International Conference on Artificial Reality and Telexistence
Eurographics Symposium on Virtual Environments (2016)
D. Reiners, D. Iwai, and F. Steinicke (Editors)

View Dependent Tone Mapping of HDR Panoramas for Head Mounted Displays

S. Cutchin and Y. Li

Computer Science Department
Boise State University, Boise, Idaho, USA

Abstract

Head mounted display(HMD)s are characterized by relatively low resolution and low dynamic range. These limitations significantly reduce the visual quality of photorealistic captures on such displays. In this paper we present an interactive view dependent tone mapping technique for viewing up to 16K wide high dynamic range panoramas on HMDs via view-adjusted mapping function stored in separate texture file. We define this technique as ToneTexture. The use of a view adjusted tone mapping allows for expansion of the perceived color space available to the end user. This yields an improved visual appearance of both high dynamic range panoramas and low dynamic range panoramas on such displays. We present comparisons of the results produced by this technique against Reinhard tone mapping operators. Demonstration systems are available for WebGL and head mounted displays such as Oculus Rift and GearVR.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms

1. Introduction

Current head mounted displays (HMD) have a significant screen door effect due to the closeness from the display screen to the viewer's eye and the insufficient pixel density of the device [ASS*11] [KUDC07]. This combined with its low dynamic range display ability leads to a notably reduced visual quality under the device resolution [SCK*13]. Given the fact that the screen resolution can't be increased easily, an alternative avenue for improving visual quality is to introduce high dynamic range (HDR) imagery to provide better image quality in terms of color space. However, most existing display monitors and head mounted displays (HMDs) have a limited color range that only supports low dynamic range (LDR) imagery [Blo13]. Tone mapping is the operation used to convert HDR images to LDR images to be shown on these devices. Because of the reduction in dynamic range, tone mapping inevitably causes information loss [MNP97, YW13, LH12, XF11].

While viewing a panorama via an HMD, the user only views a small region of the entire panorama at a particular moment. This means that a large portion of the panorama doesn't need to be rendered. Conventionally, when a user tries to view an HDR panorama whose range of intensity levels could be on the order of 10,000 to 1 [RHD*10], a tone mapping operator (TMO) is applied to the whole image. The resulting LDR panorama will have a much narrower global intensity range, normally from 0 to 255. This means that the user, who is only able to see a small region out of the whole panorama, will end up looking at an LDR image of the current viewport with intensity range even narrower than 256. Therefore visual quality for the end user could be significantly improved by only applying tone mapping on the HDR pixels within the current viewable rectangle of the whole panorama. In this way, every small region the user views could fully utilize the available 256 shades of intensity.

There have been many works relating to localized TMOs over the past decade [DD02, SJB10, FLW02]. Many of them focus on optimizing viewing quality in local regions of high contrast scenes. The mapping function used in these local operators varies spatially, depending on the neighborhood of the pixel [DD02]. In [SJB10], the authors presented a window-based tone mapping method that uses a linear function to constrain the tone reproduction inside each window in order to naturally suppress strong edges while retain weak ones. Then the HDR image is tone mapped by solving an image-level optimization problem that integrates all windows-based constraints. [FLW02] introduced a conceptually simple, computationally efficient and robust method by attenuating large gradients and then constructing the LDR image by solving a Poisson equation on the modified gradient field. It is also generally noticed that local operators, which reproduce the tonal values in a spatially variant manner, perform more satisfactorily than global operators [SJB10]. Even though these methods all generate LDR images with good visual quality while preserving contrast details, they have fundamental limitations for viewing HDR panoramas via HMDs. Firstly, they all generate one LDR images as the output. No matter how good

DOI: 10.2312/egve.20161431

LDR images are, they cannot exploit as many level of intensities as possible when user is only able to see a small portion of the whole image. Moreover, computationally efficient as these methods are, they are still too expensive to run in real time. That is to say, even when applied these methods just to the small viewable region, the computational cost will severely jeopardize the application's frame rate.

In this paper, we present a novel technique for the view-dependent tone mapping of high-resolution(16K) photographic HDR panoramas we are calling ToneTexture. The ToneTexture technique allows for the dynamic tone-mapping of an HDR panorama customized to the view direction of the user. The ToneTexture method allows for the application of a TMO customized to the panorama content visible to a user based on their current view direction while also supporting the use of global and regional panorama image intensity details to maintain a consistent coloring of the LDR panorama in whatever direction the user may look. ToneTexture method can utilize a variety of TMOs as determined to most optimally work for selected content. Our ToneTexture method combines the use of third order Bernstein mapping function with window-based image processing. The method is conceptually simple yet efficient, flexible and robust. We associate a window region to every possible view angle of the entire panorama. To achieve real-time viewing performance, our method puts most computational operations in pre-processing phase. This pre-processing phase consists of two passes. The first pass is window-based pixel intensity statistical information gathering, including minimum, maximum and average intensity values. Based on the statistical attributes, the second pass is window-based Bernstein coefficients computation. The third order Bernstein mapping function leaves us enough control and flexibility to handle windows of different intensity distribution. The results - third order Bernstein coefficients - is then stored in an extra texture file that we refer as ToneTexture. Because the computation is totally window-based, both passes can be easily parallelized. In order to preserve as much HDR details as possible, we map each pixel of the large-sized panorama to a view angle. Since each view angle is associated with one window, the number of windows developed will be the same as the number of pixels. Therefore ToneTexture would have the same size of the original panorama. At runtime, the only extra cost is looking up coefficients from ToneTexture and calculating a third order polynomial mapping function on the pixels of current viewable region.

The same idea of window-based view-dependent tone mapping is adopted in [Yu], where the author applied log-average luminance adjustment on the viewable windows and stored the calculated results in a look-up table. Comparatively, ToneTexture advances in several ways. First, ToneTexture is targeting at compressing large-sized HDR panoramas (up to 16K wide) with real time viewing performance. Second, ToneTexture is more flexible and robust. While the nature of equirectangular projection causes severe distortion nears the poles in [Yu], ToneTexture copes with this problem in pre-processing phase via proper transformation. Last and most important, ToneTexture allows for easy access to tone editing. This is the potential impact of this operator and the focus of future work.

To demonstrate the effectiveness of ToneTexture operator, we also carry out an objective tone mapped image assessment, Tone Mapped image Quality Index(TMQI) [YW13] to compare our method against the well-known Reinhard TMO applied either globally or locally [RSSF02]. Because TMQI is designed to evaluate gray-scale images only, we apply this assessment method to each color channel independently and then combine them according to human vision system [Rec11].

2. Methodology

Our work started with a simplistic two pass algorithm that keeps track of a few statistical parameters such as minimum or maximum intensity as the pixels are rendered by a fragment shader and adjust these pixels during the barrel filter pass accordingly. However, this method suffers from the drawback of losing the HDR intensities prior to tone mapping operation unless we render the image in HDR framebuffers which are not readily available on all devices currently. In addition, computing the minimum, maximum, and average of pixel intensities during the rendering process adds extra computation to our fragment shader. Thus this algorithm may not be practical for real time rendering.

Our solution calculates tone mapping parameters in pre-processing time and storing those values in an extra texture file we call ToneTexture. In order to achieve flexibility, we assign one unique 3rd order Bernstein-Base curve to each region based on the intensity distribution. The statistics are used to help us identify different types of intensity histogram. Thus in stead of directly using statistics like minimum and maximum, we store Bernstein coefficients in ToneTexture. When rendering the panorama, we take these Bernstein coefficients and calculate new intensity using the curve in the fragment shader.

Our target panorama image is 16K x 8K exr file which uses 16-bit half precision floating point numbers to store RGB and transparency information. With minimum precision of 2^{-10} in the range of $[0,2]$, we are facing an imagery of far more than 256 degrees of intensity. The following parts in this section elaborate details in viewable region identification, parallel computation, ToneTexture generation and how we use Bernstein-Base curve for tone mapping.

2.1. View Window Identification

Since the location of the view point is fixed at the center of the panoramic sphere, view orientation can be identified solely by the longitude and latitude of viewable window center. Suppose (s,t) are longitude and latitude of a view direction respectively, with their origin at the upper left corner of the unwrapped panorama. The viewable pixels from the fixed viewing point are bounded by a rectangular window whose center locates at (s,t). We ignore perspective distortion for simplicity at this point. Note that there's an one-to-one mapping between a (s,t) pair and a specific panoramic pixel coordinate within the panorama.

For any (s,t), we can now compute the ToneMapping attributes with this mapping approach. Applying this to each pixel within the Panoramic texture by using the pixel coordinate as the view direction (s,t) coordinate, we can compute the statistical properties for the all possible viewing rectangular window. The results including the minimum, maximum, and average intensity values are

stored in an intermediate texture file. In the simplest version of our method, this temporary texture file will be equivalent to the original panorama in size. Extensions allow for a reduction in the size of texture file as well as the size of the final ToneTexutre at the cost of some HDR accuracy. The basic process at this step is shown in Figure 1.

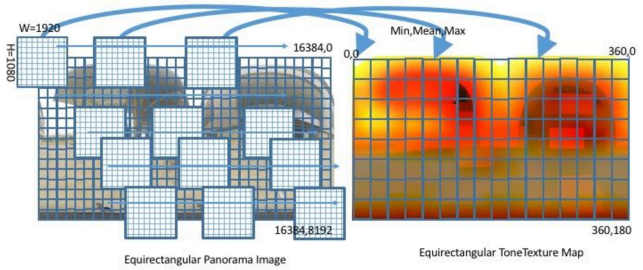

Figure 1: *Basic process for generating intermediate statistic texture. Maximum, minimum and average values are stored as RGB respectively.*

After locating viewable windows, our next task is to determine the its size in pixels. This is defined by fixing the Field-of-View angle instead of actual pixel resolution of the devicce. In this way, for various devices with different screen resolution, the exact window size in terms of pixel resolution is converted through a simple spherical coordinate projection to equirectangular coordinates. This approach grants us more flexibility across different devices.

Notably, it is more complicated on edges of the panorama where the viewing rectangle exceed the unwrapped panoramic texture borders. In our implementation, we extend the original panorama in equirectangular coordinate before we calculate the statistical attributes. As a result, the size of the panorama used to produce statistical texture is $(n+w)*(n+h)$, where n is the size of the original panorama and w,h is the width and height of the viewing rectangle. Figure 2 illustrates an extension example. In this way, our operator will not have to worry about distortion near the panoramic sphere poles.

Figure 2: *Expanding Panorama Texture for Parallel Conversion*

2.2. Dynamic Optimization

In our solution, computation for ToneTexture can be divided into two passes - calculating statistical properties and determining Bernstein coefficient based on the intensity histogram for every viewing region.

The above technique has a limited performance when running in serial. Traversing the large-sized panorama for every viewing region and inspecting all pixels within this $w*h$ region of interest lead to redundant access to the same pixel. We are not able to greatly improve the performance when searching for the minimum and maximum intensities within each window, but we manage to improve the algorithm for computing average intensity and successfully reduce the runtime by one order. The primitive algorithm is $O(n^2*w*h)$. By altering the traversal order and storing intermediary results in a temporary texture, the operation can be reduced to $O(n*(n+h+w))$. This algorithm can be easily implemented in parallel on a GPU.

The accelerated algorithm consists of two steps:

1. Traverse the panorama once by column. Calculate the average value in the first h rows. For the rest rows, use the following equation to compute new average.

$$I_{average\_col\_new} = I_{average\_col\_old} - \frac{I_{first}}{h} + \frac{I_{new}}{h}, \quad (1)$$

   Store these results in a temporary average texture with the width of the expanded panorama and the height of the original panorama. This pass takes $O(n*(n+h))$ steps. We parallelize this pass by column.
2. Traverse the intermediary average texture by row. Calculate the average intensity in the first w columns. Use Equation 2 to calculate the rest columns.

$$I_{average\_row\_new} = I_{average\_row\_old} - \frac{I_{first}}{w} + \frac{I_{new}}{w}, \quad (2)$$

   Store results in intermediate statistics texture. This pass takes $O(n*(n+w))$ steps to complete. Parallelization in row major is adopted to reduce actual calculation time.

The steps are shown in Figure 3, where the average pixel intensities within the viewable window at viewing direction $(0,0)$ are computed.

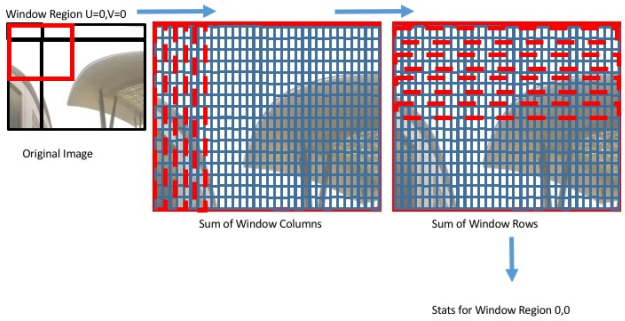

Figure 3: *Parallel Image Statistics Generation*

Unfortunately, unlike computing the average, searching for minimum, maximum intensity and identification of histogram in the second phase still have $O(n^2*w*h)$ operations. We implement these steps in 2D parallelization to improve efficiency.

2.3. Histogram and Bernstein Curve

Based on the statistical attributes we compute in previous phase, we can build the histogram of pixel intensities for each viewing rectangle in the entire panorama. we classify these histograms into 4 categories:

- **Ascending**: The overall histogram tends to ascend as intensity grows. This is the region where most of its pixels are bright.
- **Descending**:The overall histogram tends to descend as intensity grows. This is the region where most of its pixels are dim.
- **Peak**: The pixels in this region is evenly distributed and most pixels locates at the middle of intensity range.
- **Trough**: Pixels in this region form a trough in the histogram. Most pixels are either in the bright range or the dim range. As a result, this region has both bright spots and dim spots.

To identify a view window's histogram category, a preliminary approach is to use minimum and maximum to generate a three-bin intensity histogram and categorize it based on the relative heights of the bins. However, this approach could be incorrect. It's true that three bins are enough to distinguish four different distribution types, but the problem is that some pixels with extreme intensity can severely jeopardize the effectiveness. The **Peak** example in Figure 4 shows such a case where a simple three-bin intensity histogram of this window could be falsely categorized as **Descending**. Our solution is to integrate average value when generating the three-bin histogram so that we only count pixels whose intensity falls in the average-centered interval. In this way, we can omit pixels of extreme intensity reasonably while preserve efficiency.

To customize the mapping curve we use in fragment shader, we need to determine what we want to achieve after tone mapping. Our goal is to create one tone mapping curve exclusive to one view region that can equalize this region's original histogram and stretch it as wide as possible to exploit more intensity range. For each of the 4 histogram categories, its final output can be described as follow:

- **Ascending**: The whole region needs to be dimmed down and the brighter the pixels are, the more they will be dimmed down.
- **Descending**:The whole region needs to be brighten up and the dimmer the pixels are, the more they will be brighten up.
- **Peak**: For this type of region, we only need to stretch the histogram reasonably so that its center approximates the center of available intensity range.
- **Trough**: We want to scale the dim peak up and the bright peak down at the same time so that the resulting histogram are equally spread across available low dynamic range.

After we define the final histogram we want to produce, we can have a primary outline of the curve for each type of histogram. We choose Bernstein polynomial because we can easily control different parts of the curve to form its shape by the coefficients. Since ToneTexture is basically another image file and we can store up to four parameters as RGBA values in exr format. Thus we use third order Bernstein curve that grants us the most controls under the limit of the maximum number of coefficients we can store. Equation 3 is the general form of a third order Bernstein polynomial. A,B,C,D are called Bernstein coefficients and determine what the curve would be in a clear pattern: A and D control left and right ends respectively whereas B and C determine the shape of what lies between the center of the curve and left or right end.

$$B_3(x) = A(1-x)^3 + B(3x(1-x)^2) + C(3x^2(1-x)) + Dx^3, \quad (3)$$

We need four equations to calculate the values of A,B,C and D in linear equation system. In the cases of **Ascending** and **Descending**, statistical attributes minimum, maximum and average can contribute to the following 3 equations:

$$B_3(Min) = Min, \quad (4)$$

$$B_3(Max) = \alpha Max, \quad (5)$$

$$B_3(Median) = \frac{\alpha}{\alpha+1}(Min+Max), \quad (6)$$

α is a user defined adaptive equalization factor which helps to equalize the final histogram in the range of $[0,1.0]$ after exposure adjustment. To be more precise, $B_3(Max)$ will be scaled to approximate to 1.0, and $B_3(Median)$ will approximate to the scaled arithmetic mean of minimum and maximum. Equation 6 also determines whether the calculated Bernstein function is concave or convex. The last equation is different in these two cases. We use the first order derivative of the Bernstein function to constrain the curve's shape. Equation 7 and Equation 8 are applied in **Ascending** and **Descending** respectively.

$$B_3'(Min) = 0, \quad (7)$$

$$B_3'(Max) = 0, \quad (8)$$

Unlike above cases, **Peak** and **Trough** can use the same set of equations. We still use Equation 4 and Equation 5 to determine the ends of Bernstein curve. Then we divide the pixels into two parts around average and traverse the view window again to find new average value of each part, namely $average\_left$ and $average\_right$. We can then find two more equations using the newly computed average values:

$$B_3(Median\_left) = \frac{3\alpha}{4\alpha+2}(Min+Median), \quad (9)$$

$$B_3(Median\_right) = \frac{3\alpha}{2\alpha+4}(Median+Max), \quad (10)$$

Using different sets of four equations for different categories, we can create a linear equation system and calculate A,B,C and D by solving the system with various solutions. Our current implementation uses variable elimination that runs in CUDA.

Based on above algorithm, we can calculate a unique set of Bernstein coefficients for each viewable region and store the result in an exr file where coefficients are stored as RGBA values.

Figure 4 shows the 4 base histogram types, their target histogram and the curve use to achieve the result. All data comes from actual regions in the testing Redwood panorama.

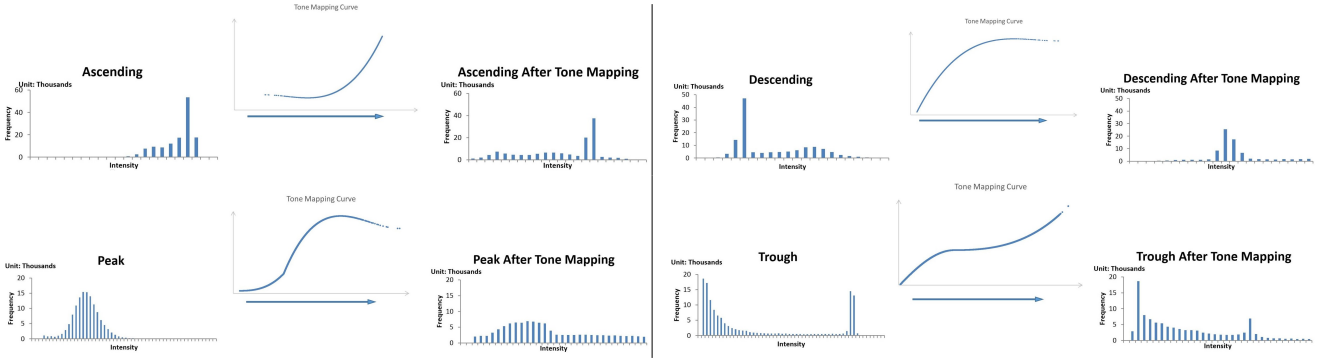

Figure 4: *Histogram processing via tone mapping curves. Frequency unit is thousand.*

3. Testing Environment and Implementation

Our current work consists of two major component: parallel ToneTexture generation program implemented in CUDA and HDR panorama viewer implemented in OpenGL and WebGL with dynamic tone mapping. The OpenGL implementation supports both stereoscopic viewing and monoscopic viewing while the WebGL solution is limited to non-stereo panorama. The OpenGL solution supports 16K panoramas while the WebGL solution is presently limited to down sampled 4k panoramas. Hardware systems consist of standard Dell Workstations for development and testing with Oculus Rift headsets, Samsung GearVR head mounted displays and NVIDIA Quadro K620s.

3.1. ToneTexture Generation

The ToneTexture generation program is used for most of the preprocessing computation. There are two passes involved: the statistical attributes computation and Bernstein coefficients calculation.

Figure 5 shows the architecture of the ToneTexture generation program. Using an HDR panorama as input, a serial process locates the neighboring areas around edges and corners and performs rotation and translation to properly enlarge the original panorama.

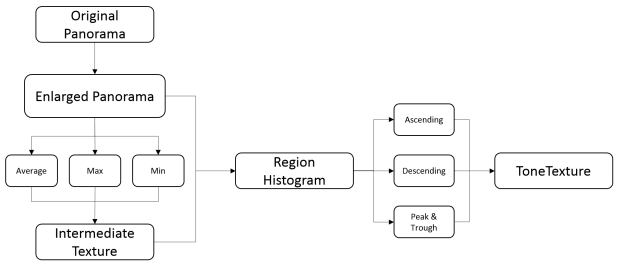

Figure 5: *ToneTexture Generation Architecture*

Then the program traverse the newly created enlarged panorama twice to compute the statistical attributes for every viewing window. The first traversal generates statistical values of each region by column major and the second one runs by row major on previous results. Final values are stored in an intermediate texture file for later use. Since every computation in this phase is completely independent, it runs in parallel for efficiency reason.

After we have the intermediate statistical texture, we can use the values to generate three-bin histograms of pixel intensities for each region. Due to the need of traversing every pixel in every window, redundant access to the enlarged panorama is inevitable. Once the histogram is generated, we immediately categorize them and solve linear equation system to get Bernstein coefficients as specified in Section 2.3. We implement full parallelization to produce the final ToneTexture. Since the distribution of intensity in different regions are rather totally content-related, the performance suffers from branch divergence.

3.2. Panorama Viewer

Even though implemented in different platforms, both OpenGL and WebGL panorama viewer share the same architecture and only differ in the final rendering stage, as shown in Figure 6.

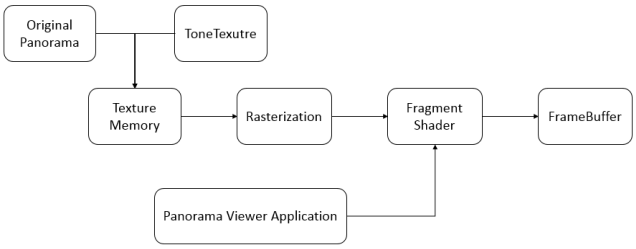

Figure 6: *Panorama Viewer Architecture in final rendering stage*

In panorama viewer, we load both the original panorama and ToneTexture into texture memory and apply rasterization. Our view dependent tone mapping operator is implemented in the fragment shader. The viewer application updates the viewing center coordinates to the shader to keep track of the viewing direction. To avoid unnecessary operations, only viewable pixels will be mapped to new intensity and those currently out of user's sight will be completely ignored. The shader then looks up Bernstein coefficient from ToneTexture using the viewing center coordinate as index and compute for new coloring values. Exposure adjustment takes place after tone mapping is performed.

4. Objective Experimental Results

In this section, we use Tone Mapped image Quality Index [YW13] to objectively access our ToneTexture TMO against modified Reinhard TMO [RSSF02] which can be expressed in Equation 11:

$$L_d(x,y) = \frac{L(x,y)(1+\frac{L(x,y)}{L^2_{white}(x,y)})}{1+L(x,y)} \tag{11}$$

where L_{white} is the smallest luminance that will be mapped to pure white. This parameter grants modified Reinhard method to work in local region. So we'll compare our technique against a globalized modified Reinhard operator whose L_{white} is taken from the entire panorama as well as a localized modified Reinhard operator whose L_{white} is taken from current view window.

All test cases are performed on HDR panorama of size 4K. The test images contain three windows from one HDR panorama called Redwood and two windows from two other HDR panoramas, namely 'Grace Cathedral' and 'Loc2_1'. We perform three TMO on them. With more than 9000 distinguishing intensities, the panoramas risk losing information when linearly mapped into LDR. In other words, all panoramas are in HDR and need proper tone mapping operator to preserve good visual quality.

The objective tone mapped image assessment method - TMQI uses original HDR image as the reference and scores on two competing factors: structural fidelity(S) and statistical naturalness(N). Structural fidelity is based on the assumption that the main purpose of vision is to extract structural information from visual scene [WBSS04]. Statistical naturalness quantifies subjective idea by focusing on brightness and contrast [CS05]. An overall quality score(Q) is given by Equation 12:

$$Q = aS^{\alpha} + (1-a)N^{\beta} \tag{12}$$

where a, α and β are user defined adaptive parameter and in our experiment we use the default setup where $a = 0.8012$, $\alpha = 0.3046$ and $\beta = 0.7088$. Note that TMQI is limited to evaluate gray-scale images only. To assess our TMO, we apply it to each color channel independently and then combine the results based on human visual system. Equation 13 is used to get the final score:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \tag{13}$$

4.1. Experiment Result

The result is shown in Figure 7 and 8. The first three regions are chosen from the same HDR panorama with different intensity distribution and the rest two regions are from two different HDR panoramas. In general, global Reinhard looks pale and dim whereas the localized version appears to be brighter and with more contrast. The result of our technique in SpotB is dimmer than the others while the results SpotA and SpotC appears to be brighter. The ToneTexture results in all regions appears to have more contrast than the other two technique. The histogram suggests ToneTexture does better histogram equalization due to wider intensity range usage. Meanwhile, our technique also preserves the overall histogram outline shape of original HDR image. The three small peaks in the original HDR histogram is well preserved in SpotA. Our technique does the best in SpotB since the other two methods scale the second peak to the right edge of the histogram so that these pixels appears washed out. This is why the tree at the right bottom corner in the back is barely visible. Notably, in Grace, pixels in right up corner have extremely high intensity. As a result, the window in the scene is washed out in both Reinhard methods whereas ToneTexture presents more details. However, ToneTexutre does wash out some details in the brighter area in SpotC. This alone with the histogram indicate that our method brightens dim pixels too much. A future improvement can be done by finding a better adaptive equalization factor than that we used in the tests.

Region	TMO	Structural Fidelity	Statistical Naturalness	TMQI Score
SpotA	GR	0.9385	0.4002	0.8895
	LR	0.9433	0.5084	0.9094
	TT	0.9410	0.4725	0.8935
SpotB	GR	0.8418	0.0241	0.7686
	LR	0.8147	0.0158	0.7631
	TT	0.8559	0.5867	0.8997
SpotC	GR	0.9098	0.8116	0.9477
	LR	0.9508	0.8486	0.9642
	TT	0.9411	0.0418	0.8068
Grace	GR	0.9010	0.1551	0.8282
	LR	0.8983	0.1549	0.8275
	TT	0.8950	0.8104	0.9455
Loc2_1	GR	0.8316	0.5516	0.8878
	LR	0.8339	0.6672	0.9068
	TT	0.8472	0.8039	0.9316

Table 1: *TMQI Result*
GR stands for global Reinhard TMO, LR is for localized Reinhard TMO and TT is for ToneTexture.

Table 1 shows the TMQI result for the tone mapped images. ToneTexture does the best in SpotB and Grace regions due to high statistical naturalness scores. Comparatively, SpotC is the region our method does the worst due to terrible naturalness score. Overall, ToneTexture preserves good structural fidelity in the test. This objective assessment agrees with our observation that our operator can tone map bright regions well and our drawback is in the dim region.

5. Conclusion and Future Work

We have presented an efficient method for extracting view optimized LDR image dynamically from HDR panoramas on HMD. This method notably improves the visual appearance in regions with extreme high intensity. In addition, modification of the technique also improves the visual appearance of Low Dynamic Range Panoramas as well.

5.1. Issues

ToneTexture provides significantly improved visual experience and allows for maximal use of available intensities while looking in a specific direction. There exist three problems with the naive implementation of the technique:

Midrange Crush Midrange color intensities can be crushed (similar to black crush) by the stretching of the color curve across the entire color range in a given view direction. This can lead to a loss of visual quality in various parts of the image.

Popping Image popping can occur as the viewer turns their head around. If the ToneTexture completely ignores global lighting information, turning view direction by a single degree can cause the image to shift dramatically. This can lead to significant visual irritation and dramatic scene damage. One solution is a fast global smoothing technique that can take global luminance into consideration.

Rotation When rotating the view point in Z-axis, the visual quality could be damaged. Since the pre-processing uses non-rotated rectangles, Z-axis rotation could include unexpected pixels. A good fix is to use circular viewable regions, yet the extra computation cost can be expensive. The global smoothing technique can help to resolve this issue at a lower cost.

Zooming Due to window-based image processing, zooming could potentially damage visual quality. While zooming in shrinks actual viewable window and just makes less use of ToneTexture, zooming out can be problematic because it brings in unprocessed pixels. Global smoothing technique can also help in this case. We're still working on this issue to find an optimal solution.

Stereo Stereo image pairs captured from different locations in space have nonidentical intensity information, meaning that ToneTextures generated separately are different as well. Thus, the tone mapping results are inconsistent in two images even when the viewer is looking at the same direction. This inconsistency can produce strange coloring behavior and lead to bad visual quality. A simple solution is to only generate one ToneTexture to be used for both images.

5.2. Extensions

In future work we intend to adjust adaptive equalization factor to improve visual quality in dim regions, compare with other TMOs, provide interface for editing ToneTextures for customized coloring and highlights, and explore the use of the technique on a wider variety of areas. Extension examples are:

- Use as a mask
- User controlled highlighted regions: editing.
- Reduced texture map resolutions.
- Mixed resolution of ToneTextures.
- Utilization in LDR panoramas.

ToneTexture may be edited after generated. By providing a tool for the user to interactively edit ToneTexture, an artist can dynamically adjust and control the coloring emphasis within a particular viewable region of the panorama. In addition, a variety of effects are possible such as having lighting fade and brighten around a specific region as the user turns and looks in different regions. This can be used for subtle emphasis effects.

While LDR imagery is already mapped into the dynamic range of the display device, various panoramas can still suffer during the compression process. By utilizing ToneTexture on LDR imagery, it is possible to enhance the visual appearance of a standard LDR image on HMD. We can expand the separation between pixel intensities by stretching the pixel intensities within the visible scene across the entire intensity range for the display.

A demo of ToneTexture in WebGL version is available via the following URL address:

http://cs.boisestate.edu/~scutchin

References

[ASS*11] AINSWORTH R. A., SANDIN D. J., SCHULZE J. P., PRUDHOMME A., DEFANTI T. A., SRINIVASAN M.: Acquisition of stereo panoramas for display in vr environments. In *IS&T/SPIE Electronic Imaging* (2011), International Society for Optics and Photonics, pp. 786416–786416. 1

[Blo13] BLOCH C.: *The HDRI handbook 2.0: high dynamic range imaging for photographers and CG artists*. Rocky Nook, Inc., 2013. 1

[CS05] CADFK M., SLAVÍK P.: The naturalness of reproduced high dynamic range images. In *Ninth International Conference on Information Visualisation (IV'05)* (2005), IEEE, pp. 920–925. 6

[DD02] DURAND F., DORSEY J.: Fast bilateral filtering for the display of high-dynamic-range images. In *ACM transactions on graphics (TOG)* (2002), vol. 21, ACM, pp. 257–266. 1

[FLW02] FATTAL R., LISCHINSKI D., WERMAN M.: Gradient domain high dynamic range compression. In *ACM Transactions on Graphics (TOG)* (2002), vol. 21, ACM, pp. 249–256. 1

[KUDC07] KOPF J., UYTTENDAELE M., DEUSSEN O., COHEN M. F.: Capturing and viewing gigapixel images. In *ACM Transactions on Graphics (TOG)* (2007), vol. 26, ACM, p. 93. 1

[LH12] LIU H., HU M.: Survey of high dynamic range image tone reproduction technology. *Journal of Jishou University (Natural Sciences Edition) 5* (2012), 019. 1

[MNP97] MATKOVIC K., NEUMANN L., PURGATHOFER W.: A survey of tone mapping techniques. *esc 1* (1997), 1. 1

[Rec11] RECOMMENDATION I.: 601-7: Studio encoding parameters of digital television for standard 4: 3 and wide screen 16: 9 aspect ratios. *International Telecommunication Union 96* (2011). 2

[RHD*10] REINHARD E., HEIDRICH W., DEBEVEC P., PATTANAIK S., WARD G., MYSZKOWSKI K.: *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2010. 1

[RSSF02] REINHARD E., STARK M., SHIRLEY P., FERWERDA J.: Photographic tone reproduction for digital images. *ACM Transactions on Graphics (TOG) 21*, 3 (2002), 267–276. 2, 6

[SCK*13] SMITH N. G., CUTCHIN S., KOOIMA R., AINSWORTH R. A., SANDIN D. J., SCHULZE J., PRUDHOMME A., KUESTER F., LEVY T. E., DEFANTI T. A.: Cultural heritage omni-stereo panoramas for immersive cultural analytics from the nile to the hijaz. In *2013 8th International Symposium on Image and Signal Processing and Analysis (ISPA)* (2013), IEEE, pp. 552–557. 1

[SJB10] SHAN Q., JIA J., BROWN M. S.: Globally optimized linear windowed tone mapping. *IEEE transactions on visualization and computer graphics 16*, 4 (2010), 663–675. 1

[WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing 13*, 4 (2004), 600–612. 6

[XF11] XIE Y., FANG J.: Development and prospect of tone mapping techniques for hdri. *China Illuminating Engineering Journal 5* (2011), 005. 1

[Yu] YU M.: Dynamic tone mapping with head-mounted displays. 2

[YW13] YEGANEH H., WANG Z.: Objective quality assessment of tone-mapped images. *IEEE Transactions on Image Processing 22*, 2 (2013), 657–667. 1, 2, 6

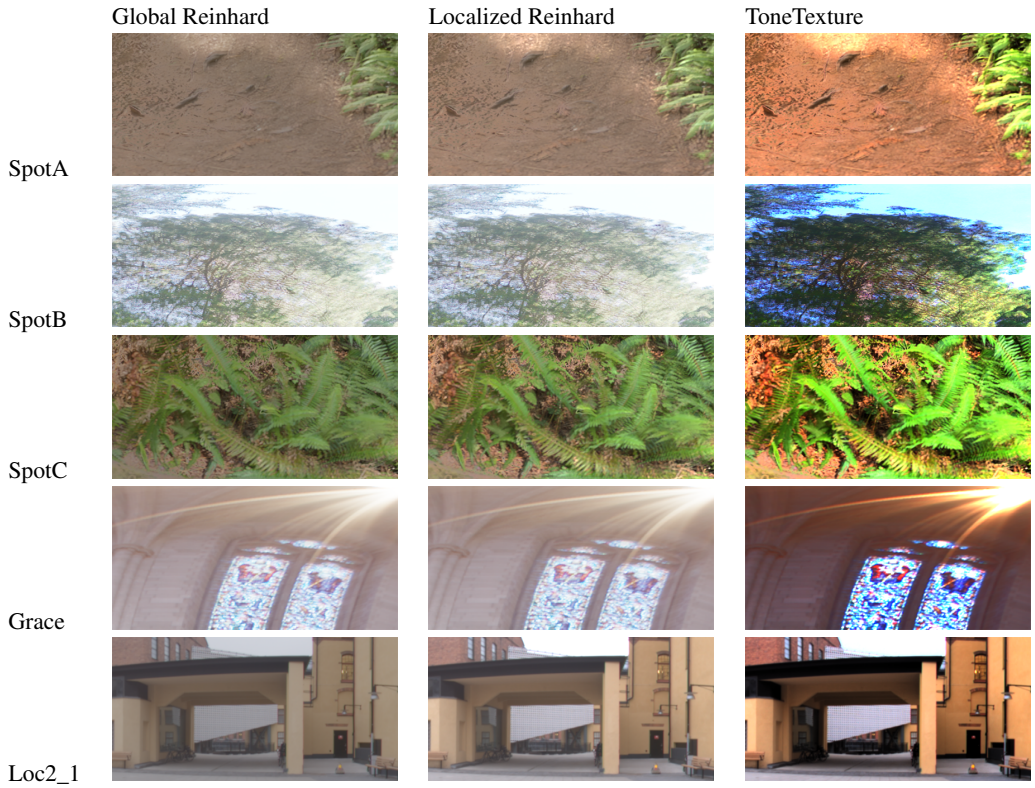

Figure 7: *Tone Mapped Image*

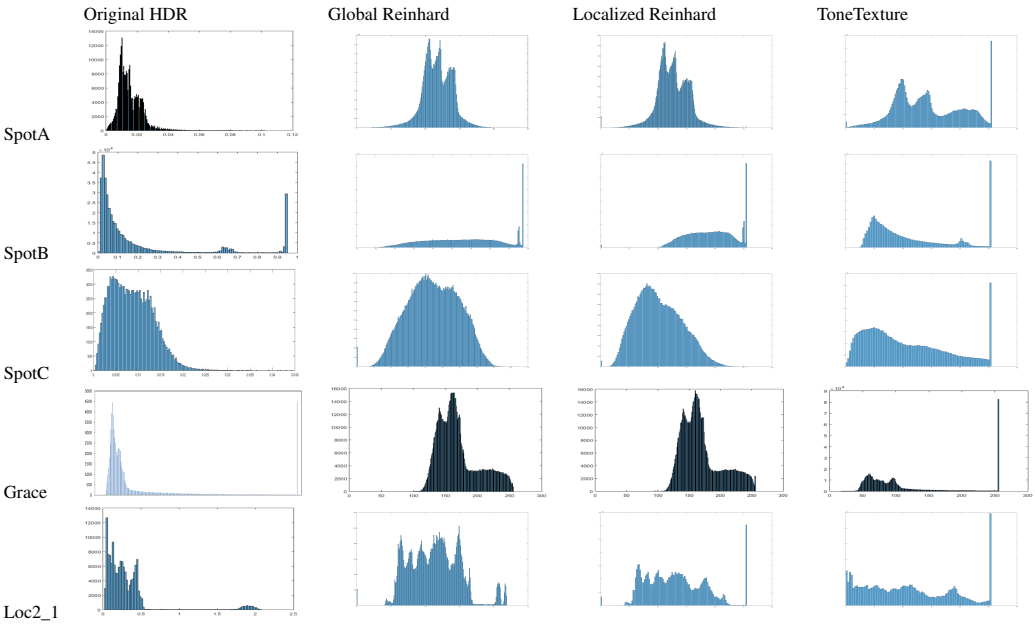

Figure 8: *Histograms of Test Comparisons*