LARGE TEXTURE STORAGE USING FRACTAL IMAGE COMPRESSION

Stachera Jerzy[1] and Nikiel Sławomir[2]
[1]Institute of Computer Science, Warsaw University of Technology, ul Nowowiejska 15/19, 00-665 POLAND, e-mail: J.Stachera@elka.pw.edu.pl, [2]Institute of Control and Computation Engineering, University of Zielona Góra, ul Podgórna 50, 65-246 Zielona Góra, POLAND, e-mail: S.Nikiel@issi.uz.zgora.pl

Abstract: Texture mapping has traditionally added visual realism to computer graphics images. Modern graphic boards reserve more space for image textures that are mapped in real time on 3D meshes. Outside the graphic accelerators texture images are stored in popular compressed formats that are not designed to meet real-time texture decompression requirements. In the paper we forward an alternative technique for image texture handling. We use fractal image compression scheme to decompress very large textures. The main idea is to use high degree of local self-similarity in texture images representing natural scenes. The method delivers very high compression ratio for complex textures and close to real-time decompression. Local compression schemes allow for Region Of Interest access to the visual data. Properties of the method are compared to those of classical DCT and wavelet compression schemes.

Key words: fractal compression, texture mapping, multiresolution

1. INTRODUCTION

Texture mapping is used to add visual detail to a computer generated scene and in its basic form it lays an image map onto polygon objects[1]. When mapped onto an object the appearance of the object is modified by a corresponding data from the image, this can be patterns, bumps or others [2,3]. The image is typically a sampled array so a continuous seamless texture must first be reconstructed from the samples. During the mapping the image must be warped to match perspective distortions. Then the warped texture is filtered to avoid aliasing artefacts. The required filtering is approximated by one of several methods [4,5].

Current applications of three dimensional computer graphics, however, require much larger textures to achieve high degree of visual realism. This is particularly visible in the humanoids and the outdoor scenes where repeating patterns create artificial feel of the scene. The problem of texture handling arises in case of games and virtual reality simulations. This is particularly visible in panned background panoramas and in extreme close-ups where visual detail of is of most importance. Some general textures and geometry are stored and accessed locally but all changes in the scene require heavy mirroring and downloads of upgraded data. Image textures can be too big to download via broadband internet connection. There is a continuous need for efficient texture transfer then.

2. PROBLEM DEFINITION

Most image compression schemes are oriented towards compression for storage or transmission. In choosing a compression scheme for texture mapping there are several restriction to consider [8].

Decoding speed. Textures are compressed once and decompressed many times, it is the asymmetric process. Thus, decoding speed it is a main factor which states whether given method is suitable.

Random access. Region Of Interest (ROI) access to a compressed texture data allows for rendering directly from compressed textures. Texture compression scheme provides fast random access to texture data. Otherwise, the process of decompression may limit performance of the rendering pipeline.

Compression rate and Visual Quality. High compression rate results in storing more textures in memory or alternatively in high-resolution textures. Texture downloads are more efficient. While lossless compression methods preserve original texture data they do not achieve high compression rates comparing to lossy methods.

3. FRACTAL IMAGE COMPRESSION

3.1 PIFS

Fractal image compression was made possible thanks to the work of M. Barnsley and A. Jacquin. For that purpose there was introduced an extension of IFS called a partitioned iterated function system (PIFS). A PIFS consist of a complete metric space (X, d), a set of domains $D \in X$, and a set of contractive transformation $W : D \to X$. By the Contraction Mapping Fixed-Point Theorem, W has a unique fixed point $f_w \in X$, satisfying $f_w = W(f_w)$. Thus, the image encoding problem is to find a PIFS such that its fixed point f_w is as close as possible to encoded image $I \in X$. Therefore, it is required that W minimises the distance between its fixed point f_w and encoded image I . Finding such a PIFS for a given image can be computational expensive since it involves a minimisation over many transformations. In order to make this problem solvable only one class of transformations is considered.

3.2 Fractal texture compression

The process of image coding is based on a set of contractive transformations W such that its fixed point f_w is an approximation to the coded image I . Thus W defines a lossy code for the image I [7] [15].The compression scheme for textures is based on a block oriented fractal compression scheme for images. Moreover, our algorithm allows ROI access and local decompression of texture regions.

For compressing textures we define additional assumptions extending the basic fractal encoding scheme:

1. $N \times N$ - the size of the texture ($N = 2^l$),
2. $B \times B$ - the size of the range block ($B = 2^n$),
3. $D \times D$ - the size of domain block is twice the size of a range block ($D = 2B$),
4. $\varphi(\cdot)$ - the spatial contraction function averages four adjacent texture elements and then maps the averaged value onto range block applying one of eight isometries. The resulting range block values are scaled by s_i and added to o_i :

$$r_i = w_i(d_i) = s_i \varphi(d_i) + o_i \tag{1}$$

5. the set of range blocks R comes from a quadtree partition of the texture [9].

In its basic form the fractal coding scheme does not allow for local decoding. It is not possible to decode only one region of a texture without decoding all the domains which are related to a given region. In our algorithm, we solve the problem of local decompression by restricting the search area to a given range. The search regions are defined by some initial number of quadtree partitions of the texture less than the number of minimal partitions. In the terms of tree representation (Figure 1) which is created after quadtree partitioning, search regions represent nodes at a level of the tree which is set between the root (initial texture) and the minimum tree depth. Thus each region is a square area. Its size is at least twice the range size and defines independent domain pool. In searching part of the algorithm we compare only ranges with domains which are contained in the same search region. Each search region can be treated independently from each other. It allows for ROI access to texture regions and local decompression. For Lenna image (Figure 2), the search region size is 32x32 (white border), the minimum range size is 4x4 and the maximum range size is 16x16 (black border).

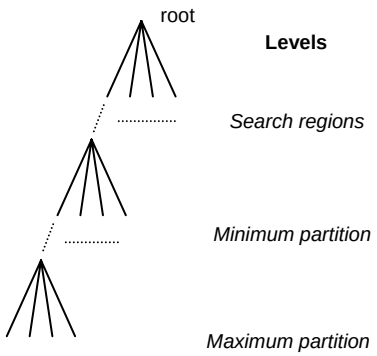

Figure 1. Quadtree partitioning

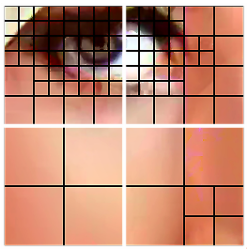

Figure 2. Lenna. Image size 512x512, minimum quadtree depth 5, maximum quadtree depth 7, search region depth 4

The fractal compression scheme was introduced to encode greyscale images. In most cases textures are colour images written in a format based on the RGB colour space. To take advantage of human insensitivity to colour changes we can convert a texture to the YUV colour space. The YUV colour space is consisted of three channels: luminance channel Y and two chrominance channels U (hue), V (saturation). The chrominance channels store information about colour and it is possible to compress them with little to no visible degradation.

The overall scheme for fractal texture compression consists of the following steps:

- we convert texture colour space to YUV colour space,
- we average the chrominance channels (hue, saturation) to one-half,
- we set the minimum quadtree depth $q_{\min}$, maximum quadtree depth $q_{\max}$ and search region depth $q_{search} \in [1,...,q_{\min}]$
- for each component: Y, U', V' we use quadtree partitioning method and region search strategy
- we save header information: quadtree depth, number of transformation for each component, texture resolution, …
- we save quantized transform parameters with quadtree information for each component using variable length code.

The independent processing of each YUV component makes it possible to utilise parallel processing during the process of texture compression and decompression.

3.3 Fractal texture decompression.

The compression scheme for textures must take into consideration their applications. As it was said before the scheme must be mainly characterised by the fast decoding algorithm. However, the main drawback of presented scheme may be an expensive compression algorithm. It is also characterised by a number of relatively simple decoding algorithms. Several decompression methods have been developed to optimize the decompression process[9][12]. It was stated that hierarchical decoding method is one order of magnitude less computational expensive than iterative method (assuming $B^2 >> it$, where it is equal to number of iterations in the iterative method) [13]. Moreover, we can decompress texture in a finite predetermined number of steps which depends on partitioning of the texture rather than on the texture image itself. The hierarchical method was introduced only for fixed size range blocks. In our scheme we modify that method extending the case of quadtree partitioning by Malah D. [12].

One property of fractals is a resolution independence (or super resolution). It is also true for textures which are compressed using fractal compression method. In contrast to linear interpolation which tends to blur the texture, the fractal decompression method ensures the richness of details even at higher than original resolution. In our compression scheme we average the chrominance information. The super resolution property of the fractal decompression method allows us to decompress the chrominance channels (hue, saturation) at original resolution without the loss of visual details.

The input data to fractal decompression scheme comes from the presented fractal compression scheme. We can decompress texture blocks which size depends on the search region size used during compression. The decompression scheme utilises the hierarchical decompression method. In the first step the transformations for given texture block (range block and domain block size) are scaled by a factor $1/2^{\max}$ (where $2^{\max} \times 2^{\max}$ is the size of the biggest range block) in order to approximate the top level fixed point of the PIFS pyramid (these step can be omitted if we do this calculation before saving the coefficient to the file). At the top level we apply the transformation only once to level grey texture in order to approximate the fixed point $f^{1/2^{\max}}$. Then we double the resolution by multiplying the transformation by factor equal to two, and then we apply the transformation. The process is repeated $\log_2(2^{\max})$ times.

The overall decompression process for given texture block may be written in following steps, for each YUV component do (assuming, that U, V components are averaged to one-half):

1. we multiply the transformation by a factor of $1/2^{\max}$ - optional
2. we apply transformation to the top level
3. we multiply transformation by 2
4. we apply transformation
5. we repeat 3), 4) until the required resolution is achieved
6. for U, V components – we multiply transformation by 2, apply transformation.

4. EXPERIMENTAL RESULTS

Table 1. Efficiency of the three compression methods (*CR – compression ratio, PSNR – peak signal to noise ratio).

Image	FCI (32x32)		FCI (64x64)		JPEG		JPEG2000	
	CR	PSNR[dB]	CR	PSNR[dB]	CR	PSNR[dB]	CR	PSNR[dB]
Building	49.35:1	27.25	65.25:1	28.9	95.6:1	30.7	197.4:1	33.6
Earth	74.75:1	29.69	100:1	30.36	120.1:1	30	200.1:1	32.8
Nebulea	62.87:1	33.4	168.3:1	31.91	136.7:1	32.7	422.2:1	34.9
Sky	63.31:1	35.68	182.6:1	33.74	161.1:1	31.4	542.9:1	36.7
Map	51.18:1	29.8	68.76:1	30.32	89.2:1	28.7	126.9:1	30.1:1
Jupiter	47.41:1	31.9	71.7:1	32.11	96.4:1	34.2	243.2:1	32.8

We carried out the experiments mainly to achieve high compression rates (the images for experiments are located at the end of publication). The table shows the compression ratio for a given peak signal to noise ratio. The results are presented for a texture stored in 2048×2048 resolution in the RGB colour space. For fractal compression scheme, search region size are 32x32 (minimum partitions 7, maximum partitions 8) and 64x64 (minimum partitions 6, maximum partitions 8). The FCI is the format of data used in our compression scheme. It is based on a bit allocation scheme proposed by Y. Fisher[9], where the transform coefficients are allocated as follows: s_i (5-bit), o_i (7-bit), φ (3-bit) and variable length codeword is used for the domain location.

There is a relation between search region size and compression ratio. Enlargement of the search region size increases the compression ratio and the minimal size of texture blocks that can be decompressed independently. The decompression time depends only on the search region size and can be done in linear time $\mathrm{O}(n)$.

5. DISCUSSION

Most image compression schemes which use transform methods such as discrete cosine transformation (JPEG) or wavelet functions (JPEG2000) are oriented towards image quality and high compression rates. High compression rates in those methods are achieved by efficient data decorrelation, quantisation and entropy coding of transform coefficients. The process of decompression is usually multi-staged with expensive computation of inverse transform. Texture mapping imposes other constraints on the compression scheme. It is required from texture compression scheme to deliver a fast decompression algorithm. Because JPEG and JPEG2000 require a multi-staged decompression process, they are not suitable for real-time texturing applications. The proposed fractal compression scheme allows for random access for given texture block and local decompression. The compression ratio in our scheme depends only on texture block size. Increasing the texture block size we increase the compression ratio and the results are comparable to JPEG algorithms, with additional advantage of local decompression.

In most cases the aliasing problem in texture mapping is solved by the mip-mapping method. Changing sizes of texture one of the mip-maps which best approximates the texture is chosen. The method solves the problem of texture transform. Theoretically, it is possible to generate a set of mip-maps which can be used at any level of detail thus solving the problem of texture magnification. However, mip-map memory handling in this case may dominate the rendering process. Therefore in most hardware the interpolation method is used. The most commonly used linear interpolation method tends to smooth the texture, resulting in a blurred texture image. The proposed fractal decompression scheme solves the problem of texture magnification utilising fractal properties, namely: super-resolution and a simple decoding algorithm. The super resolution property is used in proposed scheme to decode the chrominance channels at a twice the encoded resolution. The approach adds details in colour information at higher than original resolutions which are the result of fractal transform and preserve sharpness of decoded texture. Moreover, the simple decoding algorithm makes it possible to implement in a dedicated hardware for real-time texturing or employ on a mobile device.

6. CONCLUSIONS AND FUTURE WORK

Texture mapping is a highly efficient technique adding complex visual detail to three dimensional geometry. Popular PC-based systems offer each year a new generation of accelerated graphics boards providing more space for image textures. Concerning virtual reality systems and gaming industry, there is still need for efficient very large image texture handling. High efficient compression scheme proposed in the paper works well with textures depicting natural scenes. Our method utilises local self-

similarity in natural images allowing for multi-resolution decompression and local decompression. Fractal texture handling offers significant savings both in storage and in time of decompression which is close to meet real-time constraints.

7. TEST TEXTURES.

Figure 3. A Building

Figure 4. Sky

Figure 5. Earth

Figure 6. Map

Figure 7. Nebulea

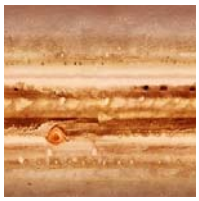

Figure 8. Jupiter

8. REFERENCES

1. Heckbert P., Survey of Texture Mapping, IEEE Computer Graphics and Applications, November 1986, pp. 56-67
2. Gardner G., Visual simulation of Clouds, Computer Graphics, Siggraph'85 Proceedings, July, 1985, pp. 297-303
3. Palewski M., Fractal shading model, MSc thesis, Institute of Control and Computation Engineering, University of Zielona Góra, February 2003 (in Polish)
4. Williams L., Pyramidal Parametrics, Computer Graphics, Siggraph'83 Proceedings, July 1983, pp. 1-11
5. Crow F. , Summed-Area Tables for Texture Mapping, Computer Graphics, Siggraph'84 Proceedings, July, 1984, pp. 207-212
6. Deering M. Et al., The Triangle Processor and normal Vector Shader: A VLSI system for High Performance Graphics, Computer Graphics, Siggraph'88 Proceedings, August, 1988, pp.21-30
7. Baharav Z., Malah D., Karnin E., Hierarchical Interpretation of Fractal Image Coding and Its Application, I.B.M Israel Science and Technology
8. Beers A., Agrawala M., Rendering from compressed textures,
9. Fisher Y., Fractal Image compression: theory and application, Springer-Verlag, 1995
10. Welstead S., Fractal and Wavelet Image Compression Techniques, SPIE, 1999
11. Saupe D., Fractal Image Compression via Nearest Neighbor Search, NATO Advance Study Institute Fractal Image Encoding and Analysis, Trondheim, 1995
12. Malah D., Hierarchical fast decoding of fractal image representation using quadtree partitioning, Technion - I.I.T, Israel
13. Cisar G., On entropy coding Fisher's fractal quadtree code, Joint Research Centre, Italy, 1996
14. Stachera J., Real-time Texturing using Fractal Image Compression, MSc thesis, Institute of Control and Computation Engineering, University of Zielona Góra, February 2003 (in Polish)
15. Stachera J., Nikiel S., Fractal Image compression for efficient texture mapping, The 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2004