Strategies for Foveated Compression and Transmission

Brian Funt, Hoamiao Jiang**, Carlin Vieri**, Sylvain Vignaud**, Behnam Bastani***

***Simon Fraser University, Vancouver**

****Google, Mountain View**

Abstract

This paper address three different strategies (F-JPEG, F-DSC and F-YUV) for foveated image compression plus methods for the transmission of foveated image data. In place of simple subsampling for foveation, the proposed foveated compression strategies use varying compression ratios--- a low compression ratio centrally, and progressively increasing compression ratios towards the periphery.

Author Keywords

Virtual Reality; Foveated Compression; Foveated Transmission.

1. Introduction

We first discuss three different strategies for foveated image compression and then address methods for the transmission of foveated image data. Foveation is a well established technique for reducing graphics rendering times for virtual reality applications [1] and for compression of regular image data [2]. In all cases, the strategy is to exploit the fact that human vision is more acute centrally (foveal region) and less acute peripherally by rendering or representing more detail around the fixation point and less towards the periphery. In the context of CG rendering, the most common foveation approach is simply to render fewer pixels in the periphery than centrally and then to upsample the reduced resolution data by interpolating pixel values for the intermediate locations. This technique has been shown to work quite well; however, because of the loss of the high-frequency spatial information it does sometimes lead to what is referred to as the "tunnel-vision effect" [1].

In place of subsampling, the foveated compression strategies proposed here, named F-JPEG, F-DSC and F-YUV, are all based on standard compression strategies, namely, JPEG, DSC and an extension of YUV 4:2:0 subsampling. The key idea is to create an image using a low compression ratio centrally, and progressively increasing compression ratios towards the periphery. In principle, any compression method that compresses the image locally can be applied to foveated compression. With F-JPEG this is accomplished by varying the JPEG's quality setting. Previously, Rosenbaum and Schumann [2] showed that JPEG2000 can be used in a similar way but here we apply JPEG. In the case of F-DSC, the DSC standard unfortunately does not provide a mechanism for varying the compression ratio, which is fixed at 2.8:1. We overcome this limitation by packing neighboring 8-bit pixel pairs into single 16-bit pixels and use DSC to compress the result. For F-YUV, the compression ratio and resulting image quality are varied by sampling the Y, U and V values differentially and at increasingly large spacings.

We also describe a foveated transmission method that packages foveated data to reduce the transmission cost between the SoC and display. The key idea of foveated transmission is to move several simple operations (e.g. upscaling) to the display TCON so that data transmission on the PHY can be greatly reduced.

2. Foveated JPEG

In the context of image compression, many methods of foveation are based on a downsampling the original image data towards the periphery. F-JPEG replaces the downsampling with differential compression using JPEG. Regions of different resolution are replaced by regions compressed at different JPEG quality settings. One advantage of using JPEG over regular downsampling-upsampling is that it specifically deals with differential compression of the colour information, compressing the blues more than the reds and so on. An example of F-JPEG is shown in Figure 1. Our implementation of F-JPEG is based on individually compressing blocks of size 128x128. It runs at 20 frames per second with a compression ratio of approximately 50:1.

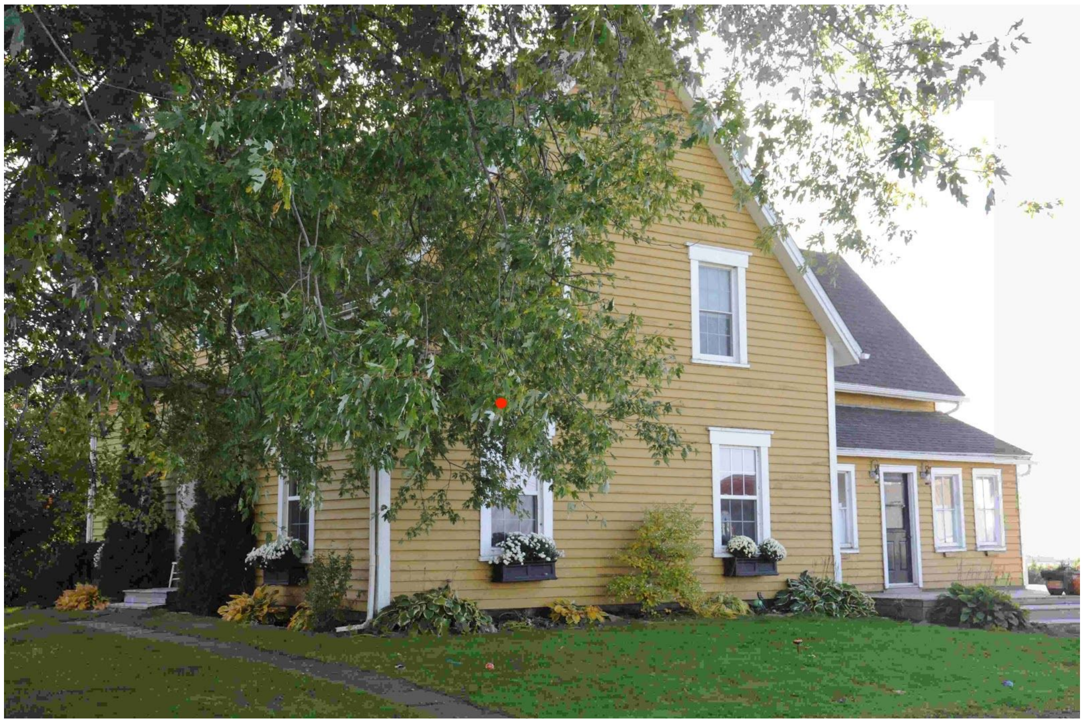

Figure 1. The original 4256x2832 24bpp image is foveated using a central area (surrounding the red dot) of 1138x759 with quality=50, surrounded by a 2392x1451 area of quality=20, surrounded by a 3328x2066 area of quality=11, with the far periphery at quality=5. When fixating on the red dot with the image at full screen size, the color problems at the far right and far left of the house and the virtually colourless branches up in the top left corner are not noticeable. Similarly, the overly green color and reduced texture of the grass in the lower right are not noticeable.

3. Foveated Display Stream Compression

Since display stream compression (DSC) is a standard for the transmission of compressed image data between DisplayPort transceiver and receivers that has a hardware implementation [3], we investigated how it might be modified for foveated compression. It also has the advantage of being an in-line compression method, which eliminates the expense of a buffer to hold the image. The primary difficulty in using DSC as it stands is that there is no simple way to vary the bitrate. DSC always compresses 8-bit-per-channel image data by a factor of approximately 2.81. It is possible, however, to get a little control over the compression ratio using a 16-bit input (DSC setting BITS-PER-COMPONENT = 16) and an 8-bit output (BITS-PER-PIXEL = 8). In this case, the compression ratio is 5.65:1. For F-DSC, we make use of this by employing the following strategy: (1) Subsample the input image into two images, one of even-numbered rows, the other of odd-numbered rows; (2) Pack the corresponding 2 sets of 3 RGB, 8-bit values from each image pixel into 3, 16-bit numbers assigning the 8 bits to alternating bit positions of the 16; (3) DSC compress and uncompress the resulting 16-bit image; (4) Unpack the 16-bit values into 8-bit per channel pixel values and interleave them into alternating rows to create the output image. Clearly, it would be better just to be able to modify the DSC standard so that DSC would provided variable compression ratios, but the above work-around produces quite reasonable results.

An example of F-DSC's output is shown in Figure 2. Since only two compression ratios are available, the F-DSC result is based on a relatively wide foveal region covering 50% of the image width and 50% of its height. The total foveal area is compressed using standard DSC and the peripheral area is compressed using 16-8 packing followed by standard DSC.

4. Foveated In-line YUV

F-YUV is another in-line compression strategy on based on extending the basic idea of the YUV 4:2:0 representation [4]. YUV 4:2:0 is based on sampling the UV color components at ¼ the rate of the Y luminance component. We extend the 4:2:0 representation to even lower sampling rates for the U and V components in order to vary the compression ratio for foveated imaging. The resulting subsampled data is then interpolated back to the original resolution. Our experiments showed that subsampling the UV components quite severely, while subsampling the Y component less severely maintained enough of the high spatial frequency information that it avoided the tunnel-vision effect. Figure 3 shows an example and expanded views of two subregions. F-YUV runs in real time.

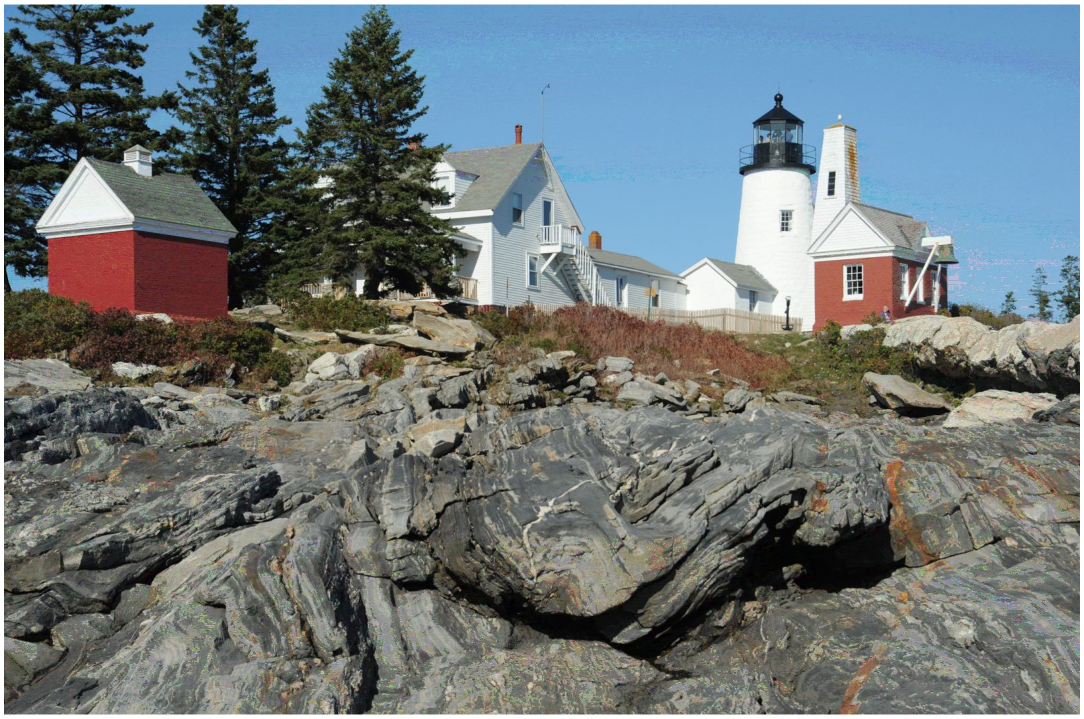

Figure 2. F-DSC foveated image resulting from regular DSC of the fovea and DSC 16-8 compression of the periphery. The overall compression ratio is 4.5:1. Fixation point is at the image center. Minor artefacts such as the sky color at the top of the image and the small pink areas to the right of the rightmost building are not noticeable when viewed as intended (large screen and fixating on center).

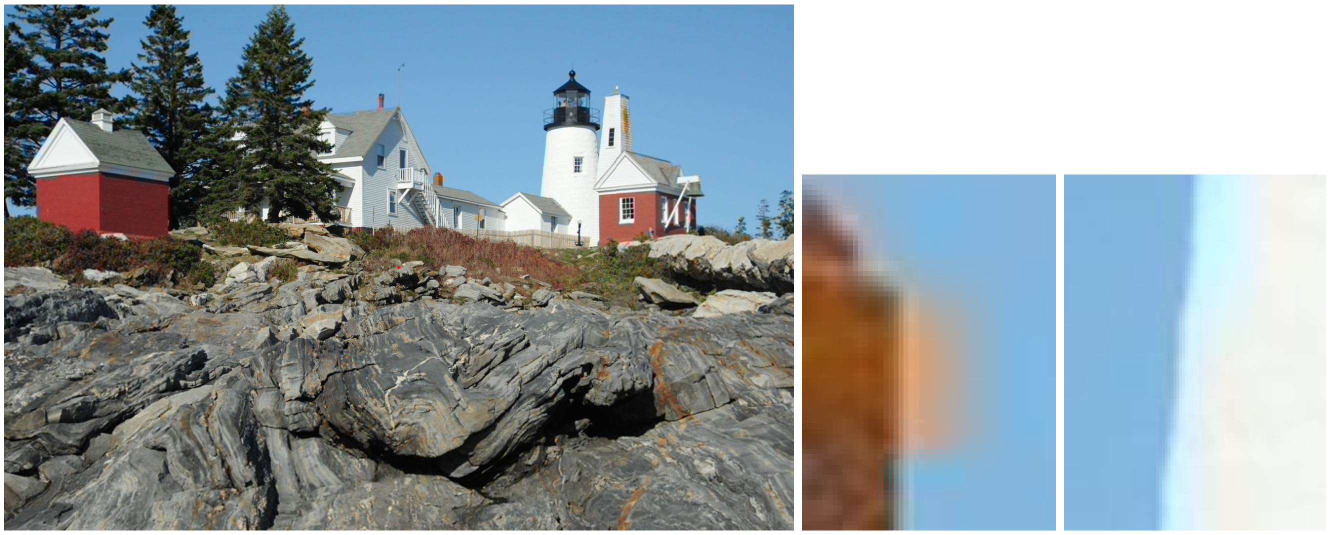

Figure 3. Lighthouse foveated F-YUV image at a compression ratio of 10.2:1. Fixation point is at the image centre (marked by a red dot that is hard to see). Foveal full-resolution region is the central 10%. Concentrically surrounding regions are created with (i) Y component subsampled every 2 pixels (resulting size of ¼) and its UV components subsampled every 6 pixels in each direction; (ii) Y subsampled every 3 pixels and its UV components subsampled every 19 pixels; (iii) Y subsampled every 5 pixels and its UV components subsampled every 25 pixels. The smaller images to the right are expanded views of the lower chimney and lighthouse tower showing some color bleeding artefacts caused by the severe subsampling of the U and V components. These artefacts, however, are not noticeable when viewed as intended.

5. Transmission of Foveated Images

In a standalone HMD system, a non-trivial, but often ignored, source of power consumption is that used for data transmission over some PHYs between the SoC and display module. Based on a conservative estimate, each payload bit takes around 10 pJ to be transmitted through a MIPI PHY to the display. This means, even for a system with relatively low resolution QHD resolution (2560x1440 pixels), transmitting the bits requires roughly 50 mW, and represents a noticeable portion of the overall power consumption of the display system. It of course increases proportionally with increasing display resolution and frame rate. As the data rate increases, more MIPI lanes need to be added to the system, which may bring further constraints such as to the overall mechanical design of a headset.

To save power and bandwidth, foveated transmission aims to transmit less data to the display by moving the upscaling and blending operations to the display side and transmitting only the foveated data. For example, in a traditional foveated rendering scheme, high acuity and low acuity regions are generated by the renderer. Instead of upscaling and blending the two regions together on the SoC and sending the full resolution content, the regions are multiplexed together and sent to the display for further processing (upscaling, blending, etc.), which greatly reduces the amount of transmitted data.

There are multiple ways to form the foveated data package. One simple and straightforward approach is putting high acuity on top of low acuity regions. This method enables the FPGA to start scanout data to display once it starts to receive the low acuity region, no matter what position the high acuity region is in the frame. The decoding part of this method is simple and can be easily implemented in display TCON. However, in most cases, this method requires a relative big memory buffer on TCON side. Other multiplexing methods, such as line-by-line or slice-by-slice interlacing, have also be developed. Different foveated data schemes imply different buffer and latency requirements for the display TCON design. A theoretical analyses shows that sliced interlacing leads to the minimum buffer size and shortest latency. Proper adjustment of the front/back porch sizes also plays an important role in minimizing buffer size and system latency.

Besides buffer and latency issues, special bit manipulation methods may also be required to maintain visual quality when DSC is used. For example, it is generally best to align the borders between regions of different acuity with slice borders. This alignment avoids compression errors spreading across different acuity regions. Besides, we need to add enough duplication to metadata header line to ensure the metadata gets transmitted to display accurately.

Considering that there is limited memory space on a TCON board, it would be hard for the display TCON to handle lens distortion and chromatic aberration corrections. Thus, it is preferred to handle distortion correction on the GPU in a foveated manner. In foveated lens distortion, we find it important to render a slightly larger field of view for each acuity region to avoid black borders along high acuity regions after distortion correction.

6. Conclusion

Three foveated imaging strategies (F-JPEG, F-DSC and F-YUV) based on differential compression ratios have been shown to work well. Both F-DSC and F-YUV can be implemented as inline methods. Foveated transmission is a way of reducing the bit rate through a MIPI PHY, which is important for reducing power consumption on mobile devices.

7. References

[1] A. Patney, J. Kim, M. Salvi, A. Kaplanyan, C. Wyman, N. Benty, A. Lefohn, and D. Luebke. “Perceptually-based foveated virtual reality,” SIGGRAPH 2016, July 2016.

[2] R. Rosenbaum and H. Schumann, “On-demand foveation for Motion-JPEG2000 encoded imagery,” Proc. ISCIT 2007 International Symposium on Communications and Information Technologies, 2007.

[3] Stolitzka, Dale. "Developing requirements for a visually lossless display stream coding system open standard." SMPTE Motion Imaging Journal 124.3 (2015): 59-65.

[4] S. Winkler, C. J. van den Branden Lambrecht, and M. Kunt . "Vision and Video: Models and Applications". In Christian J. van den Branden Lambrecht. Vision models and applications to image and video processing. Springer. p. 209, 2009.