Space Visualization and Rendering Evaluation

Vineet Ahirkar, Pratik Bhangale, Kyle Boyer, *graduate student*, UMBC.

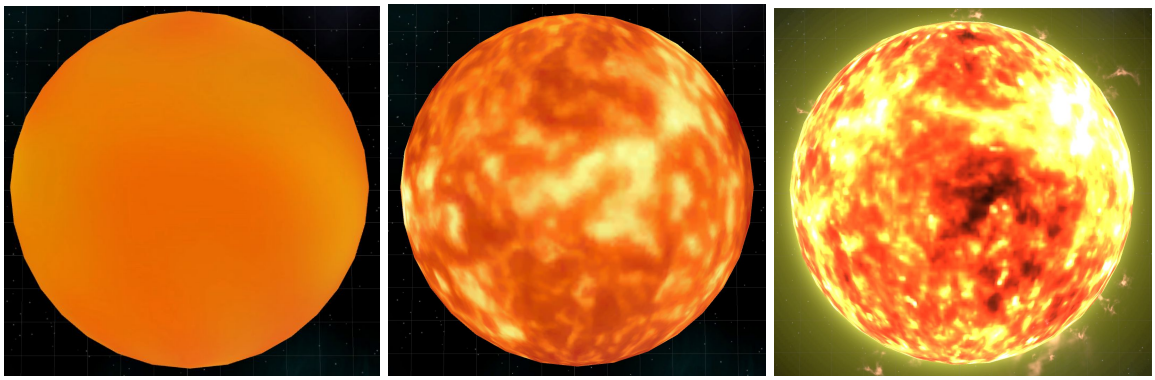

Fig. 1. Different Rendering Fidelity levels - Low, medium, and high detail.

Abstract— We wish to explore the capabilities and benefits of realistic volume and surface rendering in virtual reality. Our goal is to compare multiple rendering methods, specifically for volumetric data, when applied to VR. We will be implementing three different techniques for displaying voxelized data, and measuring the effects that these different implementations have on realism, performance, and scalability.

Index Terms— Volumetric rendering, Realism, Performance, Space visualization, Surface rendering.

1 Introduction

Virtual Reality, as a medium, is rapidly gaining momentum and popularity for all things related to graphical display. Software is being created that takes advantage of Virtual Reality's immersiveness to breathe new life into all sorts of rendering. It's being used for large scale 3D data visualization, life-like training simulations, and engaging video games. Within all of the new development for Virtual Reality, there is a common element among all applications: graphical rendering. Virtual Reality is very demanding on graphics processing units. Objects must be rendered at lightning speed, transitions must be made seamlessly, and projections must be performed to each eye in real time. As a result, when rendering anything to a Head Mounted Display (HMD), performance suffers. This raises the question of how much performance is worth sacrificing, and for what level of realism should we give up framerate. We will attempt to answer these questions by conducting an objective study of realism and performance, based on predefined levels of detail rendered in Virtual Reality.

2 Related Work

The process of visualizing volumetric data has been around for quite a few years now, and it is becoming widely used in the decision making process surrounding the data in question [2]. We utilize visualizations, both interactive and not, to make informed assessments about what we're looking at. This concept is applied to many datasets and problems, to find potential solutions. Examples include air quality index, chemical sensing fields, and spatio-temporal structures built from satellite information [1].

This method, although a useful and often effective one, comes with some inherent difficulties and potential problems. One of those issues is how to actually render the data in a 3 dimensional space. The more traditional way of rendering volumetric data is to utilize step based ray tracing, in which pixels are calculated by the summation of consistent checks inside the volume along a viewing ray. With some hardware acceleration, based on a discrete graphics card, Mayerich and Keyser share the opinion that this is by far the best method for volumetric rendering [4]. Gascon, Espadero, Perez, and Torres contest that a more graphics pipeline centered rasterization method is very effective [3]. They propose that triangle meshes can be rendered, and then deformed as need be to simulate non-rigid volumes [3].

Our project takes volumetric rendering a step further, by porting it into the VR world. We would like to expand upon these types of works and explore the costs of rendering volume data in 3D Virtual Reality. There have already been numerous studies and applications

of spatial and volumetric data visualization in VR. It introduces numerous problematic scenarios, including how to effectively handle occlusion [5]. In our implementation, we plan to use a ray casting technique akin to the one that Mayerich and Keyser describe, rather than the isosurface rasterization. However, with our research, we plan to explore different levels of fidelity with the ray casting, by implementing extra additions like stochastic ray origins. Virtual Reality tends to not support vast data space for 3d volumes very well, especially when the user is expected to traverse the data space by mere movement. Our work will be focused on solving a problem, similar to Gascon and Espadero [3], or Fischer and Bartz [5], but we will be evaluating different levels of rendering and shading techniques, to find what is necessary and effective, rather than the complications that come with how to interact with the data.

3 Design and Implementation

To evaluate the effects, and cost, of increased visual clarity on VR renderings and data representations, we designed a case study based on three settings of what we refer to as "fidelity." Fidelity represents the rendering quality of a particular scene. The higher levels of fidelity include techniques for displaying surfaces and volumes that should result in a more realistic representation of our intended objects. The surfaces are created with higher detail, and the volumes are rendered more rigorously. The extra computation required to create the higher fidelity scene should, in theory, come at a performance reduction.

Having three different levels of fidelity will allow us to evaluate performance and realism at three different input levels. We will be using a "low" fidelity, a "medium" fidelity, and a "high" fidelity environment setup. These three different scenes will then be measured objectively for graphics processing performance, and realism. The inspiration for our experimental design came from a publication by Eric Ragan and Doug Bowman, in which they conducted an experiment to evaluate the effect of visual realism on military task completion. [7]

3.1 Experiment Implementation Description

3.1.1 Independent Variables

The main independent variable in this experiment is:

1. Rendering Quality
    a. Ray casting additions/modifications
    b. Surface shading techniques

3.1.2 Dependent Variables

The dependent variables in this experiment are:

1. Realism
2. Performance

3.1.3 General Overview

Our main focus of the experiment is to compare different rendering techniques on the basis of realism and performance.

Realism of volumetric data can be best measured in a relative context. So for calculating the metrics we have a reference technique which has optimum realism. Two test techniques are then compared with the reference technique.[2]

We used 4 metrics to get an estimate of the realism in the technique being tested.

Calculate 4 metrics (in Matlab) -

1. Color Variance

Calculate the number of unique colors in the environment and divide it by the total number of pixels. Images could differ from each other in the number of unique colors, and is thus possibly perceived as less realistic. Higher value of CV denotes an image with more varied color information. Thus, relatively comparing the CVTest and CVReference we get the loss in color information thus giving a general idea of Realism. Lower value of MetricCV , more it is closer to the Reference Image and depicts more realism.

$$M_{Color} = \frac{|I_C|}{N}$$

where, I: image in RGB space, N: number of pixel values in I , I_c: { (R1,G1,B1), (R2,G2,B2), (R3,G3,B3) }

$$\mathbf{Metric_{CV}} = \left| \mathbf{CV_{Reference}} - \mathbf{CV_{Test}} \right|$$

where, CV: Color Variance.

2. Mean squared Error (MSE)

Conveys the difference in pixel values of the test image and the reference image. The main advantage of MSE is that it allows us to compare the "true" pixel values of reference image to our test images. Lower the MSE more it is similar to the reference image and thus has more realism. The problem is that it depends strongly on the image intensity scaling.

$$MSE = \frac{1}{NxM} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} [X(i,j) - Y(i,j)]^2$$

...where X: Test Image , Y: Reference Image

3. Peak Signal to Noise Ratio (PSNR)

Ratio between the maximum possible quality of an image (Reference Image) and the data loss (due to texture compression) that affects the realism in it. PSNR is calculated in the logarithmic scale and measured in terms of decimals. Higher the PSNR value more it is similar to the reference image and thus has more realism.

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

...where, MAXI: maximum possible pixel value of the image. (8 bit ~ 255) , MSE: Mean Squared Error

4. Structural Similarity Index Metric (SSIM)

Evaluates differences in structure between a reference and test image.
Structure can be essentially defined as the absence of luminance and contrast. The main advantage of this technique is that it is consistent with human visual perception. It first generates a map of local distortion values and then pools those values into a single distortion value. Higher the value of SSIM, more the realism.

The Formula for SSIM is as follows:

$$\mathrm{SSIM}\left(u^*, u^0\right) = \frac{\left(2\mu_{u^*}\mu_{u^0} + c_1\right)\left(2\sigma_{u^*u^0} + c_2\right)}{\left(\mu_{u^*}^2 + \mu_{u^0}^2 + c_1\right)\left(\sigma_{u^*}^2 + \sigma_{u^0}^2 + c_2\right)}$$

...where, μ: average of all pixels, σ: variance of all pixels, c: constant.

We hypothesize that the quantitative analysis of rendering techniques via the below mentioned metrics will prove to be a good measure of realism in an immersive VR environment, and that high fidelity rendering will negatively affect performance. We believe that the performance may be affected too much by the increase in realism, and thus realism may need to be throttled, to increase computation speed.

3.2 Software Implementation Description

3.2.1 Tools used

The rendering for this project will be implemented using a combination of many different tools and languages. Those entities are listed here:

1. Unity

Game development platform. Used to implement the construction of our world environment, and facilitate xyz coordinate space changes. We use Unity to tie everything together, and to build the final project.

2. SteamVR

Virtual Reality platform developed by Valve. This is used to implement the head tracking functionality for the HTC Vive. SteamVR offers an integration package that can be used from the Unity Asset Store, to be imported directly into a Unity project. This was how we facilitated VR control.

3. C#

Scripting language that integrates into Unity, using a library called UnityEngine. This language is used to manipulate Game Objects in the Unity environment. It is primarily utilized to create objects, read data input, texture planets, and construct the environment and interactions.

4. HL/SL

Shader language used to implement the rendering techniques we will be testing. HL/SL is a shader language that runs on the GPU. Unity has its own built in shading packages and lighting models, but we will be overriding them with our own custom shader code. The code will primarily be used for volumetric rendering, with additional rendering techniques such as stochastic ray marching implemented as well. HL/SL will be used to create both vertex shaders and fragment shaders for the implementation.

For calculating the metrics we used the following tools:

1. Matlab

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms.

We used matlab for programming realism metrics. For metric calculation we took snapshots of the environment from various angles in VR environment. We wanted to compare the 3 rendering variants so, we choose the high detailed rendering as our reference image and compare other two variants to determine how realistic they are.

Primarily we used four different matlab programs to determine the realism:

1. Color Variance
2. Mean squared Error
3. Peak Signal to Noise Ratio
4. Structural Similarity Index Metric

3.2.2 Lighting, Modelling, and Rendering

The lighting model used in our implementation was global ambient light functionality provided by Unity. However, all individual point lights were removed from our environment. This allowed us to simulate the conditions in space, creating the illusion of lit celestial bodies. Each entity was modeled using a standard Unity sphere asset, with a different texture overlaid, as per the following:

- *Low Fidelity:* Low resolution texture. (64x64 upscaled to 2048x2048)
- *Medium Fidelity:* Medium resolution texture. (1024x512 upscaled to 2048x1024)
- *High Fidelity:* Uncompressed textures, multiple,

moving. (4096x4096). Added glow and particle effects

The volumetric rendering eas handled with different Ray Marching techniques, all implemented using HL/SL in vertex and fragment shaders. These shaders were written and used in place of Unity's provided shaders, effectively overriding them. This required overriding Unity's standard lighting model for lighting surfaces with our own, for manipulating volumes. The volumetric data was rendered as follows:

- *Low Fidelity:* 3D voxel texture construction, follwed by a direct density lookup into the voxel texture.
- *Medium Fidelity:* 3D voxel texture construction, followed by the execution of a density-to-color mapping function. The function performed a secondary 2D texture lookup to complete mapping, as the last functional step, which resulted in a final pixel color.
- *High Fidelity:* 3D voxel texture construction, followed by the same mapping function with 2D texture as above. A stochastic ray origin was added to alleviate ripple artifacts very visible in the medium fidelity version.. Additional slicing was also performed through the volume, to get a more realistic appearance.

The background, as previously stated, was a standard Unity skybox with lighting effects removed.

3.2.3 User interaction system and Interaction technique

We rendered a space environment in which the subject can do actions using the HTC Vive controllers.

The triggers on both the controllers may be used to thrust in the direction the controller is pointing, thus enabling movement in space. This also simulated "floaty" spatial movement, akin to freely flying in a spacecraft.

The touchpads on the Vive controllers can be used to slice both volumes left, right, up, and down. This allows the user to see inside the rendered volume in real time, and scrutinize the data far more effectively. The position of the volumes can be reset to default by squeezing the left controller.

Similarly, squeezing the right controller changes the environment from low, to medium, to high fidelity. It effectively toggles the current level of detail being displayed to the user. Conversely, this will also lower the framerate, hence our topic of study.

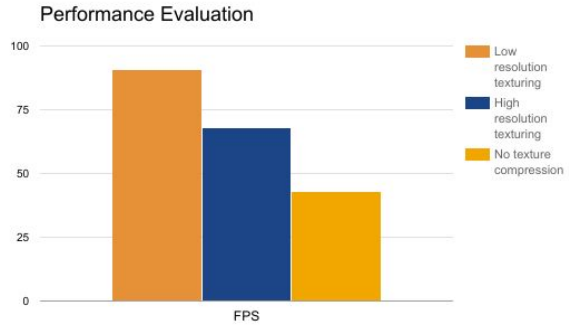

Fig. 2. Performance Evaluation

3.2.4 Display type

We used an HTC Vive head mounted display (HMD). The Vive has a Field of View of 110 degrees. It also boasts a relatively high refresh rate of 90Hz. This means that any frame rate increases above 90fps will not be visible to the user.

The implementation was developed and tested on one of the author's home systems. The system is more graphically capable than most home PCs. The specifications are as follows.

- Processor: Intel Core i7 4790k - 4.6 GHz
- GPU: nVidia GTX 1080
- RAM: 16 GB
- Storage: Samsung SSD (irrelevant for real-time rendering, but reduces texture construction time)

4 Results

4.1 Statistics

Independent Variables

The main independent variable in this experiment is:

1. Rendering Quality
    c. Ray casting additions/modifications
    d. Surface shading techniques

Realism of volumetric data can be best measured in a relative context.

Calculate 4 metrics (in Matlab) -

1. Color Variance
   Calculate the number of unique colors in the environment and divide it by the total number of pixels.

	Reference Image	Test Image 1	Test Image 2
CV	0.0731	0.0482	0.0554

	Test Image 1	Test Image 2
MetricColor	0.0249	0.0177

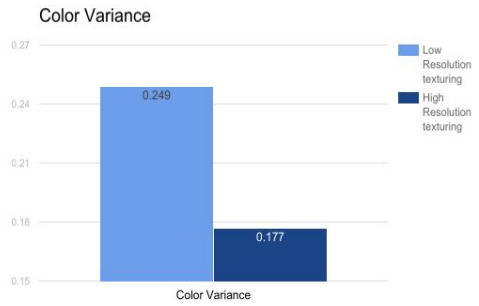

Fig. 3. Color Variance

2. Mean squared Error

Conveys the difference in pixel values of the test image and the reference image.

	Test Image 1	Test Image 2
MSE	0.03147	0.030856

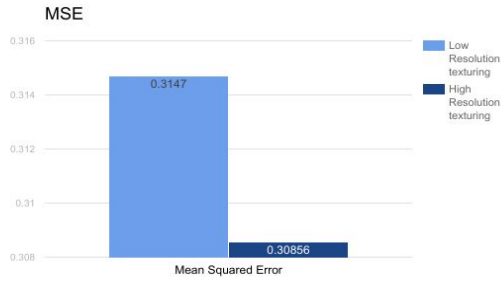

Fig. 4. Mean Square Error

3. Peak Signal to Noise Ratio

Ratio between the maximum possible quality of an image and the data loss that affects the realism in it.

	Test Image 1	Test Image 2
PSNR	13.1518	13.2375

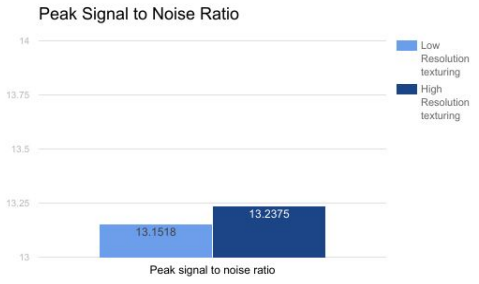

Fig. 5. Peak Signal to Noise Ratio

4. Structural Similarity Index Metric

Evaluates differences in structure between a reference and test image.

	Test Image 1	Test Image 2
SSIM	0.7894	0.7942

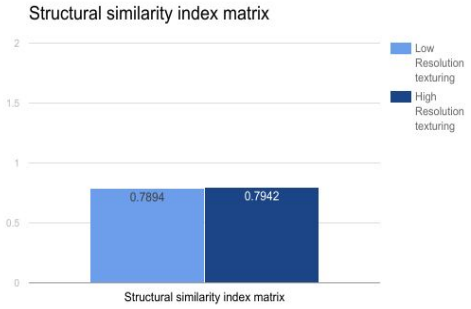

Fig. 6. Structural Similarity Index Metric

4.2 Study Results

For Color Variance (CV) and Mean Square Error (MSE) lower the value better is realism for that technique. For Peak Signal to Noise Ratio (PSNR) and Structural Structural Similarity Index Metric (SSIM) we observed that higher the value of these metrics are higher is the realism for that technique.

From the results of the metrics we used, it was seen that the 'test image 2' had better realism that 'test image 1', which could also be seen be seen by visual perception of the images by the human eye. This means that the 'High resolution texturing' technique performed better than the 'Low resolution texturing' technique.

Thus, the metrics proved to be a good measure of realism in an immersive Virtual Reality environment. Also, using only one of the metrics would give a biased opinion about the realism and thus more the metrics used better would be the results.

Calculating metrics based on a reference model proves to be beneficial as we can get a threshold of the optimum value of that metric. If no reference is used then metric values do not give sufficient information about the realism in the image.

Items we think are need to fix are to put more details in the rendered scene. To make the scene look more realistic we need to add some stars and asteroids in it.

Also, we can show an FPS counter in the scene so that people can get to know the performance of the scene being rendered.

5 Discussion

We hypothesised that the four metrics are good measures of realism in an immersive VR environment, and that an increase in realism will result in a decrease in performance. The stated hypothesis has thus proved to be true by doing the analysis on the various rendering techniques mentioned above.

Something interesting we found from results of the metrics was that, every metric works in a different aspect to measure the reality in an image. Some metrics try to extract the color while other look at structure in an image. Color Variance and Mean Squared Error tries to extract information about the color loss between two images. Peak Signal to Noise Ratio does the same, but works well in bigger images. Structural Similarity Index Metric on the other hand tries to identify the basic structure of an image and then looks at the changes in the structure between the two images supplied. Thus, using different kinds of metrics we get an overall idea of the realism in an image.

In future, we would like to compare the results of the objective study with the results from a subjective analysis. In a way, this will validate the results we see in the quantitative evaluation.

6 Conclusion

Realism in an image can be best measured in a relative context. Using multiple metrics gives information about different aspects in the image (eg - Color, Structure, illuminance) which gives detailed analysis of realism in the image.

We conclude that the correct level of fidelity to utilize when rendering in VR actually depends very heavily upon what hardware is available. High end hardware that can handle the extra rendering computation makes it worth it to display additional realism. The HMD refresh rate in VIVE is 90 Hz. Thus, any frame rate change over this value will be inconsequential. To preserve immersion, it is very desirable to maintain a frame rate relatively close to 90Hz. Inconsistent or “choppy” frame rate will very negatively affect a VR experience. It is better to sacrifice a bit of realism for performance increase in most cases, especially if the intended hardware to run the desired application is limited.

Acknowledgments

The authors wish to thank Dr. Jian Chen. This work was supported in part by a grant from NSF for the PI^2 (a CAVE2-Inspired Display) and the Virtual Reality course for Spring 2017. The authors would also like to extend their thanks to Dr. Neel Savani, NASA Goddard Space Flight Center, for supplying datasets and providing guidance through the project’s development.

References

[1] Huan Li; Hong Fan; Feiyue Mao. A Visualization Approach to Air Pollution Data Exploration—A Case Study of Air Quality Index (PM 2.5) in Beijing, China

[2] Norman Wang , Wendy Doube : How real is reality?
A perceptually motivated system for quantifying visual realism in digital images

[3] Bryan N. Duncan a, *, Ana I. Prados a, b, Lok N. Lamsal a, c, Yang Liu d, David G. Streets e,Pawan Gupta a, c, Ernest Hilsenrath b, f, Ralph A. Kahn a, J. Eric Nielsen g,Andreas J. Beyersdorf h, Sharon P. Burton h, Arlene M. Fiore i, Jack Fishman j,Daven K. Henze k, Chris A. Hostetler h, Nickolay A. Krotkov a, Pius Lee l, Meiyun Lin m,Steven Pawson a, Gabriele Pfister n, Kenneth E. Pickering a, R. Bradley Pierce o, Yasuko Yoshida a, g, Luke D. Ziemba h : “Satellite data of atmospheric pollution for U.S. air quality applications”: Examples of applications, summary of data end-user resources, answers to FAQs, and common mistakes to avoid.

[4] Jorge Gascon; Jose M. Espadero; Alvaro G. Perez; Rosell Torres; Miguel A. Otaduy. Fast Deformation of Volume Data Using Tetrahedral Mesh Rasterization.

[5] David Mayerich; John Keyser: Hardware accelerated Segmentation of Complex Volumetric Filament Networks.

[6] Jan Fischer; Dirk Bartz; Wolfgang Straber: Occlusion Handling for Medical Augmented Reality using a Volumetric Phantom Model.

[7] Eric Ragan; Doug Bowman; Effects of Field of View and Visual Complexity on Virtual Reality Training Effectiveness for a Visual Scanning Task