# Objective Image Quality Assessment of Texture Compression

Wesley Griffin*
National Institute of Standards and Technology

Marc Olano†
University of Maryland, Baltimore County

## Abstract

Texture compression is widely used in real-time rendering to reduce storage and bandwidth requirements. Recent research in compression algorithms has explored both reduced fixed bit rate and variable bit rate algorithms. The results are evaluated at the individual texture level using Mean Square Error, Peak Signal-to-Noise Ratio, or visual image inspection. We argue this is the wrong evaluation approach. Compression artifacts in individual textures are likely visually masked in final rendered images and this masking is not accounted for when evaluating individual textures. This masking comes from both geometric mapping of textures onto models and the effects of combining different textures on the same model such as diffuse, gloss and bump maps.

We evaluate final rendered images using rigorous perceptual error metrics. Our method samples the space of viewpoints in a scene, renders the scene from each viewpoint using variations of compressed textures, and then compares each to a ground truth using uncompressed textures from the same viewpoint. We show that masking has a significant effect on final rendered image quality, that graphics hardware compression algorithms are too conservative, and reduced bit rates are possible while maintaining quality.

**CR Categories:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture;

**Keywords:** MIP mapping, bump maps, texture compression, image quality assessment

## 1 Introduction

Real-time rendering makes extensive use of texture compression for reducing storage size on the Graphics Processing Unit (GPU), bus bandwidth for uploading textures, and memory bandwidth for deferred rendering buffers. Current GPU compression algorithms use fixed bit-rates [Iorcha et al. 1999; Microsoft 2013] and recent research has explored reducing the fixed bit rate [Ström and Pettersson 2007; Khronos Group 2013] as well as using variable bit-rate algorithms [Olano et al. 2011].

In evaluating texture compression, the most common approach is to use some combination of the Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), or visual image inspection on the individual textures. While comparing or inspecting individual textures is straightforward to implement, this method does not properly account for the two ways in which textures are used.

Textures are not single images but are instead used in two specific ways. First, they are mapped onto geometric objects and then those texture-mapped objects are rendered from some viewpoint. Second, multiple textures are frequently used together, such as a diffuse and bump map, or a diffuse, gloss, and bump map and these sets of textures are used to evaluate the rendering equation.

In both cases, the textures are used to render a final image from a specific viewpoint. In that rendered image, there are likely masking effects from geometric distortion or overlapping textures that affect that quality of the image [Ferwerda et al. 1997]. We define *geometric* masking to be the effects from texture mapping and *texture set* masking to be the effects from multiple textures. These masking effects cannot be accounted for when evaluating individual textures, they only appear when rendering an image from a viewpoint.

In order to account for these masking effects we propose to evaluate texture compression on final rendered images. Rendering a scene from a single viewpoint, however, is effectively the same as evaluating individual textures, as the masking effects will likely vary between viewpoints. Since the set of possible viewpoints in a scene is infinite but discrete, we propose to sample this viewpoint space and evaluate the final rendered images at each sampled viewpoint.

For every sampled viewpoint, we render the scene using variations of compressed textures. At a viewpoint, we also render a ground truth image using uncompressed textures. We then compute error metrics at each viewpoint, comparing each compressed variation to the uncompressed ground truth. By using final rendered images, we can apply perceptually rigorous objective image quality assessment metrics from the signal processing literature.

Our results show:

- Geometric and texture set masking does occur and cannot be accounted for when evaluating individual textures.
- Perceptual sensitivity to masking varies with the type of texture, such as diffuse, gloss or bump maps.
- Current GPU compression algorithms are too conservative and bit rates can be reduced while maintaining final rendered image quality.

## 2 Related Work

We are interested in evaluating compression artifacts and present results on several common GPU compression algorithms. We choose to use two different perceptual error metrics for our evaluation, in addition to using root mean square (RMS) color error for a baseline reference. In this section, we review GPU compression followed by objective image comparison research.

### 2.1 GPU Texture Compression

GPU texture compression algorithms have evolved over the years but still remain based on the fixed-bit-rate block-truncation coding algorithms introduced by Delp and Mitchell [1979] and extended by Iorcha et al. [1999]. These algorithms allow direct access to

*e-mail:wesley.griffin@nist.gov
†e-mail:olano@umbc.edu

compressed blocks in a texture and fast decompression in hardware. The main idea behind the technique is to encode a pair of representative colors for a 4×4 block of pixels and then store per-pixel index values that interpolate along the gradient defined by the endpoint colors. Recent variants such as BC7 [Microsoft 2013] improve compression quality by subdividing the blocks to encode multiple color gradients at the same bit rate.

Recently, Adaptive Scalable Texture Compression [Khronos Group 2013] was introduced. ASTC is modeled after the existing block compression algorithms but supports varying the number of texels per block (per texture) to enable granular changes to the bit rate. The compressed block size remains fixed to still allow direct access. ASTC also supports varying the endpoint encoding method per block as well as partitioning the block into multiple endpoints.

Several compression algorithms are tailored for low-power hardware. PVR uses two low-resolution source images combined with a full-resolution modulation image [Fenney 2003]. ETC encodes two 2×4 blocks of pixels together using a single chrominance value and a luminance modulation table [Ström and Akenine-Möller 2005] and ETC2 [Ström and Pettersson 2007] increases compressed quality by using invalid bit combinations for additional modes.

### 2.2 Objective Image Comparison

A widely-used technique for objective image comparison is to compute the PSNR between a reference and distorted image. However, research has led to the realization that the PSNR does not correlate with how humans evaluate images [Wang et al. 2004]. Objective Image Quality Assessment (IQA) research has since focused on metrics with strong correlation to human perception. Current top-performing metrics are formulated for luminance natural images such as real-world photographs or realistically rendered images.

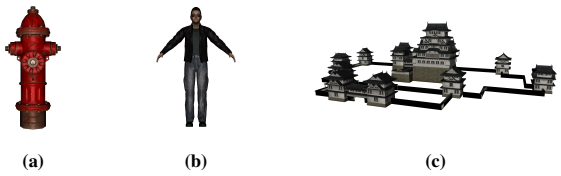
IQA metrics can be classified in two ways: by the type of input or type of model. The input is either luminance or color while the model is either *bottom-up* or *top-down*. Input classification is distinct, that is, metrics work on either luminance or color. Model classification, however, is more of a convenience and most IQA metrics are a mix of both models. Bottom-up metrics attempt to accurately model the human visual system while top-down metrics just model the human visual system's input-output characteristics.

The luminance-only metrics have very strong correlation with human perception, even for chrominance-based distortions such as texture compression. This performance is due to the human visual system being more perceptive to errors in luminance than errors in chrominance [Hao and Shi 2000]. One of the earliest, but still competitive metrics is the Structural Similarity Index Metric (SSIM) introduced by Wang et al. [2004].

SSIM is a top-down metric that evaluates differences in *structure* between a reference and distorted image. Wang et al. hypothesized that local luminance and local contrast changes do not strongly affect perceived image quality and thus define structure as the absence of luminance and contrast in an image. SSIM operates in two stages by first generating a map of local distortion values and then pooling those values into a single distortion value using Minkowski pooling.

Current color metrics attempt to combine rigorous perceptual-based CIELAB color differencing [Fairchild 2005] with bottom-up models of the human visual subsystem. The results, however, do not correlate as well with the evaluation databases [Ajagamelle et al. 2010] and color image quality assessment remains difficult to solve.

Additionally, as Čadík et al. [2012] show, existing image quality metrics, which were developed to evaluate compression artifacts, have difficulty in evaluating global illumination artifacts in synthesized images. Their results indicate that further research is needed to improve image quality metrics for rendering.



**(a)** **(b)** **(c)**

**Figure 1:** **(a)** *Fire Hydrant* **(b)** *Urban Guy* **(c)** *Japanese Castle.*

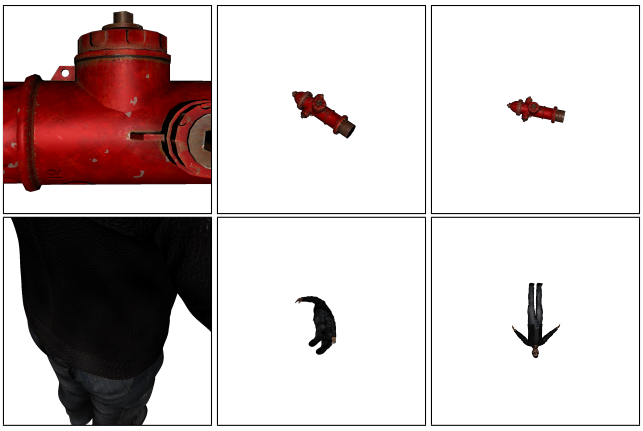


**Figure 2:** *Closest, median (by distance), and furthest viewpoints.*

## 3 Approach

In order to perform the evaluations, we need to render the scene from a set of viewpoints. We choose to sample the infinite, but discrete, space of viewpoints and then render a final image at each viewpoint using variations of compressed textures. After rendering, we then compare each final rendered image to a render at the same viewpoint with uncompressed textures.

### 3.1 Viewpoint Sampling

Our evaluation method cannot use just a single viewpoint, as that would not account for the possible differences in masking effects from various viewpoints. Since the viewpoint space is infinite, we must sample it to make our approach computationally tractable. We choose to sample the viewpoint space uniformly. Sampling techniques such as importance sampling based on expected viewer locations could improve the evaluation, but we leave these for future work. Viewpoint sampling has been used in other areas of computer graphics, such as for image-based mesh simplification [Lindstrom and Turk 2000], but not, to our knowledge, for texture compression.

For the single-object models, Fire Hydrant and Urban Guy, we choose to restrict the set of viewpoints to a bounding sphere. We use the quasi-random Sobol sequence generator [Joe and Kuo 2003] to sample this sphere of viewpoints. To ensure quality viewpoint samples, we constrain the sphere to a multiple of the object's bounding sphere. We also reject any viewpoint samples closer than a multiple of the bounding sphere radius, to prevent the viewpoint samples from being inside the model. The multipliers were chosen such that for each model, the final rendered images ranged from having 0.8% to 70% pixels filled. Figure 2 shows the closest, median (by distance), and furthest viewpoint for both models.

The set of possible viewpoints is frequently restricted to some subset of space in the scene. For example, in many games, the viewpoints are restricted to a character's point of view or to a path of camera movements that follow a character. With an instrumented rendering engine, a trace of viewpoints could be captured during, for instance, game testing. These traces would become the space of viewpoints to sample. To demonstrate this, a set of key viewpoints for the third model, Japanese Castle, were captured while navigating around the model. These key viewpoints were then interpolated to form a full set of viewpoints that were evaluated.

### 3.2 Rendered Image Comparisons

At each viewpoint, we render a ground truth image using uncompressed textures. We then render the same viewpoint using variations of compressed textures. After all the images are rendered from a single viewpoint, we compare each compressed variation final rendered image to the uncompressed ground truth image. We extend the method, introduced by Beers et al. [1996], with perceptually rigorous objective image quality assessment metrics.

We evaluate three metrics: the Root Mean Square (RMS) color error for a baseline reference, the CIELAB $\Delta E_{94}$ color differencing metric to evaluate chroma error, and the Structural Similarity Index Metric (SSIM) to evaluate structural error. Since SSIM captures error in the structure of an image, it is well suited to capturing compression artifacts in the bump and gloss maps.

Each metric is averaged across the set of viewpoints to compute a mean metric score for each compression algorithm. To further isolate just the compression artifacts, we modified each metric to compute the error over just the pixels that were rendered. This was done by using an RGBA framebuffer for the rendering, and then weighting the metric at each pixel by the alpha value.

SSIM is formulated for luminance-only natural images. While some have applied SSIM to the color channels of an RGB image, SSIM is only a valid perceptual metric when applied to luminance. For our evaluation, we transform the rendered RGB images into the CIELAB color space and the compute SSIM on the $L^*$ component. We also compute the CIELAB $\Delta E_{94}$ color differencing metric on the entire CIELAB image as our full-color perceptual metric.

We also present results using the Root Mean Square (RMS) color error (Equation 1). This metric has no perceptual foundation, but is widely used and we include it for reference. Note that we are still measuring the influence of geometric and texture set masking on final rendered image quality and not individual texture error.

$$\text{RMS}_{\text{Color}} = \sqrt{\frac{1}{N}\sum_{1}^{N}(\Delta r^2 + \Delta g^2 + \Delta b^2)} \qquad (1)$$

Finally, we report the RMS angular error (Equation 2) computed on individual bump textures. This metric is used to evaluate the individual bump textures to compare against our evaluation method.

$$\text{RMS}_{\text{Angular}} = \sqrt{\frac{1}{N}\sum_{1}^{N} arccos(n_0 \cdot n_1)^2} \qquad (2)$$

where $n_0$ and $n_1$ are the reconstructed unit normals.

## 4 Results

We applied our method to a set of models and compression algorithms. In this section we describe the models and selected compression algorithms and discuss our results.

### 4.1 Models

We use three models shown in Figure 1: Fire Hydrant, Urban Guy, and Japanese Castle. Each model has at least one texture atlas set which consists of a bump map, gloss map, and diffuse map, all using one parameterization. The Fire Hydrant model (Figure 1a) uses one texture atlas set at a resolution of $2048 \times 2048$. The Urban Guy model (Figure 1b) uses four texture atlas sets, one for the skin and clothes at $2048 \times 2048$, one for the jacket at $1024 \times 1024$, and two for the accessories (sunglasses and watch) at $512 \times 512$. The Japanese Castle model (Figure 1c) has 21 texture atlas sets for the various buildings and props, one set at $2048 \times 2048$, four at $1024 \times 1024$ and the rest at lower resolutions (most at $512 \times 512$).

We use the Cook-Torrance shading model with true Fresnel assuming unpolarized light. The mipmaps for each texture were generated with a box kernel. The bump and gloss maps were transformed into the second moments of variance for proper linear filtering [Olano and Baker 2010], then converted back into gloss for texture storage. The bump maps are stored in the dual-paraboloid encoding [Heidrich and Seidel 1998] and the gloss maps encode the Cook-Torrance variance, which is reconstructed in the pixel shader using: $\sigma^2 = 2/(2^{10r+1} + 2)$, where $r$ is the gloss value.

### 4.2 Compression Algorithms

We use three different formats for Direct3D Block Compression: BC5 (8 bits/pixel compression rate) for the bump maps, BC4 (4 bits/pixel compression rate) for the gloss maps and BC3 (8 bits/pixel compression rate) for the diffuse maps. The NVIDIA Texture Tools Library [NVIDIA 2013] was used for compression, with the quality set to 'normal'.

We use three different block sizes for Adaptive Scalable Texture Compression: 4×4 (8 bits/pixel compression rate), 8×8 (4 bits/pixel compression rate), and 12×12 (0.89 bits/pixel compression rate) with RGBA textures for all three maps. The bump maps are compressed in the RG channels, while the gloss maps remain RGBA. The ARM Mali ASTC Evaluation Codec [ARM 2013] was used for compression, with the quality set to 'thorough'.

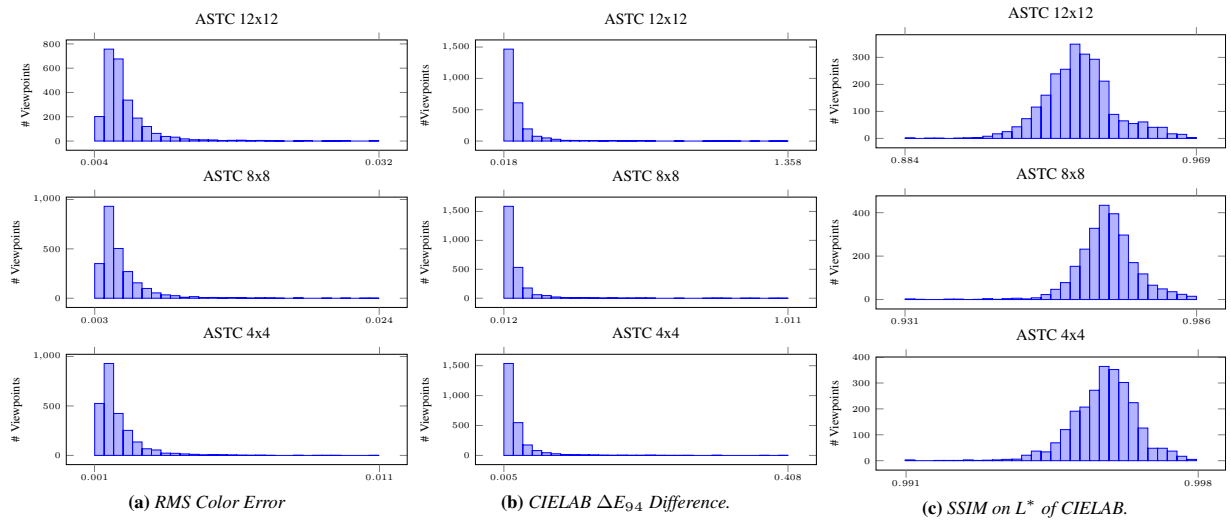
To explore how far compression can be pushed, we "compressed" the textures by dropping from 1–7 bits from each color plane. In addition to compressing all of the textures, we also chose to compress just a single class of texture, such as diffuse or gloss textures, leaving the other textures uncompressed to determine which class had the most effect on the final rendered image quality. Finally, we sample the viewpoint space with 2500 samples.

### 4.3 Discussion

We first explain our results and then make several conclusions. Table 1 and Figures 3, 4 and 5 show the histograms and statistics (over all sampled viewpoints) for the individual viewpoint error metrics.

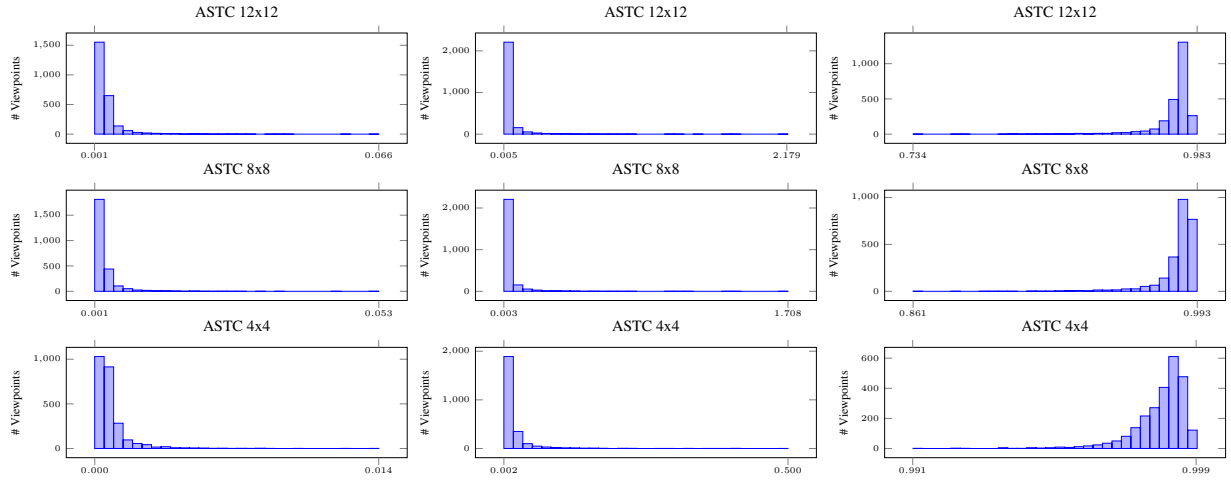
Table 2 reports the mean of the error metrics over all of the models. The **Sampled Viewpoint Mean Error** columns report the mean error over all 2500 viewpoint samples. The **Individual Texture Error** columns report the mean error over all of the MIP levels for the largest-sized texture set. The All column is for compressing all of the textures, while the Diffuse, Gloss, and Bump columns are for compressing just that respective texture. The NVTT/BC rows refer to Direct3D Block Compression using the NVIDIA Texture Tools Library and the ASTC rows refer to Adaptive Scalable Texture Compression using the ARM Mali ASTC Evaluation Codec.

Figures 6, 7, and 8 plot the **Sampled Viewpoint Mean Error** data in Table 2. The left two vertical plots are color error against com-



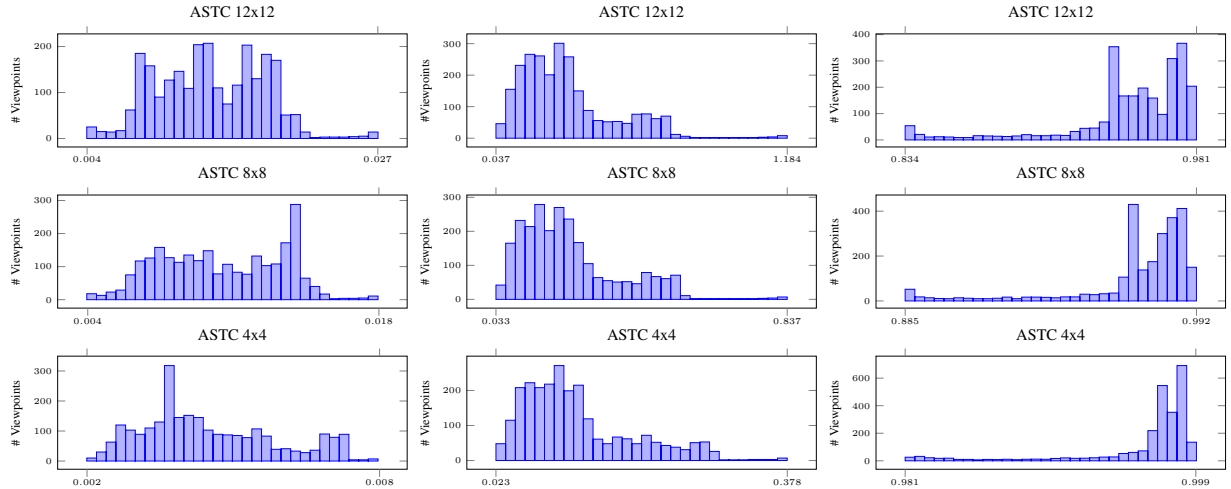

**(a)** *RMS Color Error* **(b)** *CIELAB $\Delta E_{94}$ Difference.* **(c)** *SSIM on $L^*$ of CIELAB.*

**Figure 3:** *Fire Hydrant error metric histograms. Only the ASTC algorithms are shown and the error metric is on compressing all textures.*




**(a)** *RMS Color Error* **(b)** *CIELAB $\Delta E_{94}$ Difference.* **(c)** *SSIM on $L^*$ of CIELAB.*

**Figure 4:** *Urban Guy error metric histograms. Only the ASTC algorithms are shown and the error metric is on compressing all textures.*




**(a)** *RMS Color Error* **(b)** *CIELAB $\Delta E_{94}$ Difference.* **(c)** *SSIM on $L^*$ of CIELAB.*

**Figure 5:** *Japanese Castle error metric histograms. Only the ASTC algorithms are shown and the error metric is on compressing all textures.*

| Algorithm | Metric | Fire Hydrant | | | | | Urban Guy | | | | | Japanese Castle | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Min | Max | Median | Mean | $\sigma^2$ | Min | Max | Median | Mean | $\sigma^2$ | Min | Max | Median | Mean | $\sigma^2$ |
| ASTC 4×4 | RMS Color | 0.0009 | 0.0110 | 0.0015 | 0.0017 | 0.00000069 | 0.0004 | 0.0135 | 0.0009 | 0.0011 | 0.00000076 | 0.0018 | 0.0080 | 0.0040 | 0.0044 | 0.00000196 |
| | $\Delta E_{94}$ | 0.0049 | 0.4076 | 0.0155 | 0.0249 | 0.00104390 | 0.0016 | 0.5002 | 0.0108 | 0.0195 | 0.00113213 | 0.0234 | 0.3780 | 0.1043 | 0.1219 | 0.00451420 |
| | SSIM | 0.9905 | 0.9982 | 0.9958 | 0.9957 | 0.00000071 | 0.9913 | 0.9990 | 0.9982 | 0.9980 | 0.00000054 | 0.9808 | 0.9990 | 0.9971 | 0.9959 | 0.00001542 |
| ASTC 8×8 | RMS Color | 0.0025 | 0.0236 | 0.0039 | 0.0044 | 0.00000365 | 0.0010 | 0.0525 | 0.0021 | 0.0028 | 0.00000827 | 0.0035 | 0.0175 | 0.0098 | 0.0099 | 0.00000837 |
| | $\Delta E_{94}$ | 0.0116 | 1.0105 | 0.0370 | 0.0580 | 0.00570639 | 0.0033 | 1.7083 | 0.0199 | 0.0404 | 0.00809949 | 0.0332 | 0.8371 | 0.2093 | 0.2360 | 0.01864330 |
| | SSIM | 0.9309 | 0.9864 | 0.9694 | 0.9694 | 0.00003179 | 0.8605 | 0.9925 | 0.9866 | 0.9833 | 0.00010744 | 0.8850 | 0.9920 | 0.9775 | 0.9700 | 0.00053054 |
| ASTC 12×12 | RMS Color | 0.0037 | 0.0320 | 0.0059 | 0.0067 | 0.00000689 | 0.0013 | 0.0658 | 0.0031 | 0.0041 | 0.00001363 | 0.0042 | 0.0266 | 0.0134 | 0.0137 | 0.00001624 |
| | $\Delta E_{94}$ | 0.0178 | 1.3582 | 0.0562 | 0.0848 | 0.01049680 | 0.0046 | 2.1879 | 0.0271 | 0.0522 | 0.01294340 | 0.0373 | 1.1837 | 0.2811 | 0.3156 | 0.03478230 |
| | SSIM | 0.8835 | 0.9688 | 0.9344 | 0.9347 | 0.00010633 | 0.7337 | 0.9833 | 0.9690 | 0.9639 | 0.00035621 | 0.8343 | 0.9812 | 0.9538 | 0.9472 | 0.00103269 |

**Table 1:** *Error metric statistics for the compression algorithms.*

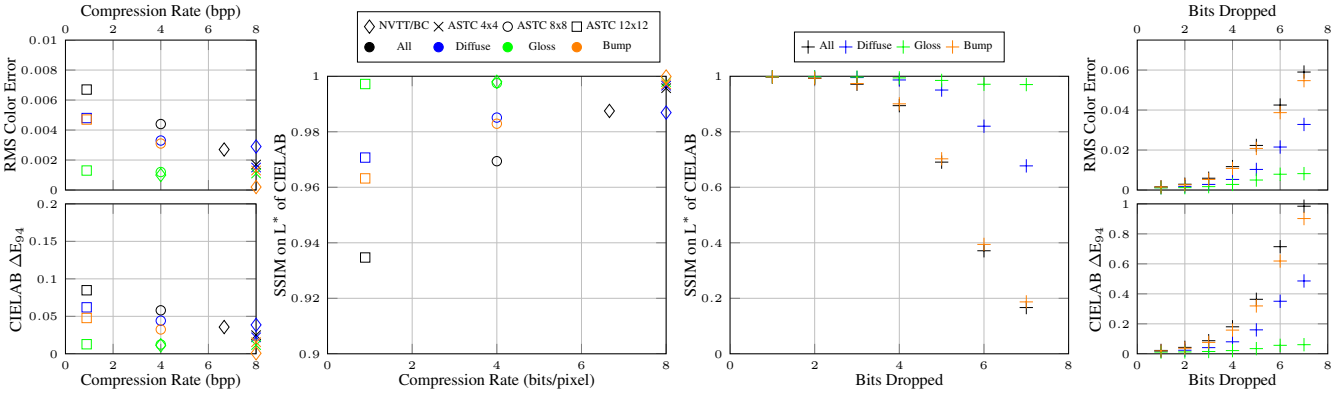



**Figure 6:** *Fire Hydrant mean error. The outer four vertical plots are color error against compression rate or bits dropped and the inner plots are SSIM. Note the vertical error scales **are different** across all eight plots to improve readability. See Section 4.3 for discussion.*

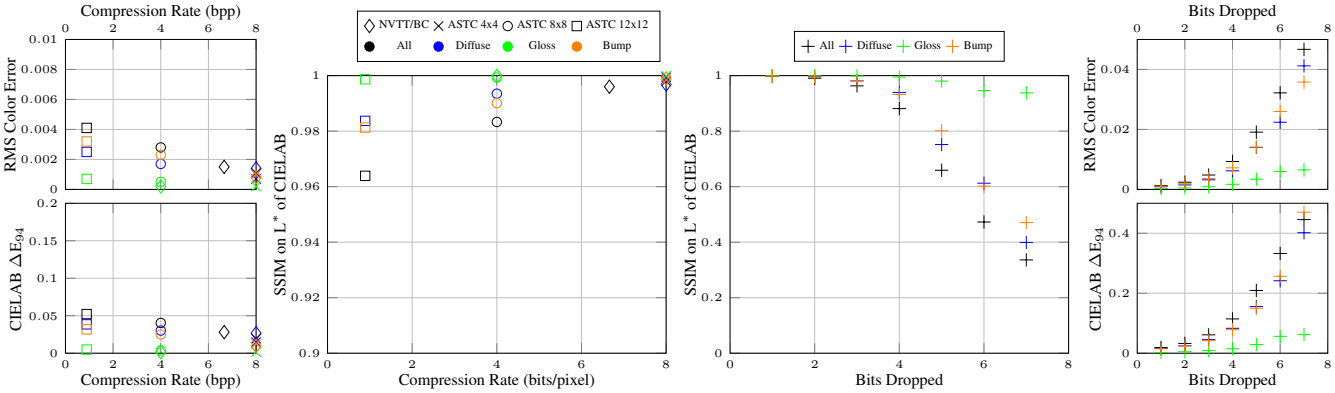



**Figure 7:** *Urban Guy mean error. The outer four vertical plots are color error against compression rate or bits dropped and the inner plots are SSIM. Note the vertical error scales **are different** across all eight plots to improve readability. See Section 4.3 for discussion.*

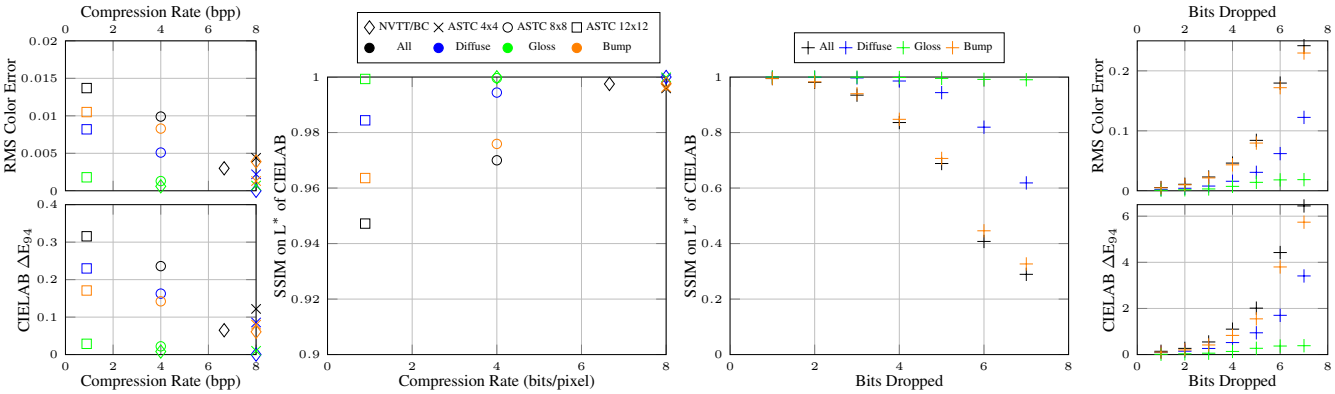
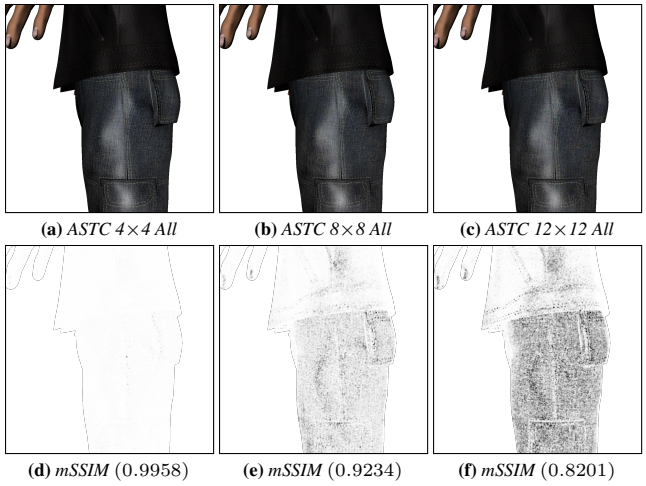



**Figure 8:** *Japanese Castle mean error. The outer four vertical plots are color error against compression rate or bits dropped and the inner plots are SSIM. Note the vertical error scales **are different** across all eight plots to improve readability. See Section 4.3 for discussion.*

**(a)** *Uncompressed* **(b)** *Drop 4* **(c)** *Uncompressed* **(d)** *Drop 4*

**(a)** & **(b):** *Diffuse Texture* **(c)** & **(d):** *Rendered Image*

**Figure 9:** *Zoomed view of diffuse texture (left) and rendered image (right). Notice the color banding in the diffuse texture is masked in the final rendered image due to the high frequency bump map.*



**(a)** *ASTC 4×4 All* **(b)** *ASTC 8×8 All* **(c)** *ASTC 12×12 All*

**(d)** *mSSIM* (0.9958) **(e)** *mSSIM* (0.9234) **(f)** *mSSIM* (0.8201)

**Figure 10:** *Visual comparison of a single compressed viewpoint. Figures (d)–(f) are the SSIM maps with the mean SSIM.*

pression rate for the NVTT/BC and ASTC algorithms. The left inner plot is SSIM against compression rate. The shape of the points corresponds to compression algorithm and the color to variations in compressed textures. For NVTT/BC compression rate, the All points are average bit rate at each pixel. The right plots are SSIM and color error against dropping bits from each color plane and the color corresponds to the variations in "compressed" textures.

For the RMS Color Error, CIELAB $\Delta E_{94}$ Difference, RMSE, and RMS Angular Error metrics, lower is better and 0.0 is a perfect match. For For the SSIM metric, higher is better and 1.0 is a perfect match. Note the vertical error scales in Figures 7, 6, and 8 **are different** across all eight plots to improve readability.

**Comparing final rendered images is stable across viewpoints.** As the histograms in Figures 3, 4, and 5 illustrate, the error metrics do vary with the viewpoint. We can conclude that these differences are due to geometric and texture set masking and can only be accounted for by evaluating final rendered images. The variation also indicates that a single viewpoint cannot be used to for the evaluation and that the viewpoint space must be sampled. Finally, the small variance in Table 1 indicates that each error metric can be averaged over the set of viewpoints resulting in a single error value.

**Geometric and texture set masking effects are present.** Figure 9 is a visual example of masking of the diffuse texture. Notice the color banding caused by dropping four bits in Figure 9b is masked in Figure 9d by the high frequency bump map.

Additionally, In Table 2, the **Sampled Viewpoint Mean Error** columns indicate that NVTT/BC has significantly higher quality than ASTC 12×12. However, looking at the **Individual Texture Error** Bump Angular RMSE, we would conclude that NVTT/BC has lower quality than ASTC 12×12. These results indicate masking is occurring. However, since we render final images using diffuse, gloss, and bump maps, it is unclear whether this masking is caused by geometric masking or texture set masking. We leave the relative masking effects between the two cases for future work.

**Perceptual sensitivity to texture classes.** Table 2 and Figures 6, 7, and 8 also reveal that perceptual sensitivity to compression artifacts and masking effects varies based on the class of texture. The Gloss only columns show very little reduction in quality from the ASTC 4×4 block size to the 12 × 12 block size. Likewise, dropping 4 bits from the gloss texture still results in very good final rendered image quality.

**Current GPU compression algorithms are too conservative.** Figures 10, 11, and 12 show several individual viewpoints of the models with various compression algorithms and corresponding SSIM maps. As these visual comparisons illustrate, geometric and texture set masking effects can be leveraged to reduce the bit rate of compressed textures while still maintaining good final rendered image quality. In particular, Figure 11 implies SSIM values as low as 0.9 still result in good image quality.

To explore how far compression can be pushed, Figure 13 visually shows the effects of dropping from 1–7 bits from each color plane for all textures at a single viewpoint and the SSIM map and mean.

## 5 Conclusion

In these results we can see evidence of geometric and texture set masking. These masking effects cannot be accounted for when evaluating individual textures and are only present when rendering from a viewpoint. Additionally, the masking effects vary based on the viewpoint and so evaluating a single viewpoint is not sufficient. Our approach of sampling the viewpoint space and comparing final rendered images accounts for both the masking effects and the viewpoint variation. By using final rendered images, the evaluation can use perceptually rigorous objective image quality assessment metrics which match how humans perceive image distortions.

Using our evaluation method, we can conclude that perceptual sensitivity to compression artifacts and masking effects varies with the type of texture, such as diffuse, gloss or bump maps. This implies that these different texture classes can be compressed more efficiently depending on the perceptual sensitivity. Our results also illustrate that current GPU compression algorithms are too conservative. The bit rates in these algorithms can be reduced while still maintaining final rendered image quality.

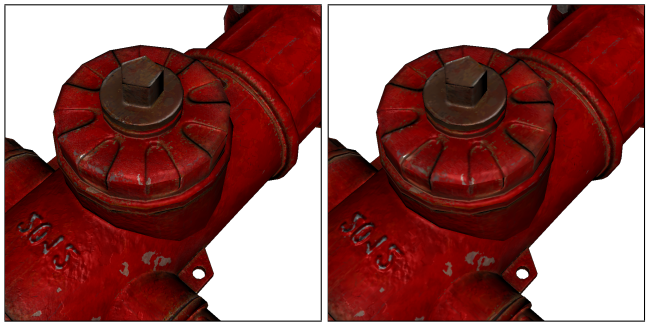
We have several ideas for further research in this area. Based on the statistics of the per-viewpoint error metrics, kernel density estimation could be used to estimate the probability distribution function of the error metric. The probability density can then be used for importance sampling of the viewpoint space. This would improve the efficiency of our algorithm by primarily sampling viewpoints that strongly contribute to the final mean error metric. We would also like to determine the relative effects between geometric masking and texture set masking and explore more interesting compression algorithms, such as variable bit rate algorithms.

## Acknowledgments

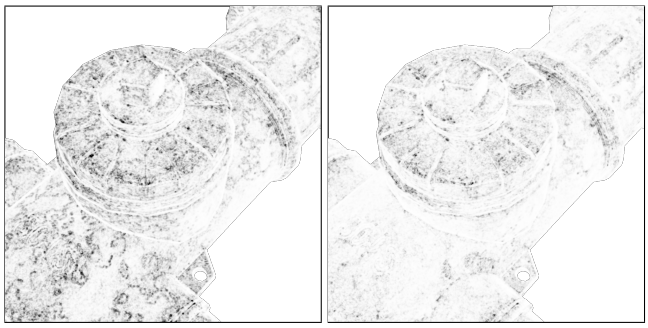
Thanks to the reviewer comments which improved this paper. The Urban Guy model is by Inventerion Productions from Turbo Squid.

| | | Sampled Viewpoint Mean Error | | | | | | | | | | | | Individual Texture Error | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | RMS Color Error | | | | CIELAB $\Delta E_{94}$ Difference | | | | SSIM on $L^*$ of CIELAB | | | | Diffuse | Gloss | Bump |
| Model | Algorithm | All | Diffuse | Gloss | Bump | All | Diffuse | Gloss | Bump | All | Diffuse | Gloss | Bump | $\Delta E_{94}$ | RMSE | aRMSE† |
| Fire Hydrant | NVTT/BC | 0.0027 | 0.0029 | 0.0010 | 0.0002 | 0.0357 | 0.0386 | 0.0114 | 0.0007 | 0.9875 | 0.9859 | 0.9980 | 0.9999 | 2.3904 | 0.0101 | 1.6464 |
| | ASTC 4×4 | 0.0017 | 0.0015 | 0.0011 | 0.0013 | 0.0249 | 0.0220 | 0.0117 | 0.0155 | 0.9957 | 0.9966 | 0.9979 | 0.9971 | 1.2364 | 0.0091 | 1.3375 |
| | ASTC 8×8 | 0.0044 | 0.0033 | 0.0012 | 0.0031 | 0.0580 | 0.0442 | 0.0124 | 0.0325 | 0.9694 | 0.9851 | 0.9975 | 0.9829 | 2.8451 | 0.0352 | 1.3497 |
| | ASTC 12×12 | 0.0067 | 0.0048 | 0.0013 | 0.0047 | 0.0848 | 0.0620 | 0.0128 | 0.0478 | 0.9347 | 0.9707 | 0.9972 | 0.9632 | 3.6900 | 0.0600 | 1.3561 |
| Urban Guy | NVTT/BC | 0.0015 | 0.0014 | 0.0002 | 0.0008 | 0.0281 | 0.0267 | 0.0014 | 0.0096 | 0.9960 | 0.9968 | 0.9999 | 0.9990 | 1.9768 | 0.0085 | 1.7431 |
| | ASTC 4×4 | 0.0011 | 0.0007 | 0.0002 | 0.0010 | 0.0195 | 0.0147 | 0.0016 | 0.0128 | 0.9980 | 0.9994 | 0.9999 | 0.9984 | 0.7348 | 0.0105 | 1.3063 |
| | ASTC 8×8 | 0.0028 | 0.0017 | 0.0005 | 0.0023 | 0.0404 | 0.0302 | 0.0038 | 0.0249 | 0.9833 | 0.9935 | 0.9993 | 0.9901 | 2.1219 | 0.0369 | 1.3181 |
| | ASTC 12×12 | 0.0041 | 0.0025 | 0.0007 | 0.0032 | 0.0522 | 0.0385 | 0.0049 | 0.0319 | 0.9639 | 0.9837 | 0.9987 | 0.9813 | 3.3601 | 0.0465 | 1.3247 |
| Japanese Castle | NVTT/BC | 0.0030 | 0.0000 | 0.0006 | 0.0030 | 0.0650 | 0.0000 | 0.0082 | 0.0609 | 0.9975 | 1.0000 | 0.9999 | 0.9976 | 0.0000 | 0.0057 | 1.5937 |
| | ASTC 4×4 | 0.0044 | 0.0022 | 0.0007 | 0.0039 | 0.1219 | 0.0857 | 0.0101 | 0.0799 | 0.9959 | 0.9995 | 0.9999 | 0.9963 | 0.8114 | 0.0060 | 1.3411 |
| | ASTC 8×8 | 0.0099 | 0.0051 | 0.0013 | 0.0083 | 0.2360 | 0.1627 | 0.0221 | 0.1421 | 0.9700 | 0.9944 | 0.9996 | 0.9759 | 2.0095 | 0.0202 | 1.3555 |
| | ASTC 12×12 | 0.0137 | 0.0082 | 0.0018 | 0.0105 | 0.3156 | 0.2299 | 0.0288 | 0.1710 | 0.9472 | 0.9844 | 0.9993 | 0.9636 | 2.7080 | 0.0269 | 1.3648 |

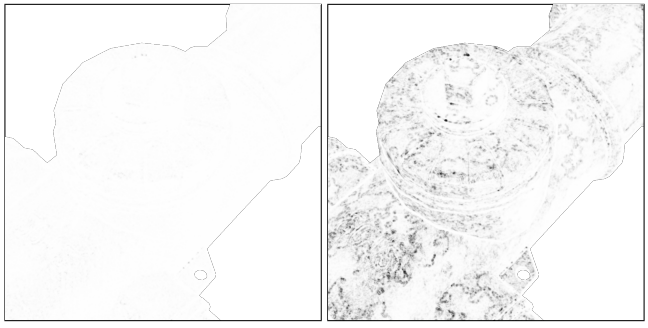
**Table 2:** *Mean error compared to no compression.* ***Sampled Viewpoint Mean Error*** *is the mean error over all viewpoints.* ***Individual Texture Error*** *is the mean error over all MIP levels of the largest-sized texture set. For RMS Color Error, $\Delta E_{94}$, RMSE and aRMSE, lower is better and 0.0 is a perfect match. SSIM is on the $L^*$ channel of the CIELAB color space and higher is better while 1.0 is a perfect match. The All column is all textures compressed, while the Diffuse, Gloss, and Bump columns are for compressing just that respective texture class. The Bump error aRMSE† is defined in Equation 2. NVTT/BC is Direct3D block compression and the three ASTC rows are the three different Adaptive Scalable Texture Compression block sizes. See Section 4.3 for further discussion.*



**(a)** *Uncompressed* **(b)** *All Textures Compressed*

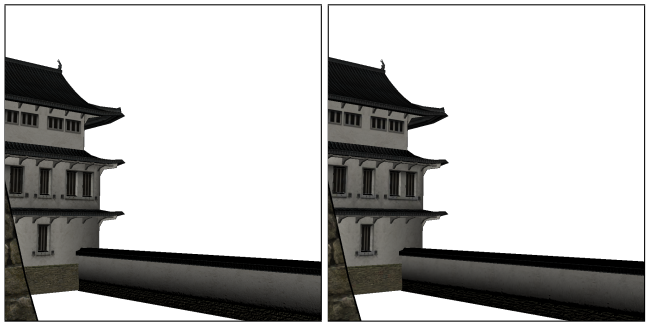


**(c)** *All* (mSSIM = 0.8908) **(d)** *Diffuse* (mSSIM = 0.9482)

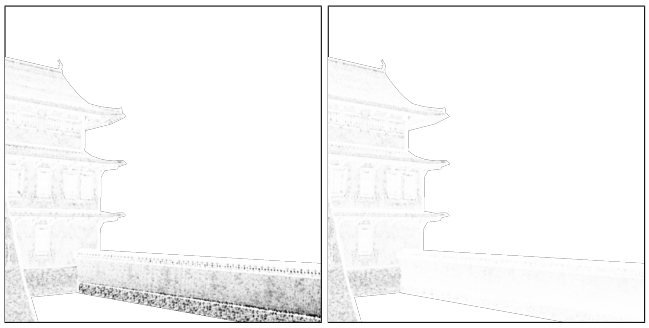


**(e)** *Gloss* (mSSIM = 0.9960) **(f)** *Bump* (mSSIM = 0.9414)

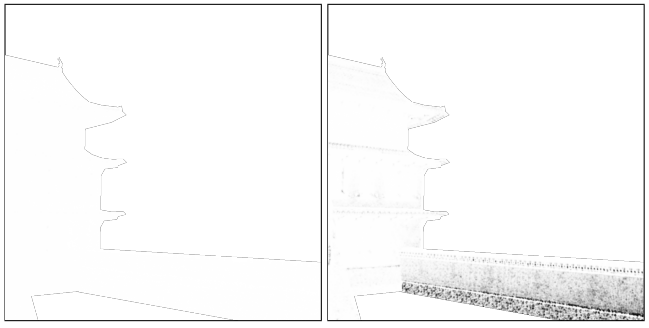
**Figure 11:** *Visual comparison of a single viewpoint of ASTC 12×12. Figures (c)–(f) are SSIM maps.*



**(a)** *Uncompressed* **(b)** *All Textures Compressed*

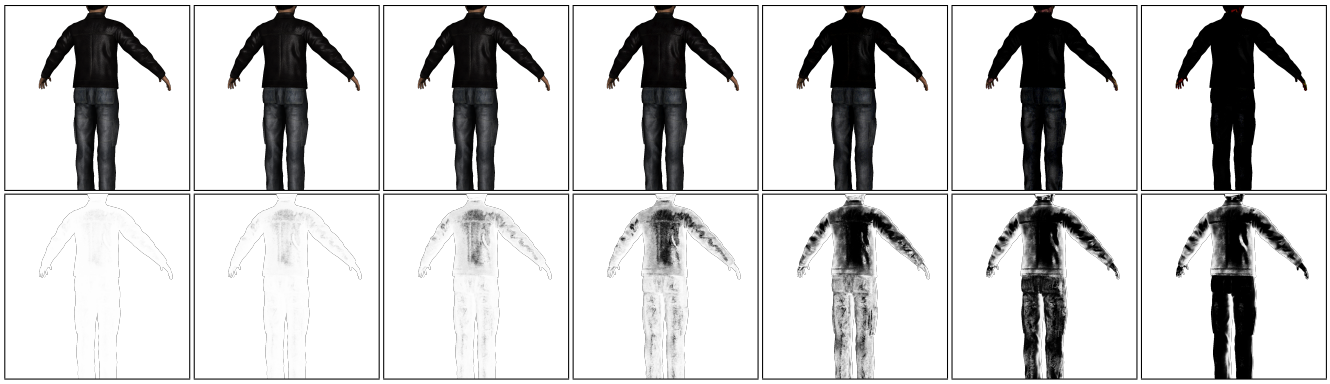


**(c)** *All* (mSSIM = 0.9159) **(d)** *Diffuse* (mSSIM = 0.9795)



**(e)** *Gloss* (mSSIM = 0.9999) **(f)** *Bump* (mSSIM = 0.9368)

**Figure 12:** *Visual comparison of a single viewpoint of ASTC 12×12. Figures (c)–(f) are SSIM maps.*



**(a)** *Drop 1* (0.9942) **(b)** *Drop 2* (0.9772) **(c)** *Drop 3* (0.9171) **(d)** *Drop 4* (0.7764) **(e)** *Drop 5* (0.5359) **(f)** *Drop 6* (0.3207) **(g)** *Drop 7* (0.2450)

**Figure 13:** *Visual comparison of dropping bits on all textures. The bottom row is the SSIM maps with the mean SSIM from this viewpoint.*

The Fire Hydrant model is by Trap Door and the Japanese Castle scene is by JD Creative Machine, both from the Unity Store.

## References


AJAGAMELLE, S. A., PEDERSEN, M., AND SIMONE, G. 2010. Analysis of the difference of Gaussians model in image difference metrics. In *Proc. 4th European Conf. Colour in Graphics Imaging, and Vision (CGIV)*, Society for Imaging Science and Technology, 489–496.

ARM, 2013. Encoder and Decoder Tool for Evaluation of ARM Adaptive Scalable Texture Compression (ASTC). Retrieved August 1, 2013 from `http://malideveloper.arm.com/develop-for-mali/tools/astc-evaluation-codec/`.

BEERS, A. C., AGRAWALA, M., AND CHADDHA, N. 1996. Rendering from compressed textures. In *Proceedings of SIGGRAPH 96, Annual Conference Series*, ACM, New York, USA, 373–378.

ČADÍK, M., HERZOG, R., MANTIUK, R., MYSZKOWSKI, K., AND SEIDEL, H.-P. 2012. New measurements reveal weaknesses of image quality metrics in evaluating graphics artifacts. *ACM Trans. Graph. 31*, 6 (Nov.), 147:1–147:10.

DELP, E. J., AND MITCHELL, O. R. 1979. Image compression using block truncation coding. *Communications, IEEE Trans. 27*, 9, 1335–1342.

FAIRCHILD, M. D. 2005. *Color Appearance Models*, 2nd ed. John Wiley & Sons, West Sussex, England.

FENNEY, S. 2003. Texture compression using low-frequency signal modulation. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, Eurographics Association, 84–91.

FERWERDA, J. A., SHIRLEY, P., PATTANAIK, S. N., AND GREENBERG, D. P. 1997. A model of visual masking for computer graphics. In *Proceedings of SIGGRAPH 97, Annual Conference Series*, ACM, New York, USA, 143–152.

HAO, P., AND SHI, Q. 2000. Comparative study of color transforms for image coding and derivation of integer reversible color transform. In *Pattern Recognition, Proceedings of the 15th International Conference on*, vol. 3, 224–227.

HEIDRICH, W., AND SEIDEL, H.-P. 1998. View-independent environment maps. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, ACM, New York, US, 39–45.

IORCHA, K., NAYAK, K., AND HONG, Z., 1999. System and method for fixed-rate block-based image compression with inferred pixel values. US Patent #5956431.

JOE, S., AND KUO, F. Y. 2003. Remark on algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Trans. Math. Softw. 29*, 1, 49–57.

KHRONOS GROUP, 2013. KHR_texture_compression_astc_ldr. Retrieved October 1, 2013 from `http://www.opengl.org/registry/`.

LINDSTROM, P., AND TURK, G. 2000. Image-driven simplification. *ACM Trans. Graph. 19*, 3 (July), 204–241.

MICROSOFT. 2013. Block Compression (Direct3D 10). In *Programming Guide for Direct3D 10*. MSDN.

NVIDIA, 2013. NVIDIA Texture Tools. Retrieved August 1, 2013 from `http://code.google.com/p/nvidia-texture-tools/`.

OLANO, M., AND BAKER, D. 2010. LEAN mapping. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, New York, US, 181–188.

OLANO, M., BAKER, D., GRIFFIN, W., AND BARCZAK, J. 2011. Variable bit rate GPU texture compression. *Computer Graphics Forum 30*, 4, 1299–1308.

STRÖM, J., AND AKENINE-MÖLLER, T. 2005. iPACKMAN: high-quality, low-complexity texture compression for mobile phones. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, ACM, New York, US, 63–70.

STRÖM, J., AND PETTERSSON, M. 2007. ETC2: texture compression using invalid combinations. In *Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, Eurographics Association, 49–54.

WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Trans. 13*, 4, 600–612.