的内存就可以实现对需要大幅或多幅高分辨率纹理图像的静态和动态场景的实时绘制.

要利用可编程图形硬件进行实时绘制,必须首先满足它对于绘制数据的一些特殊要求. Beers 等在文献[2]中指出,绘制数据必须满足两个重要要求:硬件能够实时解压和随机存取图像的任意区域.这使得通用的图像压缩算法(如 JPEG),不能适用于此类应用.这些通用算法在每个像素处的压缩率都不一样,致使现有的 GPU 随机存取图像的任意区域的计算代价大大增加,且变换编码导致解压速度大大降低,严重影响硬件绘制速度.

同时纹理图像作为增加物体表面细节的一种特殊图像,有别于一般的图像.人们主要关注的是纹理图像所表现的细节.而物体表面细节一般都呈现出高度相似性,其分布可看成由静态马尔可夫随机场(Markov random field)产生的随机过程.纹理图像的另一个重要应用是,利用多幅相似纹理图像形成动画序列.因而在压缩此类图像时,必须考虑如何有效地保留这些细节特性以及利用图像的自相似性.

Beers 在文献[2]中提出使用矢量量化(Vector Quantization, VQ)的方法压缩纹理图像并用软件模拟直接从压缩数据中绘制纹理图像,这种方法压缩率较高且解压速度很快.但是这类算法压缩速度慢,解压图像对于细节的保持不是很好. S3 公司提出了 S3TC 纹理压缩技术且已经在绘制硬件中实现,这种算法简单且压缩图像质量较高^[3],但此方法对所有图像的压缩率固定,一般为 6,没有针对纹理图像的自相似特点来达到更好的压缩结果.后来陆续出现一些算法对 S3TC 进行了改进,如文献[4]中提出了将 S3TC 扩展以支持 Mipmap 技术;文献[5]中改进了 S3TC 中块大小的选取和块对应的颜色的选取,使解压图像质量更高;文献[6]中提出颜色共享的策略,使每个块可对应的颜色更多,从而压缩质量更高;文献[7]使用低频信号的连续性来减少块与块

之间的不连续.但这些改进没有针对具有纹理自相似性特点的图像进行,都是面向一般图像.

本文针对纹理图像的特点和可编程硬件的要求,利用纹理合成的思想提出了一种新的图像编码方法:增量式纹理编码算法及绘制算法.基于样本的纹理合成^[8-11]通过输入的纹理样本按某种规则不断重用样本中的块合成出任意大小的与样本相似的纹理图像.而本文纹理压缩的过程就类似于它的逆过程,即从大幅的纹理图像中找到代表性的小幅纹理样本及其合成规则.由于这个小幅图像是直接由纹理图像取得,因而很好地保留了纹理图像的细节.算法将获取的纹理样本和合成规则,组装为一个新的纹理图像存入纹理内存,然后利用可编程硬件从纹理内存中解压绘制.本文提出的算法简单明了且易于实现,对于自相似性较高的纹理图像取得了很好的压缩效果,有效地解决了纹理存储容量、真实感之间的矛盾,并利用可编程图形硬件实现了实时解压绘制.

2 基于图形硬件的快速编码与绘制算法

本文提出的算法分为两个部分:编码和解码绘制算法.编码算法是离线算法,可以事先将程序要求的纹理图片进行压缩保存.解码绘制算法是实时的算法,当进行绘制的时候,程序直接从纹理内存中解压数据并且进行绘制.编码算法压缩的结果由码表和索引两部分组成.码表由可代表纹理图像的小幅纹理样本组成.索引保存了从纹理样本生成纹理图像的规则,具体而言,记录了纹理图像中每一部分可由哪个纹理样本表示.本文所提的纹理编码算法,采用“增量式”的形式.所谓增量式,系指图像压缩编码过程中,码表内容渐进动态地增加.该方法的基本想法是:如码表内容能表示当前图像区域,不需更新码表;只有在当前图像区域无法用码表内容合理表示时,算法才需要往码表中添加内容.这种增量式的纹理编码方法,不仅能找到单幅图像中重复区域的代表块,从而达到压缩图像的效果,而且由于码表的动态更新特性,使算法也适合于不断读入图像序列中的新图像进行流式编码.由于纹理图像的自相似性的特点,与整体求解码表的 VQ 算法相比,这个算法可以更高效快捷的压缩图像.同时,码表生成方法更为简便和灵活.在使用传统光栅化流水线技术绘制纹理时,程序直接根据纹理坐标查找到纹理图像中

相应的像素值,经过一定的滤波操作后将此像素值拷贝到对应屏幕区域上.现有可编程图形硬件,已能做到一次访问多个纹理数据,并可调用特定的顶点和像素着色器(pixel shader)程序来定制用户所需的绘制效果.基于图形硬件的这种特性,本文方法能做到在每次纹理查找操作直接包含解码步骤.这样,可在纹理内存中存储被压缩的纹理数据,并使用现有的图形硬件直接解压绘制出来.

2.1 编码结果的存储结构

本文算法采取两级层次的数据存储结构,包含纹理码表(texture codebook)和纹理图像的索引数据(index data).算法将输入纹理图像均匀分成正方形网格,每个网格包含的图像称之为纹理块.纹理码表由与纹理图像中有代表性的纹理块组成,这些纹理块我们称之为纹理样本.不同纹理样本存储在纹

理码表的不同位置,他们可由在纹理码表中的存储位置唯一表示.纹理图像的每个纹理块都有一个索引值,我们称为索引数据.索引数据存储的是可用于表示当前纹理块的纹理样本在纹理码表中的位置.所有这些索引数据组成了第一层结构,而纹理码表则构成第二层结构.图1中给出了利用两级层次的数据存储结构表示输入图像的一个例子.图1(a)表示输入的纹理图像和剖分的纹理块;图1(b)表示索引数据,左上角的纹理块的索引值是(0,4),表明该纹理块可由纹理码表中位置为(0,4)的纹理样本表示.图1(c)表示的实际的纹理样本,它的大小是 16×8 ,但是只包含了6个纹理样本.由于纹理图像被分割得到的不同的纹理块可以被同一个纹理样本表示,所以纹理码表中的纹理样本数比原图的纹理块数要少很多.

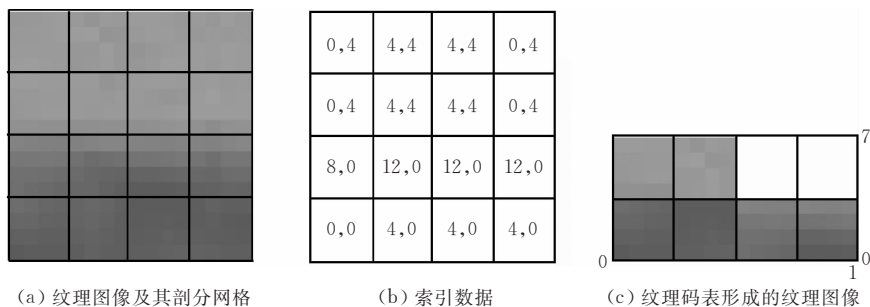


图1 算法数据结构示例图

2.2 纹理码表的增量式生成

纹理图像的索引数据的大小由纹理块的大小决定.纹理块越小,则剖分的网格越多,索引数据越多;纹理块越大,则索引数据越少.给定纹理样本和纹理码表,生成索引数据是一个显而易见的过程.而纹理码表的生成则是一个相对复杂的过程,而且它的生成质量也极大地影响了压缩效果和绘制质量.

假设纹理图像分成大小均匀的 $N_s \times N_t$ 纹理块.纹理码表看成可以动态增加纹理样本的队列.在对图像刚开始压缩时,将纹理码表的内容置空,按扫描线顺序对纹理块依次处理并编码.算法的主要步骤如下:

1. 将纹理图像中的第一个纹理块直接放入纹理码表.
2. 取出纹理图像中需要处理的当前纹理块,将其与纹理码表中的所有纹理样本进行比较,得到与当前纹理块最相似的纹理样本及相应的误差值 e_i .
3. 将 e_i 与程序中设定的最大误差阈值 e 进行比较,若 e_i 比阈值 e 小,则可以用第2步中找到的对应纹理样本来表示当前纹理块;若 e_i 比阈值 e 大,则认为当前纹理块无法由纹理码表中的纹理样本表示,将当前纹理块作为纹理样本加入纹理码表队列中.

4. 若已处理完纹理图像所有的纹理块,则压缩算法结束;若还有剩余纹理块,算法转到第2步继续执行.

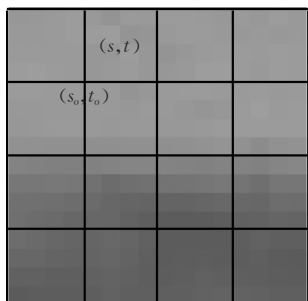
在比较纹理块和纹理样本时,我们采用 RGB 颜色空间的 $L2$ 颜色距离来度量两个块的相似程度.很多纹理合成的文献^[5-11]表明该度量是一个很有效的度量.假设要比较的纹理块和纹理样本分别是 X 和 Y ,两者之间的差异用 E 来表示,则我们有

$$E = \frac{1}{b^2} \sum_{i,j} ((X(i,j)_R - Y(i,j)_R)^2 + (X(i,j)_G - Y(i,j)_G)^2 + (X(i,j)_B - Y(i,j)_B)^2) \quad (1)$$

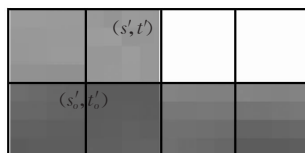
其中, b 是纹理块(纹理样本的)正方形的边长; $X(i,j), Y(i,j)$ 表示在纹理块和纹理样本中的 (i,j) 处的像素;下标 R, G, B 分别表示三个颜色的分量.根据式(1)计算纹理块和纹理样本之间的差异计算复杂度为 $O(b^2)$.可以预见随着编码的继续,码表长度增加,算法在码表中查找匹配纹理样本的时间也会随之增加.幸运的是,由于纹理图像的特殊性,这个缺陷并不影响算法效率.但是考虑到算法也应可以处理一些通用的图像,因此有必要引入一些加速的计算方法,如采用多分辨率表示纹理样本的方法、

超立方体测试算法^[12]等。

在纹理编码过程中,算法可类似分形算法^[13]引入块的旋转编码,增加当前块与码表中的块相匹配的几率。算法对纹理码表中的每个纹理块进行 8 种变换,包括 4 个方向的旋转和翻转并旋转 4 个方向。当前纹理块与码表中进行变换过的纹理块一一进行比较,找到最匹配的块,并记录下匹配块的变换方式



(a) 通过 (s, t) 得到所在纹理块的左下角纹理坐标 (s_0, t_0)



(b) 对应的纹理码表中的纹理样本的原点位置为 (s'_0, t'_0) , (s, t) 对应于块中的 (s', t')

图 2 对于给定的纹理坐标 (s, t) 查找对应的纹理数据

2.3 纹理码表的存储

由于硬件的纹理内存只能用于存储图像格式。压缩完成后,为了将纹理图像的索引数据和纹理码表载入硬件纹理内存来进行绘制,我们必需将索引数据和纹理码表组装成图像的形式。我们分别将由索引数据和纹理码表组装的图像称为索引数据图像和纹理码表图像。索引数据图像的生成十分简单,假设正方形纹理块的宽度是 b ,则索引数据图像相当于一副缩小了 b 倍的原图,唯一的区别是现在每个点存储的数据不是 RGB 图像值,而是索引值。在组装纹理码表图像时,由于码表中纹理样本的个数不是一个固定值,纹理码表图像的形式没有一个固定的形式。为了便于统一,将纹理码表图像的宽一致设为 256,而纹理码表图像的高度则根据纹理码表中纹理样本的个数来决定。根据此方法,如果纹理样本大小为 4×4 ,则纹理码表图像中可能得到的最大图像空白区域大小为 63 个元素,由于原纹理图像较大,通常为几十万个元素大小,所以此数据对于压缩率的计算几乎没有什么影响。得到纹理码表图像后,在索引数据图像中存入当前纹理块对应的纹理样本在纹理码表图像中的位置。

2.4 基于图像硬件的纹理绘制

在利用图形硬件进行绘制时,我们需从纹理内存中读取当前绘制的像素的对应纹理值。而由于纹理内存中存储的压缩数据,程序不能直接获取该值,必须利用索引数据图像和纹理码表图像,采用 GPU

(8 种方式之一)。显然,加入旋转可以增加同一个纹理块被重用的几率,即增加图像压缩比。但是为了记录每个匹配块的变换方式,需要为原图每个索引数据块加上额外记录信息,这样会增加索引数据的存储量。这样引入旋转编码并不一定总是有好处的,在实验结果中我们会对此进行分析。

的像素着色器编程来解压和绘制纹理图像。当查找并绘制原纹理图像的点时,假设其坐标为 $(s, t) \in [0, 1] \times [0, 1]$,我们需要找到它在纹理码表图像中的对应点,假设其坐标为 (s', t') 。从而获取该点的纹理数据。

具体绘制算法如下:

1. 根据当前纹理坐标 (s, t) 确定其所属纹理块和该纹理块左下角在原图像中的纹理坐标 (s_0, t_0) 。
 2. 根据索引数据图像得到纹理块的索引值,即对应纹理样本在码表图像中的位置,从而确定该纹理样本左下角在纹理码表图像中纹理坐标 (s'_0, t'_0) 。
 3. 根据相对关系,利用 (s, t) , (s_0, t_0) 和 (s'_0, t'_0) 可计算出当前点位于纹理码表图像中纹理坐标 (s', t') 。
 4. 根据 (s', t') 在纹理码表图像中查找对应的纹理数据。
- 将原纹理图像的大小表示为 $n_s \times n_t$,同时定义纹理码表图像的大小为 $n'_s \times n'_t$ 。则第 3 步的计算可表示为

$$s' = s'_0 + (s - s_0) \times \frac{n'_s}{n_s}, \quad t' = t'_0 + (t - t_0) \times \frac{n'_t}{n_t} \quad (2)$$

按上述方法直接绘制的纹理图像在放大时会出现块状走样,因为此方法类似于 OpenGL 中的最近邻滤波技术。为此在纹理绘制时需要进行线性滤波消除走样。绘制算法首先通过 (s, t) 计算出与其最相邻的 4 个纹元像素的位置,根据这 4 个像素位置按照上述的方法(1~4 步)来得到对应的纹理数据值,然后对这 4 个纹理数据值进行双线性插值得到最终的纹理值。实验发现采用线性滤波的绘制效果比最近邻滤波好,在图像放大时消除了块状走样现象。

3 实验结果及讨论

本节将介绍本文纹理图像编码算法所选用的参数对于图像编码结果的影响,并在文中给出了算法对于不同图像的实验结果.我们在 Nvidia GeForce FX 5600 显卡上实现了基于 GPU 的像素纹理坐标查找和插值计算.

3.1 算法参数的选取

本文算法中影响压缩算法效果的两个主要参数是:纹理块(纹理样本)的大小 b 及最大误差阈值 e .

对于纹理块大小 b ,在实现中考虑了 2×2 和 4×4 两种情况.当 b 为 2×2 时,原图像被分割成较多的块,索引数据量较大,是 b 为 4×4 时的索引数据量的 4 倍.但同时发现当 b 为 2×2 时,纹理码表对于原图像有较高的重用率,即可以生成较小的纹理码表,而 b 为 4×4 时,在同一个最大误差阈值下压缩得到的纹理码表元素个数大大多于 b 为 2×2 时得到的码表元素个数,即纹理码表所需的存储量



原图像



压缩率 = 3.4, MSE = 10.0



压缩率 = 6.0, MSE = 13.6



压缩率 = 9.6, MSE = 16.7

图 4 图 3 标识区域压缩图像比较

计算压缩率时,需要计算压缩后得到的两幅图像,即索引数据图像和纹理码表图像的大小.在我们的实验中索引数据图像中每个像素的大小为两个字节,其中每个字节表示当前纹理块在纹理码表图像 x 或 y 方向的位置,这样最多能表示 256 个不同位置,足够表示出组装纹理图像 x 或 y 方向不同的位置.原图像为 RGB 图像,每个像素占 3 个字节.当纹理块是 2×2 大小时,显然得到索引数据图像的像素个数为原图像的像素个数的 $1/4$,而每个像素大小为 2 个字节,这样求得此时索引数据图像的大小为原图像的 $1/6(1/4 \times 2/3)$.同样可以算出,当纹理块大小为 4×4 时,索引数据图像大小为原图像的 $1/24(1/16 \times 2/3)$.在不同的误差阈值下,算法得到不同大小的纹理码表图像,当阈值小时,相应的纹理码表图像较大,阈值越大,纹理码表图像越小.从表 1 中的结果可以看出,当纹理块较小时(如 2×2),索引数据图像很大,是决定压缩比的主要因素,而纹理码表图像的大小对最终压缩比的影响不大,所以提高误差阈值对压缩比影响不大.当纹理块较大时

较大,有关算法中的最大误差阈值可由用户给定.当阈值较小时,解压图像可以得到较好的质量,但压缩比有所下降;阈值较大时,压缩比较高,但解压图像质量有所降低.在表 1 中给出了图 3 在不同纹理块大小(2×2 和 4×4)与不同误差阈值下的压缩率及解压图像与原图像相比得到的 MSE (Mean Square Error, 均方差)值.在图 4 中给出了图 3 中的矩形框内的局部区域在不同压缩率下的压缩图像质量比较,可以看出压缩图像质量与压缩比成反比,当压缩比增加时,压缩质量随之有所下降.



图 3 石头纹理图像(512×512)

(如 4×4),索引数据图像较小,此时纹理码表图像的大小是决定压缩比的主要因素,提高误差阈值可以增加压缩率,但同时所得到的压缩图像的质量也有所下降.

在纹理编码过程中,如果引入旋转编码,需要为原图每个索引数据块加上额外的 3 个比特(8 种方式),这样会增加索引数据的存储量.由于增加了 3 个比特,则不适于再选取 2 个字节来表示索引数据图像的每个像素,应当选取 3 个字节.此时当块为 4×4 时,索引数据图像为原图像的 $1/16$.在增加块的旋转后,对于图 1 计算出的相应压缩结果如表 2 所示,在表 2 中没有计算块为 2×2 时的压缩结果,因为此时索引数据为原图像的 $1/4$,则压缩率一定低于 4,不能达到提高压缩率的效果.从表 2 中看出在引入块的旋转后,解压图像质量基本没有变化,但压缩率有了一定的提高.但是引入块的旋转使得解压过程变得复杂,解压速度有所影响.所以在强调绘制效率且压缩率不是要求很高的情况下,算法不宜引入块的旋转.

表 1 图 1 的压缩数据比较

纹理块大小/像素	误差阈值	索引数据图像：原图像	纹理码表图像：原图像	压缩率/%	MSE
2×2	15	1：6	1：39	5.2	9.8
2×2	20	1：6	1：85	5.6	12.7
4×4	15	1：24	1：4	3.4	10.0
4×4	20	1：24	1：8	6.0	13.6
4×4	25	1：24	1：16	9.6	16.7

表 2 加入块的旋转后图 1 的压缩结果

纹理块大小/像素	误差阈值	索引数据图像：原图像	纹理码表图像：原图像	压缩率/%	MSE
4×4	15	1：16	1：7	5.0	10.1
4×4	20	1：16	1：16	8.0	13.4
4×4	25	1：16	1：32	10.7	16.7

3.2 静态图像的压缩绘制结果

在图 5 中给出了两个纹理图像,其中图 5(a)是采用 Image Quilting 算法^[9]合成的纹理图像,图 5(d)为一幅木材纹理图像.表 3、表 5 分别列出了采用本文算法后的压缩结果数据.此处纹理块大小均为 4×4.从表中给出的压缩结果中可以看到增量式纹理图像编码算法可以有效地对图像进行编码,压缩率很高,且压缩质量也较好.

在图 5 中还分别列出了在 OpenGL 双缓存环

境下,用原始纹理映射三维模型和用图像硬件实时解压压缩数据绘制三维模型的效果图.图 5(a)绘制到人头模型上,模型有 10431 个顶点和 20720 个面片;图 5(d)绘制到木桌的三维模型上,此模型有 41 个顶点和 42 个面片.表 4 和表 6 列出了原始图像和相应的压缩数据的绘制帧速和绘制时产生的 MSE.在图 5 中给出了绘制效果图.从表 4 和表 6 的数据看出,本文算法的解码速度几乎达到原来的绘制速度,且能绘制出视觉效果很好的结果.

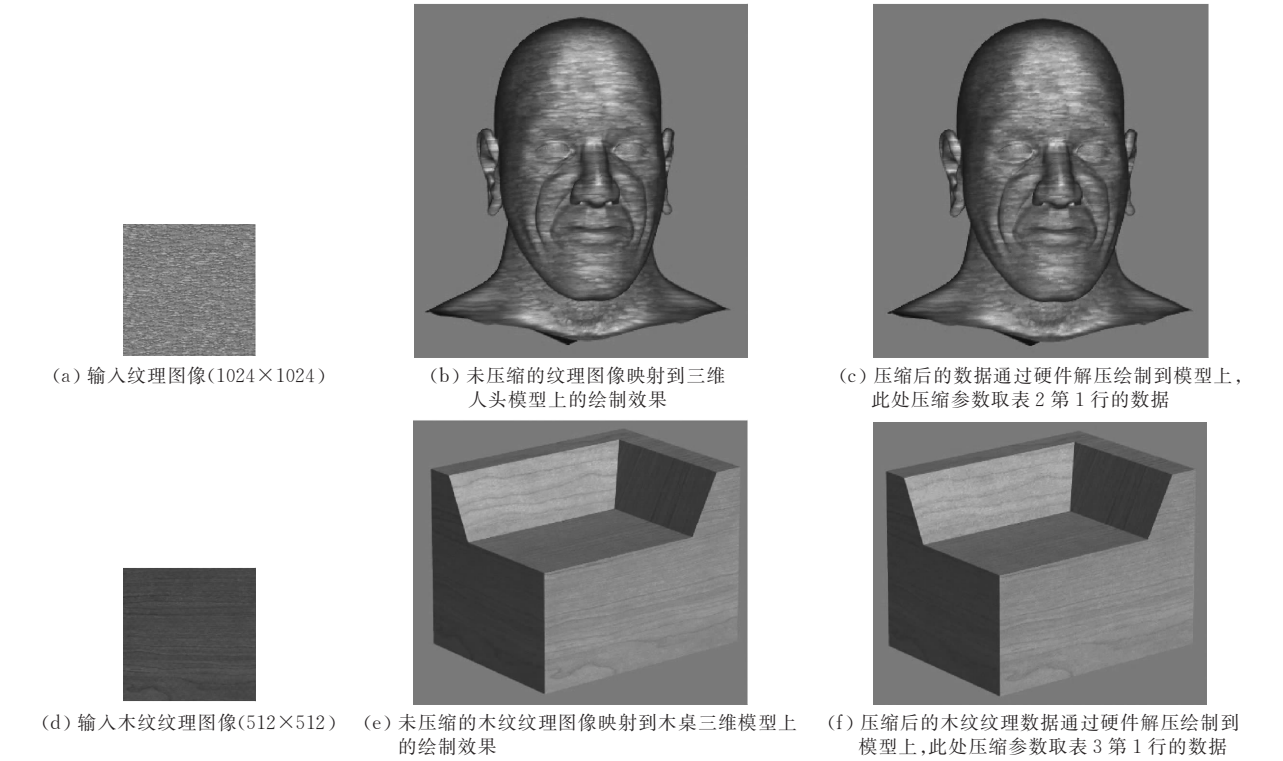


图 5

表 3 图 5(a)的压缩结果

纹理块大小(像素)	误差阈值	索引数据图像：原图像	纹理码表图像：原图像	压缩率/%	MSE
4×4	15	1：24	1：29	12.9	11.3
4×4	20	1：24	1：128	20.2	14.3

表 4 图 5(a)及其压缩数据的绘制时间和压缩数据与原图绘制效果相比所得的 MSE

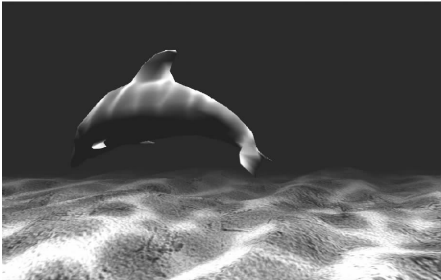
使用纹理图像	采样方式	平均每秒帧数	平均绘制的 MSE
图 5(a)	双线性插值	25.50	
根据表 2 第 1 行参数压缩图 5(a)	点采样	25.20	11.0
所得的压缩数据	双线性插值	21.30	9.0

表 5 图 5(d)的压缩结果

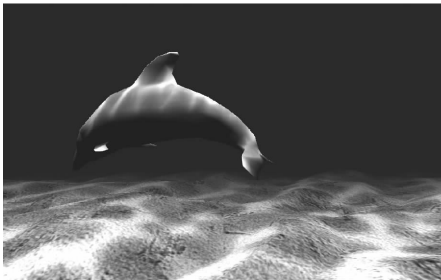
纹理块大小/像素	误差阈值	索引数据图像：原图像	纹理码表图像：原图像	压缩率/%	MSE
4×4	15	1：24	1：42	15.3	11.0
4×4	20	1：24	1：256	21.9	13.3

表 6 图 5(d)及其压缩数据的绘制时间比较压缩数据与原图绘制效果相比所得的 MSE

使用纹理图像	采样方式	平均每秒帧数	平均绘制的 MSE
图 5(d)	双线性插值	42.50	
根据表 4 第 1 行参数压缩图 5(d)	点采样	42.45	10.5
所得的压缩数据	双线性插值	42.10	8.0



(a) 原始例程绘制结果



(b) 对算法生成的压缩数据解压并线性滤波绘制的结果

图 6 海豚游动时的单幅截图(这两幅图像之间的 MSE 是 12.0)

3.3 动态图像序列的压缩绘制结果

根据增量式压缩算法可以灵活增加码表的特点,本文拓展该算法来压缩由多幅相似纹理图像形成的动画序列.对于多幅相似图像,算法不用为每幅图像单独生成纹理码表,所有的图像共用一个纹理码表,每幅图像的索引都指向这个纹理码表.当算法逐幅压缩图像时,当前的纹理图像可以重用先前已生成的纹理码表中的内容,由于这些图像都很相似,所以当前图像重用纹理码表的概率较高,这样可以大大节省纹理码表存储空间.传统的视频压缩方法,如 MPEG 等,并不适用于动画序列的压缩,因为它们需要对图像序列进行变换编码,图像不同区域的压缩率不一致,在这种编码方式下图形硬件不能很好地支持对压缩数据的实时高速解压和随机访问图像的任意区域.而本文的编码算法则不存在此问题.

我们对 DirectX 9.0 SDK 提供的海豚例程(DolphinVS sample)修改多幅焦散(caustic)纹理的读取和绘制机制,实现了本文提出的对于动画序列的压缩算法.此例程中为了表现海豚在水里游动时光在水底和海豚身上的折射效果,使用 32 幅相似纹理图像来产生动画.算法对这 32 幅图像进行压缩,得到 32 幅索引数据图像和一个纹理码表图像.在表 7 中列出了相应的压缩结果,其中原图像大小包

括所有 32 幅纹理图像.在修改的例程中读入这些压缩数据,而不是原来的 32 幅纹理图像.在绘制纹理时,不能直接使用 D3D 提供的纹理采样函数,而是采用在 D3D 下编写的 pixel shader 代码来实时解压和采样压缩数据.算法同时实现了点采样和双线性滤波采样,我们发现在点采样时,纹理走样比线性滤波采样时明显,但绘制帧速明显比线性滤波快.点采样时的平均帧速为 380 帧/秒,线性滤波时的平均帧速为 80 帧/秒,都达到了实时绘制的要求.图 6 给出了在海豚游动时的绘制效果比较,其中左图为原始程序绘制结果,右图为利用算法生成的压缩数据绘制的结果,此处纹理采样时采用了线性滤波.读者可访问 <http://www.cad.zju.edu.cn/home/ytang/> 下存放的两段视频来比较绘制效果,一段为未压缩纹理图像时的海豚游动动画,另一段是采用压缩纹理数据来进行绘制的动画.

表 7 海豚例程中 32 幅纹理图像的压缩结果

纹理块大小 /像素	索引数据图像： 原图像	纹理码表图像： 原图像	压缩率/%
4×4	1：24	1：18	10.3

从上述结果中可以看到本文提出的算法对于一系列相似的纹理图像形成的动画效果具有较好的压缩和解压绘制效果.可以预测当图像序列越长时,因

为重用率更高,压缩率会越大.其实不仅动态图像序列适用于此算法,凡是对于需要多纹理图像映射且这些纹理图像相似的应用程序,都可以采用此算法来进行压缩.比如在 Visual Hull 中为了绘制带有纹理的物体,需要在物体上映射并拼接多幅不同角度拍摄的纹理图像,对于这种应用也可以运用本文的算法来达到节省纹理内存空间的目的.

4 结论与未来的工作

针对纹理图像的特点,本文提出了一种基于“逆”纹理合成思想的面向绘制的增量式纹理图像编码算法.此算法通过灵活简便的方法来动态增加纹理码表中的元素,进而达到对图像进行编码和压缩图像的目的.该算法利用现有的可编程图形硬件的特性,可以实时解压绘制编码后的图像.实验结果显示,本文的算法对于静态纹理图像和相似纹理图像组成的图像序列具有很好的压缩和绘制效果.

增量式纹理编码算法与基于 VQ 的纹理压缩算法^[2]比较相似,最后都得到索引数据和码表,都能使用现有的可编程硬件进行实时绘制,两者之间的区别在于纹理码表的生成.VQ 算法是一种全局优化的算法,预先给定码表的大小,然后进行迭代优化生成码表中的项.本文的算法不用预先给定码表的大小,而是在压缩过程中根据需要动态增加码表的项,且不需要迭代.和 VQ 相比,增量式算法所得的码表也许不是在同样大小的码表中最优的码表,但是增量式算法中码表不用预先固定,可以根据需要灵活的改变,且其码表中的项也可以看成局部最优算法所得的结果.同时,本文算法在比较两个纹理块时还可以类似分形引入块的变换,这样也与一般的 VQ 算法不同.由于不需要迭代,本文算法的编码速度也比 VQ 快.

下一步,我们将在此基础上研究对图像的自适应编码,根据图像不同区域的不同重要度来有效地进行编码.

参 考 文 献

- [1] Catmull Edwin. Computer display of curved surfaces//Pro-

ceedings of the Conference on Computer Graphics, Pattern Recognition, and Data Structure. Los Angeles, California, 1975

- [2] Beers A C, Agrawala M, Chadda N. Rendering from compressed textures//Proceedings of the Annual Conference on Computer Graphics, ACM SIGGRAPH. New Orleans, Louisiana, 1996: 373-378
- [3] Konstantine Iourcha, Krishna Nayak, Zhou Hong. System and Method for Fixed-Rate Block-Based Image Compression with Inferred Pixel Values. US Patent 5,956,431
- [4] Pereberin A V. Hierarchical approach for texture compression//Proceedings of the GraphiCon. Moscow, Russia, 1999: 195-199
- [5] Levkovich-Maslyuk L, Kalyuzhny P G, Zhirkov A. Texture compression with adaptive block partitions//Proceedings of the 8th ACM International Conference on Multimedia. Los Angeles, CA, USA, 2000: 401-403
- [6] Ivanov Denis, Kuzmin Yevgeniy. Color distribution—A new approach to texture compression. Computer Graphics Forum, 2000, 19(3): 283-289
- [7] Fenney Simon. Texture compression using low-frequency signal modulation//Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware. San Diego, CA, USA, 2003: 84-91
- [8] Wei L Y, Levoy M. Fast texture synthesis using tree structured vector quantization//Proceedings of the Annual Conference on Computer Graphics, ACM SIGGRAPH. New Orleans, Louisiana, 2000: 479-488
- [9] Efros A A, Freeman W T. Image quilting for texture synthesis and transfer//Proceedings of the Annual Conference on Computer Graphics, ACM SIGGRAPH. Los Angeles, California, 2001: 341-346
- [10] Liang L, Liu C, Xu Y Q, Guo B, Shum H-Y. Realtime texture synthesis by patch-based sampling. ACM Transactions on Graphics, 2001, 20(3): 127-150
- [11] Paul Harrison. A non-hierarchical procedure for re-synthesis of complex textures//Proceedings of the 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision. Plzen, Czech Republic, 2001: 190-197
- [12] Soleymain M R, Morgera S D. An efficient nearest neighbor search method. IEEE Transactions on Communications, 1987, 35(4): 677-679
- [13] Fisher Yuval. Fractal Image Compression: Theory and Application to Digital Images. New York: Springer Verlag, 1995



TANG Ying, born in 1977, Ph. D., lecturer. Her research mainly focuses on mesh parameterization, texture synthesis, texture compression and visualization.

ZHANG Hong-Xin, born in 1975, assistant professor. His research interests include geometric modeling, texture synthesis and machine learning.

ZHANG Mei-Yu, born in 1965, associate professor. Her research mainly focuses on artificial intelligence, computer graphics and image processing.

Background

This research is supported by the National Natural Science Foundation Project "The Virtual Simulation of Forests Based on the Dynamic Growing Models" (No. 60403046) and the National Natural Science Foundation Project "Local Wave Analysis and Application on Discrete Geometric Signal" (No. 60673063). The first project focuses on building the complete forests model based on the gaps and the dynamic growing models of the trees in forests. The second project is to study the application of newly developed theory and techniques in local wave analysis to geometric signal processing.

The objective of this research is to solve the conflict between the limited cache size of graphic hardware and the requirement of larger scale texture which increase the richness of the scene as well as the demand on memory size. Texture and texture mapping play significant roles in computer graphics to add visual richness without increasing scene geometric complexity. Most graphics hardware provides dedicated

memory cache to store textures for real-time texturing. However, the cache is limited in size and easily filled by high-resolution or large-sized texture images, necessitating the highly efficient compression and on-the-fly decompression of texture. Beers et al. proposes to compress texture image using Vector Quantization. This approach achieves high compression rate with acceptable visual quality loss. S3TC is another remarkable compression strategy widely used in graphics hardware community. However, these approaches did not take into the special characters of textures pictures, which resemble itself like fractural. Since texture mapping is widely used in computer graphics, it is necessary to develop special approach to handle this problem by utilizing the newly developed function of computer hardware. To our knowledge, this paper is the first paper to address this problem. The results show that the proposed method is effective and efficient in encoding and rendering still/animated textures.