FOVEATED TEXTURE MAPPING WITH JPEG2000 COMPRESSION

Timothy M. Henry and Bala Ravikumar
Department of Computer Science, University of Rhode Island, Kingston, Rhode Island

ABSTRACT

A current technique to control the rendering speed in interactive virtual worlds is the representation of object geometry using a hierarchical level of detail. Missing features can be supplied through the use of texture maps, but the reduced rendering time is bought at a trade-off in the amount of memory required to store the texture images. Compressed texturing techniques (e.g. S3TC, FTX1, VQ) offer some relief, but at a cost to image quality and detail.

This paper proposes a solution using texture images compressed with JPEG2000. The foveated JPEG2000 texture mapping method (fJ2k) decompresses codestreams to resolutions that vary based on current rendering performance and position of the requested texture image in the field of view. The final result is a foveated viewport rendered at interactive framerates using significantly less memory than current methods.

1. INTRODUCTION

As interactive three-dimensional simulations and virtual worlds continue to be used in a large number fields, there remain several limitations forcing design compromises. The motivation for foveated JPEG2000 texture mapping is to assist in the rendering of images for wide field of view devices, (head mounted displays), low memory devices, (game consoles and embedded devices), or three-dimensional environments requiring very large texture images. The memory requirement for storing each individual texture image can be greatly reduced, permitting a significant increase in the number of high quality textures that can be stored in fast internal or texture memory.

One of the most difficult design balancing acts is the one between level of detail, memory usage and frame rate. Two common techniques used to achieve the balance are hierarchical levels of detail (LOD) in model geometry and texture maps. To accomplish a frame rate that maintains an interactive feel, the complexity of modeling surface detail is reduced by mapping high quality images to an object's surface in order to represent features. While this reduces the frame rendering time, memory usage must increase to store the texture map and the resulting model detail is decreased.

Texture mapping is not a perfect solution. If a texture image has been reduced in size to save memory, pixelation or blurring can occur if the image is magnified. If a texture must be reduced during mapping (minification), artifacts, such as aliasing, can appear. To correctly map images to an object's displayed size, a hierarchical structure of textures, a mipmap, can be created. While this creates better detail, at least 30% more memory is required to store the mipmap structure. Though mipmapping can produce good results, if a textured surface stretches between several detail levels, such as when a surface is viewed edge-on, better results can be obtained by through anisotropic techniques [1], but these require even greater memory storage.

Texture caching can be used to reduce the number of textures or parts of textures in memory, but the increased time to access images outside the cache requires the designer to re-evaluate earlier design decisions to balance the use of texturing techniques and frame rates. Texture compression permits a greater number of images to be stored in texture memory at a sacrifice of frame rate since the images must be decompressed prior to rendering. The impact to the interactivity can be reduced when standardized codecs are implemented in hardware. Several texture compression implementations, S3TC [2], FTX1 [3], and VQ [4] use codebooks to represent pixels in the original image and though they have fast implementations, they result in lossy compression at ratios of only 12-15%.

This paper proposes a novel solution using images compressed using JPEG2000 [5]. The foveated JPEG2000 texture mapping method (fJ2k) uses JPEG2000 codestreams that are decompressed in the graphics pipeline to resolutions that vary based on current rendering performance and the importance of the requested texture image in the field of view. The final rendered result is a foveated image, with a high level of detail at the center of the field of view (fovea). The great flexibility and comparable image quality provided by the JPEG2000 standard come at compression ratios of less than 1bpp – under 5% compression for 24-bit color images.

1.1. JPEG2000 Overview

The JPEG2000 standard provides a flexible image compression method that allows random access into the encoded data stream and fast decoding [6]. There are several image compression techniques available in addition to those mentioned earlier, such as JPEG, JPEG-LS, PNG, and MPEG-4 VTC, but they do not afford the random access and highly flexible progressive bitstream options of JPEG2000. These and several other key advantages of JPEG2000 are identified and compared in [7]. This paper will highlight only the ones relevant to this research. The JPEG2000 encoding process can be summarized in the following steps:

- The image is decomposed into components. The standard supports two different component transformations, an irreversible component transformation (ICT) that approximates a YC_bC_r transformation of RGB components and a reversible component transformation (RCT) that approximates a YUV transformation.
- The image and its components are decomposed into rectangular tiles. The tile-component is the basic unit of the original or reconstructed image. The fJ2k algorithm uses these tile components to access and decode only the visible regions of the image.

• A wavelet transform is applied to each tile-component to create decomposition levels. The number of decomposition levels can vary amongst the tile-components and are made up of subbands of coefficients that describe the frequency characteristics of local areas.

• The subbands of coefficients are quantized and collected into rectangular arrays of code-blocks. The bit-planes of the coefficients in a code-block are entropy coded.

• Some of the coefficients can be coded to provide a region of interest (ROI). At resolutions below the image's maximum, the ROI will have a greater level of detail than surrounding areas.

• The encoded information is then placed in a progression that order that best suits the implementation's requirements (e.g. network transmission of codestreams).

The codestream header has information on the decomposition and coding styles used in the compression process and is used to locate, extract, decode and reconstruct portions of the image with the desired resolution, fidelity, region of interest and other characteristics.

1.2. Foveation and Motion Blurring

Human vision uses many cues to construct what we perceive from the vast amount the information in the world around us [8]. Many computer visualization algorithms take advantage of these to prioritize what order and how objects are rendered in virtual environments [9]. Because of the additional computations the fJ2k method introduces into the rendering pipeline to decode JPEG2000 images, we look to exploit cues – namely foveation and motion blurring – that lean on JPEG2000's strengths.

A foveated image is a non-uniform resolution image whose resolution is highest at the focal point (fovea) but decreases away from the fovea [10]. This distribution of resolution provides our visual system with a fast and simple way of reducing information in the visual field, without sacrificing the size of the field or the information around the center. Because foveation mimics aspects of biological vision, it is directly applicable to the processing of visual information for applications where a significant portion of the user's field of view is computer generated, such as for head-mounted-displays (HMD).

Algorithms that use traditional foveation begin with a uniform resolution image and apply a logarithmic function that decreases the level of detail based on the distance from the center of the image. The fJ2k 'method uses this concept to reduce the number of computations required to decode texture images. Textures used in rendering each frame of the scene are decoded to a resolution that varies depending on the distance the texture is from the center of the field of vision. This creates a three-dimensional sphere that is used to determine the level of detail and creates a frame that appears to have been generated from traditional foveation. The amount of frame visual information is reduced, but high detail is maintained at a specific region of interest.

Human vision tends to perceive quickly moving objects as slightly blurred. On our periphery, motion is more readily observed than detail and color.[8] FJ2k uses these vision characteristics to adjust the resolution level and color decoding for texture images. The goal is to decode only image detail that can be effectively utilized by the visual system.

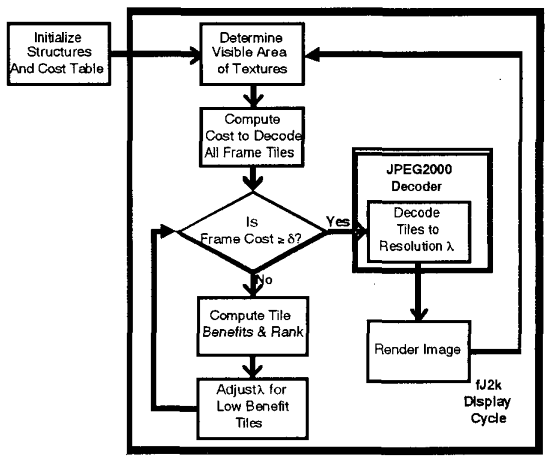
Initialize Structures And Cost Table
Determine Visible Area of Textures
Compute Cost to Decode All Frame Tiles
Is Frame Cost ≥ δ?
Yes
No
JPEG2000 Decoder
Decode Tiles to Resolution λ
Compute Tile Benefits & Rank
Render Image
Adjust λ for Low Benefit Tiles
fJ2k Display Cycle

Figure 1 – Foveated JPEG2000 Method

2. FJ2K METHOD OVERVIEW

The high compression efficiency provided by JPEG2000 comes at a cost of complexity and memory accesses during decompression. The foveated JPEG2000 method seeks to exploit the powerful feature set of JPEG2000 to offset the drawbacks making it a practical texture compression standard.

The JPEG2000 standard breaks an image into tiles for decoding. These tiles do not have to be aligned to the borders of an image, but are portioned from a reference grid that overlays the image. The reference grid can encompass an area larger than the image allowing regular sized tiles to have different sized blocks from the image. For fJ2k, the reference grid and tile sizes are adjusted to balance the compression ratio and model's structure. The goal is to have the tile division closely match the contours of the object. This design criterion provides a codestream structured to minimize extraneous region decoding. The entire texture codestream for an object can be stored in memory, but only regions that are currently visible need to be decoded.

The texture images are then encoded using the reversible component transform (RCT). This incurs a few additional operations per final texel in the decoding process, but it provides significant image information in the Y-component allowing the decoding process to be halted if necessary after a single component has been fully decoded. Additionally, the Y-component can be encoded at a higher bit-rate to naturally provide greater detail.

The compressed texture codestreams are loaded into memory and parsed during the initialization of the virtual environment. When it is time for an object to be rendered the tiles to be decoded are determined from the visible surfaces of the object. The initial goal resolution level is a function of the location of the surface in the field of view.

This JPEG2000 multiresolution functionality replaces the need for mipmaps – the decoded resolution of a tile corresponds to a specific mipmap level. It also partially resolves the image mapping artifacts that can occur when the surface is viewed edge on. Since each tile of a texture image can be independently decoded to a different level, the detail discrepancies between the near and distant parts of the surface can be reduced for anisotropic texture images.

The resolution level is then adjusted based on the decoding cost of reaching that level versus an estimate of the benefit of that level of image detail. The decoding cost of all image textures and frame rendering time must be kept below the target frame rate.

Decoded texture bitmaps can be stored in a frame texture cache to reduce their decoding cost in subsequent frames. If the decoding of a texture is suspended prior to the goal resolution, the decoding is resumed during the next frame's processing. Decoding begins with the partially decoded image and adjusts the cost accordingly. This takes advantage of JPEG2000 progressive by resolution decoding.

3. IMPLEMENTATION DETAILS

This paper discusses an implementation of the fJ2k texture-mapping algorithm integrated in to the OpenGL rendering pipeline. This can easily be extended to any arbitrary three-dimensional world with only minor modifications to existing world parsers and rendering engines. By tightly integrating this algorithm into the texture mapping routines of a rendering engine, additional computational cycles can be saved during frame rendering.

3.1. Cost and Benefit Heuristics

For this experiment, the cost calculation was a simple function for each tile-component-resolution triple. The function estimated the computational cost, C, to decompress the current resolution level of the tile to the goal resolution's dimension. Every tile begins at resolution level 0 before it is first decoded. JPEG2000 permits the individual tile resolutions to vary within an image, therefore each tile in an image can potentially have a different cost to reach specific dimensions. The cost-goal table for each tile is created during the initialization process. Once an image tile has been accessed and decompressed to a new level, the cost of the different goal states is adjusted to reflect this new level.

A threshold cost value, δ, is determined by the goal frame rate. If the total decompression cost of all textures exceeds the threshold, the textures are prioritized based on the estimated benefit they provide to the final scene. The cost formula is represented by:

$$\delta_{frame} \geq \sum_{S} (C(\lambda_{goal}) - C(\lambda_{current})) \qquad \textbf{Eq. 1}$$

where S is the set of tiles to be mapped that frame.

The goal resolution benefit is an estimate of how much detail the final tile-component contributes to the frame. The final rendered dimensions of the goal resolution determine the maximum benefit – the larger a texture image tile appears on the screen, the greater the benefit or contribution to the final frame. The screen ratio, r, of a textured surface is an estimate of the rendered area of the texture as a percentage of screen area:

$$r = \frac{rendered surface \;\; area}{screen \;\; area} \qquad \textbf{Eq. 2}$$

The benefit of the resolution level goal state is then weighted by two primary factors:

- **Angular distance from the viewer's line of site, ϕ.** As described earlier, viewers do not require a great level of detail for objects that appear in the periphery of their field of vision. The viewing angle for key surface vertices is used to determine if the benefit can be reduced. The farther a tile appears from the viewer's line of site, vertically and horizontally, the lower the benefit. The experiments were performed on frames that had a 10 x 3 aspect ratio to simulate a view field of approximately 160°.

- **Surface speed relative to viewer, ω.** If the speed of the surface in ratio to its size is large, the benefit of the resolution level is reduced. This gives an approximation of motion blurring.

The benefits of U- and V- tile-components are decreased slightly to positively weight the luminance (Y-component) of the image, giving more surface detail, but potentially blurring colors. The chrominance components can be decoded to several resolutions lower than the luminance component and still maintain acceptable image quality. The resulting benefit curve has a sharper drop for the UV-components than Y-component.

The resulting benefit, β, is:

$$\beta(\lambda_{tile-component}) = f(r) * g(\phi) * h(\omega) * c(\gamma) \qquad \textbf{Eq. 3}$$

where: $f(r)$ is the screen ratio function,
$g(\phi)$ the foveation weight function,
$h(\omega)$ motion blur weight function

$$c(\gamma) = \begin{cases} 1.00 & \gamma = Y-component \\ 0.75\gamma & = U-, V-components \end{cases}$$

The decreased quality is only temporary, since the codestream contains full quality data that can be used when decoding time allows.

3.2. Frame Cost Optimization

If the total cost to decode all texture image tiles for the frame is greater than the threshold, we must attempt to optimize, but as described in [11], this problem is NP-complete. We implemented a simple ranking of the tiles based on their value to the frame:

$$value = \beta / Cost \text{ for each tile-component} \qquad \textbf{Eq. 4}$$

The parameters of the function used to adjust the tile-component benefit are weighted so that for equal cost tile-components, the value ordering from lowest to highest is approximately:

- U/V-components for tiles outside the fovea.
- U/V-components for tiles on fast moving surfaces.
- Y-component for tiles outside fovea.
- Y-component for tiles on fast moving surfaces.

Once the tile-components have been value-ordered, the resolution of each tile-component is decreased, from lowest value to highest, until the overall cost falls below the threshold. After decreasing the resolution, the tile-component value is recalculated and it is reinserted into the list at the new value rank.

The foveation benefit adjustment function is based on work performed by Chang et al. in [10] and is structured so that texture images in the "fovea" of the viewer have significantly higher (positive) benefit than those on the periphery. If an excessive number of tile-components were required to reduce resolution level on previous frames, the fovea radius is decreased.

3.3. Decoded Texture Image Cache

Objects using this algorithm share a common scratch texture area or cache. Each tile-component is decoded into this data structure which is then bound as the mapping texture. The scratch area from each frame is saved and used to create the textures for the next frame. If the required resolution or visible area has not changed, the image requires no further decoding and is mapped directly from the scratch area. If a higher resolution is necessary, the region already decoded is used as starting point for the decoder thereby reducing the number of computations required to generate the higher detail texture.

4. TESTING METHODS

The experiments used a static virtual environment and a fixed path for the camera flyby. The simulation was run once with traditional mipmapping to determine the maximum frame rate and threshold decompression cost. Texture images for this baseline run were compressed using JPEG2000, but they were decompressed and mipmap hierarchies constructed prior to beginning the animation. The frames generated by this run were used as the "ideally" rendered baseline frames.

Two sets of experimental runs were made; the first was with textures decompressed on the fly, but caching intermediate results. Fully decoded textures and mipmaps were saved in texture memory until no longer needed. This run offered the potential for a high frame rate, but it also carried a high cost in memory. The second experiment set used no cache and decompressed textures from the codestream every frame, but decoded each component to a different level as necessary. This run conserved memory, but resulted in lower frame rates or less detail in some frames.

The frames from the two runs using fJ2k were compared to the baseline frames to identify missing frame detail. The success of the method was determined by comparing the trade-off between several metrics:

Storage Space (disk and texture memory) — impacts the amount of hardware necessary to store and transmit texture images, either over a network or internally within a computer's architecture. This includes the codestream and any data structures necessary to hold the pre-processed header and tile marker information and dynamic data structures used to store decoded image bitmaps.

Frame Rate — The algorithm should adjust the level of detail in texture images so that there is minimal impact to the frame rate and it must maintain interactivity.

Frame Detail — The total error in each frame can be computed after rendering by comparing the partially foveated image with a fully rendered image. To better reflect the impact of on visual quality, the results are compared by component.

5. CONCULSION AND FUTURE WORK

Initial testing has shown foveated JPEG2000 texture mapping is a viable method when a minimum frame rate must be maintained. Using JPEG2000 for image compression gives recognizable surface textures when less than full texture image resolution is necessary to maintain a target frame rate. When caching was used and while maintaining a frame rate within 20% of the ideal, less than 1% of the frames had textures below full quality. This was the result of lower resolution chrominance components and the differences could not be observed visually since they lasted on a few frames.

Running the experiments without caching exercised the fJ2k algorithm more fully. The flyby would have required at least twelve 128x128 pixel textures to be decoded to 64x64 or full resolution every frame without using fJ2k. During these runs, fJ2k was able to maintain a 20% framerate degradation threshold. While most frames required periphery textures to have lower resolution chrominance components, these frames did not have noticeably lower visual quality.

Several potential directions for future work have been identified during the fJ2k implementation. This method lends itself to environments where multiresolution object meshes are used, such as for Internet streaming. Low-resolution image details can be transmitted intermixed with the model's vertex stream and used on the client without the waiting for the entire image codestream to be transmitted. FJ2k implementations that exploit JPEG2000's SNR scalability and ROI feature could lead to additional optimizations.

6. REFERENCES

[1] Tim McReynolds, David Blythe, Brad Grantham, and Scott Nelson; "Programming with OpenGL: Advanced Techniques," Course 17 notes at SIGGRAPH '98, 1998.

[2] Dan McGabe and John Brothers, "DirectX 6 Texture Map Compression," *Game Developer Magazine,* vol. 5, no. 8, pp. 42-46, August 1998.

[3] "FTX1™ Texture Compression Technical Specifications" 3DFX Interactive Inc., San Jose CA, April 1999. http://www-dev.3dfx.com/fxt1/

[4] Andrew C. Beers, Maneesh Agrawala, and Navin Chaddha. "Rendering From Compressed Textures". *SIGGRAPH 96 Conference Proceedings.* pp 373-378, 1996.

[5] ISO/IEC JTC 1/SC 29/WG 1, *ISO/IEC/ FCD 15444-1: Information Technology – JPEG 2000 Image Coding System: Core Coding System [WG 1 N 1646]*, March 2000,

[6] Charilaos Christopoulos, Athanassios Skodras and Touradj Ebrahimi; "The JPEG2000 Still Image Coding System: An Overview;" *IEEE Transactions on Consumer Electronics,* Vol 46, No.4, pp 1103-1127, November 2000.

[7] Diego Santa Cruz, Touradj Ebrahimi, Joel Askelöf, Mathias Larsson and Charilaos Cristopoulos, JPEG 2000 Still Image Coding Versus Other Standards. *Proceedings of SPIE – Applications of Digital Image Processing XXIII,* volume 4115, Oct 2000, Bellingham, WA

[8] Donald D. Hoffman, *Visual Intelligence: How We Create What We See*, W. W. Norton, New York, NY, 1998

[9] G. J. F. Smets and K. J. Overbeeke. Trade-off Between Resolution and Interactivity in Spatial Task Performance. *IEEE Computer Graphics and Applications,* pp 46-51, September 1995

[10] E.C. Chang and S. Mallat and C. Yap, "Wavelet Foveation", *Journal of Applied and Computational Harmonic Analysis*, Feb 2000

[11] Thomas A. Funkhouser and Carlo H. S'equin. "Adaptive Display Algorithm For Interactive Frame Rates During Visualization Of Complex Virtual Environments." In *Computer Graphics (SIGGRAPH '93 Proceedings)*, James T. Kajiya, editor, volume 27, pages 247--254, August 1993.