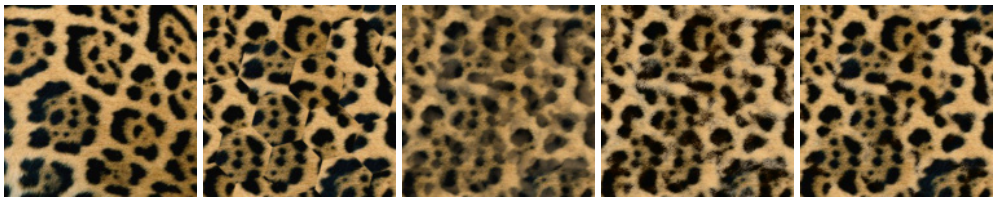
On Histogram-preserving Blending for Randomized Texture Tiling

Brent Burley
Walt Disney Animation Studios

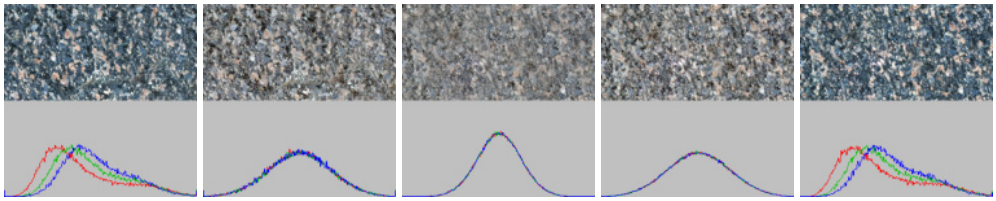


(a) Input (b) Tiled (c) Linearly blended (d) Histogram preserved (e) Exponentiated

Figure 1. An input texture (a) is randomly tiled (b). Overlapping the tiles with linear blending (c) removes the seams but suffers ghosting and contrast loss. Histogram-preserving blending (d) preserves the contrast. Using exponentiated blending weights (e) reduces ghosting artifacts and better preserves structured texture details. Our implementation is per-channel and avoids lengthy precomputation. Our formulation avoids clipping artifacts by using the truncated Gaussian distribution with a soft-clipping contrast operator. For textures that would exhibit coloration artifacts (such as this one), we blend in YCbCr color space and perform histogram preservation only on the luminance. [Input photograph by Steve Winter, National Geographic Creative.]

Abstract

To support interactive authoring of high-resolution randomly tiled textures, we modify the histogram-preserving tiling algorithm of Heitz and Neyret to avoid any lengthy preprocessing. Instead of calculating a 3D histogram transformation by optimal transport, which can take minutes even at low resolution, the input texture is transformed using per-channel 1D look-up tables, constructed trivially from the input histogram on texture load. Three sources of clipping are described. Modifying the algorithm to use a truncated Gaussian distribution and a novel soft-clipping contrast operator avoids clipping artifacts while retaining very high rendering performance. Per-channel histogram preservation is sufficient for most textures, but some will produce unwanted colorations; these can often be avoided by performing histogram preservation only on luminance. Exponentiating the blending weights can reduce ghosting artifacts and better preserve structured texture details.



(a) Input (b) Gaussianized (c) Linearly blended (d) Contrast restored (e) Histogram restored

Figure 2. Illustration of histogram-preserving blending. The input texture (a) is first transformed (using our per-channel implementation) to have a Gaussian histogram (b). After randomized tiling with linear blending (c), the histogram is still Gaussian but with reduced variance and spatially inconsistent contrast. A local contrast operator restores the original variance (d), and inverting the Gaussianization transformation restores the original histogram (e). [The input texture is from [Heitz and Neyret 2018]; only a portion is shown.]

1. Introduction

Texture maps are a critical component of production in our studio, with each 3D surface typically having dozens of texture layers controlling its appearance. Due to the vastness of our worlds, mapping unique high-resolution texture detail for every surface would require a prohibitive amount of memory, and hand-painting every surface would require a prohibitive amount of authoring time. Instead, artists hand-paint textures only where needed and apply tiled textures everywhere else. To break up repetition, artists typically apply tiled textures in two or more layers which are offset and overlapped, but blended regions can appear blurry and suffer reduced contrast as in Figure 1(c).

Heitz and Neyret [2018] proposed randomized texture tiling using a histogram-preserving blending operator to preserve the contrast in blended texture regions. They tiled the image plane with a virtual triangle lattice, randomly selected a texture tile at each lattice point, and then blended the three nearest textures across each triangle using barycentric weights. To preserve contrast, they recalled that blending statistically independent samples has the effect of convolving their histogram, that convolving Gaussian distributions produces a Gaussian distribution with reduced variance, and that the variance lost due to linear blending can be restored using linear scaling around the expected value. To leverage this, they transformed the input texture to have a Gaussian histogram before blending. After blending, they restored the variance of the Gaussian distribution using a per-pixel adjustment based on the blending weights, and then inverted the Gaussianization to restore the original histogram. The histogram-preserving blending process is depicted in Figure 2.

To avoid coloration artifacts, Heitz and Neyret performed the histogram transformation as a 3D process. To transform the input texture to have a 3D Gaussian

distribution, they first generated a texture with uncorrelated Gaussian color noise of the same resolution as the input, and then rearranged the texels to have the same structure as the input using a 3D optimal transport algorithm to minimize the squared difference between the two images. After linear blending and scaling to restore the original variance, they recovered the original histogram using a 3D LUT which they constructed using 3D optimal transport in the reverse direction. Heitz and Neyret reported that the generation of the Gaussianized input texture and 3D LUT took several minutes for a 256×256 texture.

Heitz and Neyret's results are impressive, but our requirements differ. In some ways, ours are more demanding; in others less so:

- To support our interactive authoring workflows, we must process texture edits within a fraction of a second.
- Our tiled texture resolutions are typically 4096×4096 or higher.
- Most of our textures are grayscale (used as masks to blend between layers or modulate material properties).
- Most of our textures are 8-bit (which has no impact on the implementation except that it exacerbates clipping artifacts as shown in Section 3).
- We prefer to implement all transformations using 1D LUTs for rendering efficiency.

Based on our requirements, we perform 1D (per-channel) histogram preservation which is trivial compared to 3D histogram preservation and requires only minimal preprocessing time. We were pleasantly surprised that only a small minority of color textures exhibited coloration artifacts. However, we did encounter two artifacts unrelated to color: we found that using the infinite Gaussian distribution of Heitz and Neyret can introduce clipping artifacts in some textures, and we found that textures with strong structure can exhibit ghosting artifacts. In our implementation, we aim to reduce or eliminate these artifacts while avoiding lengthy preprocessing.

The remainder of this paper is structured as follows. In Section 2 we describe how histogram-preserving blending can be performed using 1D LUTs which can be trivially built on texture load. In Section 3 we discuss sources of clipping and describe how using the truncated Gaussian distribution with our soft-clipping contrast operator can avoid clipping artifacts. In Section 4 we investigate unwanted colorations in a problematic texture and show that blending in the YCbCr color space and performing histogram preservation only on the luminance can avoid coloration artifacts while sacrificing some color contrast. In Section 5 we show that exponentiating the blending weights can reduce ghosting artifacts and better preserve structured texture details. In

Section 6 we detail our implementation. Finally, we present and discuss our results in Section 7.

All of the figures in this paper were generated using our per-channel implementation, and unless otherwise noted, used our truncated Gaussian formulation and soft-clipping contrast operator. YCbCr blending and exponentiated blending are optional features and were only used in figures where indicated.

2. Per-channel Histogram-preserving Blending

In this section, we describe naive per-channel (1D) histogram-preserving blending, which we then refine in subsequent sections.

Gaussianization. Transforming samples through the CDF of a given distribution results in a uniform distribution; likewise, transforming uniformly distributed samples through the inverse CDF results in the given distribution, and transforming between arbitrary distributions can be achieved by combining these two operations [Bergen and Heeger 1995].

To Gaussianize the texture, we first transform each texel through the CDF of the input histogram to produce a uniform distribution and then transform to the Gaussian distribution using the Gaussian distribution's inverse CDF; we combine these two steps into a single 1D LUT per channel.

The *infinite* Gaussian distribution as used in [Heitz and Neyret 2018], its CDF, and inverse CDF are, respectively,

$$\begin{aligned}
\mathcal{H}_G(x;\sigma) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\frac{1}{2})^2}{2\sigma^2}\right), \\
\mathrm{CDF}_G(x;\sigma) &= \frac{1}{2}\left(1+\mathrm{erf}\left(\frac{2x-1}{2\sqrt{2\sigma^2}}\right)\right), \\
\mathrm{CDF}_G^{-1}(x;\sigma) &= \frac{1}{2}+\sqrt{2\sigma^2}\ \mathrm{erfinv}\,(2x-1)
\end{aligned}$$

where σ^2 is the variance. We use the recommended $\sigma = \frac{1}{6}$ from Heitz and Neyret.

Linear blending. As in [Heitz and Neyret 2018], we linearly blend texel values $x_1 \dots x_N$ using weights $w_1 \dots w_N$ (which are assumed to sum to one):

$$\hat{x} = w_1 x_1 + \cdots + w_N x_N. \tag{1}$$

Each x represents a single value, with operations repeated per channel for multi-channel textures. With the randomized tiling of Heitz and Neyret, $N = 3$, but the method is applicable to any N. [We use the hat symbol to denote blended values and distributions such as $\hat{x}$ and $\widehat{G}$.]

Contrast restoration. A linearly blended distribution is a convolution of the distribution with itself. The convolution of linearly blended Gaussian distributions is a Gaussian distribution with reduced variance:

$$\begin{aligned}\mathcal{H}_{\widehat{G}}(\hat{x};\sigma) &= \frac{1}{w_1}\mathcal{H}_G\left(\frac{x}{w_1};\sigma\right) * \cdots * \frac{1}{w_N}\mathcal{H}_G\left(\frac{x}{w_N};\sigma\right) \\ &= \mathcal{H}_G\left(\hat{x};\sigma W\right);\ \ W = \sqrt{w_1^2 + \cdots + w_N^2},\end{aligned} \tag{2}$$

and therefore,

$$\mathrm{CDF}_{\widehat{G}}\left(\hat{x};\sigma\right) = \mathrm{CDF}_G\left(\hat{x};\sigma W\right).$$

The contrast restoration operator, which can be derived by transforming through the CDFs, becomes a simple linear expansion:

$$\begin{aligned}S_{\widehat{G}}(\hat{x};\sigma,W) &= \mathrm{CDF}_G^{-1}\left(\mathrm{CDF}_{\widehat{G}}\left(\hat{x};\sigma\right);\sigma\right) \\ &= \mathrm{CDF}_G^{-1}\left(\mathrm{CDF}_G\left(\hat{x};\sigma W\right);\sigma\right) \\ &= \frac{1}{W}\left(\hat{x} - \frac{1}{2}\right) + \frac{1}{2}.\end{aligned}$$

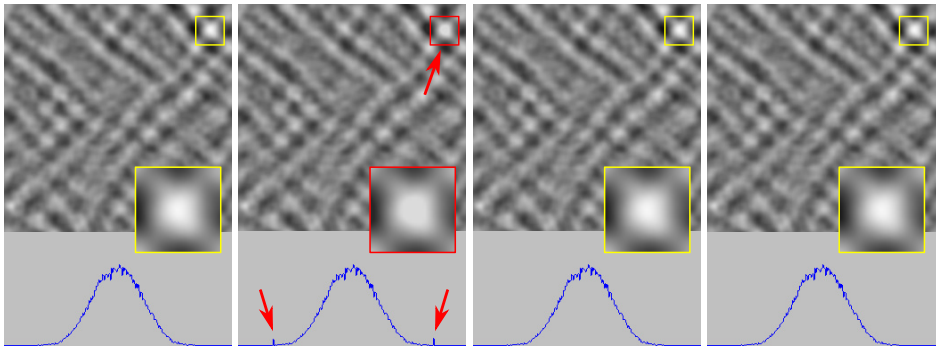
Histogram restoration. To restore the original histogram after blending, we simply transform through the inverse of the 1D LUTs we used for Gaussianization.

3. Avoiding Clipping Artifacts

We encountered three sources of clipping artifacts in our naive implementation. The first is due to the fact that the Gaussian distribution is defined over $(-\infty,\infty)$ and the Gaussianization transformation maps some of the texels to values outside of $[0,1]$ which then get clipped when stored in a fixed-point texture format. Though the percentage of affected texels is small (roughly 0.3%, or ~ 3000 pixels in a 1024×1024 texture), artifacts can be visible if the clipped texels are concentrated in one part of the image as in Figure 3(b).

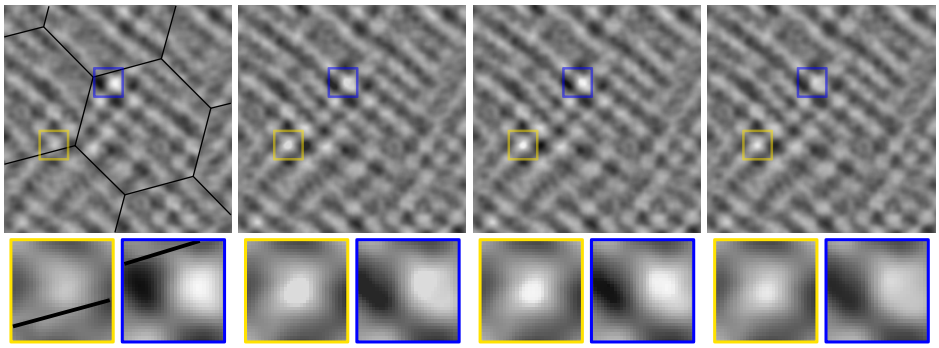
Another source of clipping, illustrated in Figure 4(b), occurs when the linear contrast operator pushes blended texel values beyond $[0,1]$ which then get clipped if the inverse Gaussianization is implemented with a LUT (which is typically specified only for values within $[0,1]$). For instance, when blending three textures with equal weights, $S_{\widehat{G}}$ produces values outside of $[0,1]$ for any $\hat{x} \lessapprox 0.21$ or $\hat{x} \gtrapprox 0.79$. This could affect as many as 8.2% of the blended pixels where features happen to align on adjacent tiles.

Given that CDF_G maps $(-\infty,\infty)$ back to $[0,1]$, both forms of clipping would be avoided if a floating-point texture were used and inverse Gaussianization were performed analytically, but doing so would come at some cost. And even though no values would be clipped, the result could still *appear* clipped as illustrated in Figure 4(c). This third type of clipping artifact occurs when the contrast operator expands



(a) Input (b) Fixed-point texture (c) Floating-point texture (d) Truncated Gaussian

Figure 3. Illustration of clipping artifacts resulting from storing a texture with an infinite Gaussian histogram in a fixed-point format. An input texture (a), shown with its native histogram, is Gaussianized and then restored without tiling or blending (b-d). Clipping occurs if the intermediate Gaussianized texture is represented in fixed-point (b), resulting in spikes in the histogram and visible artifacts, indicated by red arrows. Representing the intermediate Gaussianized texture in floating-point (c) avoids the clipping. Using the truncated Gaussian distribution (d) avoids the clipping without requiring a floating-point texture. [The input texture is from [Heitz and Neyret 2018, suppl.]]



(a) No blending (b) Clipping due to inverse LUT (c) Apparent clipping due to $S_{\widehat{G}}$ (d) Our soft-clipping operator, $S^*_{[\widehat{G}]}$

Figure 4. Illustration of clipping artifacts resulting from the linear expansion of contrast operator $S_{\widehat{G}}$. The texture from Figure 3 is randomly tiled, shown with no blending (a) and histogram-preserving blending (b-d). Even though a floating-point texture is used here, artifacts can still be introduced when $S_{\widehat{G}}$ pushes values beyond $[0, 1]$ that get clipped by the inverse Gaussianization LUT (b). And even if inverse Gaussianization is implemented analytically, the result can still appear clipped (c) where $S_{\widehat{G}}$ compresses texture details into the tails of the distribution. Using the truncated Gaussian distribution with our soft-clipping contrast operator (d) allows the use of a fixed-point texture and inverse LUT while avoiding these artifacts and otherwise preserving the blended result.

texture details into the tails of the distribution, where they are strongly compressed by the inverse Gaussianization, giving them a flattened appearance.

In our implementation, we avoid the first two types of clipping artifacts by using the *truncated* Gaussian distribution that maps values strictly within $[0, 1]$, and we are thus permitted to use fixed-point textures and LUTs without penalty. We address the third type of clipping artifact by using a *soft-clipping* contrast operator. We describe the truncated Gaussian distribution and present our soft-clipping contrast operator in the following subsections.

3.1. Truncated Gaussian Distribution.

The truncated Gaussian distribution, denoted $[G]$, is defined strictly within $[0, 1]$:

$$\begin{aligned}
\mathcal{H}_{[G]}(x;\sigma) &= \begin{cases} \frac{C(\sigma)}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\frac{1}{2})^2}{2\sigma^2}\right); C(\sigma) = \frac{1}{\operatorname{erf}\left(\frac{1}{2\sqrt{2\sigma^2}}\right)} & 0 \le x \le 1 \\ 0 & \text{otherwise,} \end{cases} \\
\mathrm{CDF}_{[G]}(x;\sigma) &= \frac{1}{2}\left(1 + C(\sigma)\operatorname{erf}\left(\frac{2x-1}{2\sqrt{2\sigma^2}}\right)\right), \\
\mathrm{CDF}^{-1}_{[G]}(x;\sigma) &= \frac{1}{2} + \sqrt{2\sigma^2}\ \operatorname{erfinv}\left(\frac{1}{C(\sigma)}(2x-1)\right). \qquad (3)
\end{aligned}$$

This differs from the infinite Gaussian distribution only in the constraint over the domain of x and the normalization constant $C(\sigma)$ that accounts for the truncation. As before, we use $\sigma = \frac{1}{6}$.

The linear contrast operator, $S_{\widehat{G}}$, no longer applies as it generates values beyond $[0, 1]$ that are undefined for $\mathrm{CDF}_{[G]}$. Instead, the contrast operator for the truncated Gaussian distribution, $S_{[\widehat{G}]}$, must be rederived using the convolution of truncated Gaussian distributions.

Note that $S_{[\widehat{G}]}$ does not have a practical closed form and must be approximated. Rather than approximating $S_{[\widehat{G}]}$ precisely, which would only serve to reproduce the tail compression artifact from Figure 4(c), we take this as an opportunity to reduce the flattening by introducing a "soft-clipping" contrast operator, $S^*_{[\widehat{G}]}$.

3.2. Soft-clipping Contrast Operator

As can be observed in Figure 5(left, black), $S_{[\widehat{G}]}$ is nearly linear for the middle portion of the curve (matching the slope of $S_{\widehat{G}}$), rolling off smoothly and becoming nearly flat as it approaches 0 or 1. For our approximation, $S^*_{[\widehat{G}]}$, we choose a piecewise function which is linear (and equal to $S_{\widehat{G}}$) for the middle half of the range (i.e. $S^*_{[\widehat{G}]} \in [\frac{1}{4}, \frac{3}{4}]$), and blends smoothly to 0 or 1 using a quadratic segment at the ends. As before, we compute W from the blend weights as in Equation (2). These constraints fully define

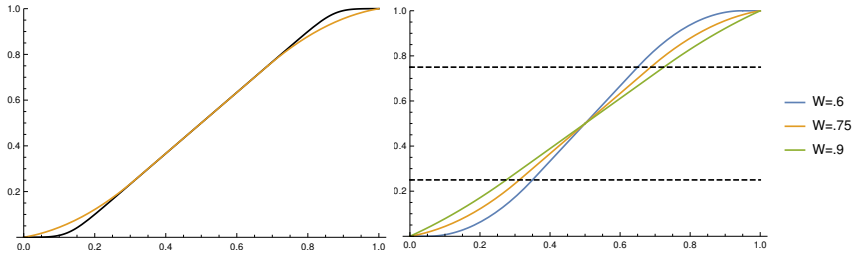


Figure 5. A comparison of our soft-clipping contrast operator, $S^*_{[\widehat{G}]}$, yellow, vs. $S_{[\widehat{G}]}$, black, for $W = .75$. (left) $S^*_{[\widehat{G}]}$ is shown for a range of W values; the portion between the dashed lines is linear and matches $S_{[\widehat{G}]}$ (right).

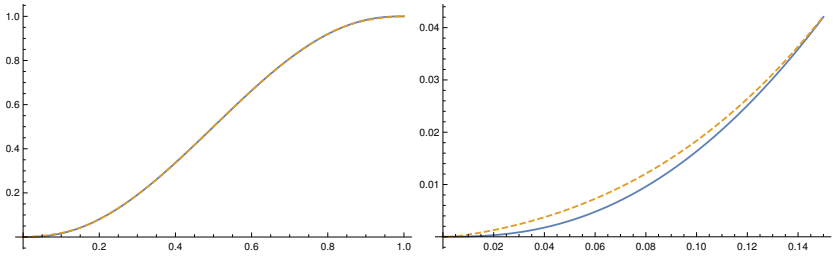


Figure 6. Plot comparing the effect of the infinite Gaussian contrast operator (blue) versus our soft-clipping operator (yellow, dashed) for $W = \sqrt{1/3}$, the minimum W for blending three textures. The two differ only on close inspection (right) where the soft-clipping operator exhibits less value compression near the ends of the distribution. [The actual functions plotted are $\mathrm{CDF}_G\left(S_{\widehat{G}}\left(\mathrm{CDF}^{-1}_{\widehat{G}}(\hat{x})\right)\right)$ and $\mathrm{CDF}_{[G]}\left(S^*_{[\widehat{G}]}\left(\mathrm{CDF}^{-1}_{[G]}(\hat{x})\right)\right)$ representing the aggregate effect of Gaussianization and contrast adjustment.]

our operator which becomes

$$S^*_{[\widehat{G}]}\left(\hat{x}|\hat{x} \le \frac{1}{2}; W\right) = \begin{cases} \frac{1}{W}\left(\hat{x} - \frac{1}{2}\right) + \frac{1}{2} & \hat{x} \ge \frac{2-W}{4} \\ 8\left(\frac{1}{W} - 1\right)\left(\frac{\hat{x}}{2-W}\right)^2 + \left(3 - \frac{2}{W}\right)\frac{\hat{x}}{2-W} & W \ge \frac{2}{3} \\ \frac{1}{W^2}\left(\hat{x} - \frac{2-3W}{4}\right)^2 & \hat{x} \ge \frac{2-3W}{4} \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

$$S^*_{[\widehat{G}]}\left(\hat{x}|\hat{x} > \frac{1}{2}; W\right) = 1 - S^*_{[\widehat{G}]}(1 - \hat{x}).$$

$S^*_{[\widehat{G}]}$ is plotted and compared with $S_{[\widehat{G}]}$ in Figure 5. When viewed in conjunction with the Gaussianization transformation, as illustrated in Figure 6, our truncated Gaussian formulation with soft clipping only deviates minimally from the infinite Gaussian result, and only at the ends of the distribution where it avoids becoming overly flat.

(a) Input (b) Linearly blended (c) Gaussianized (d) Contrast restored (e) YCbCr

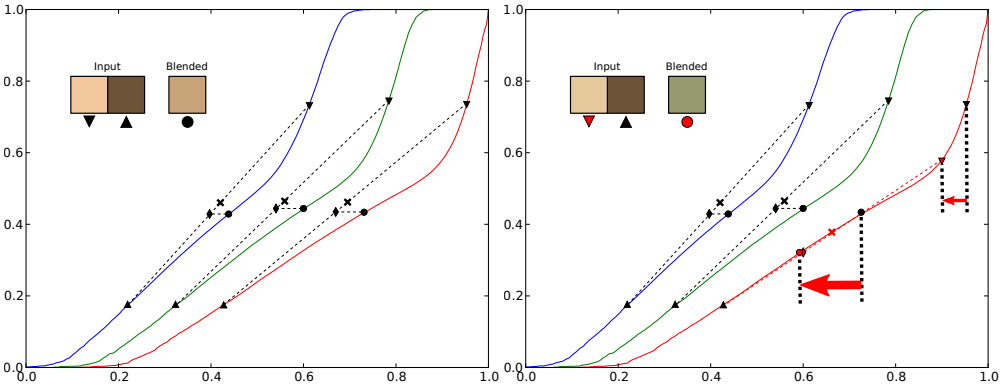
Figure 7. Example of coloration artifacts. A portion of an input texture with subtle red and green hue variation is shown in (a). Randomized tiling with linear blending in the native color space (i.e. without histogram preservation) shows contrast loss but no unusual coloration (b). Blending in a Gaussianized color space produces unexpectedly saturated red and green pixels (c). Applying the histogram-preserving contrast operator makes the red and green colorations even more pronounced (d). Blending in YCbCr color space using histogram-preservation only for Y avoids unwanted coloration (e). [The input texture is from [Heitz and Neyret 2018]]

4. Avoiding Coloration Artifacts

Simply blending color values in their native color space can produce colors that were not in the original texture; blending color values in a histogram-transformed space and performing per-channel contrast adjustments can make unexpected coloration even more likely. Three-dimensional histogram preservation as in [Heitz and Neyret 2018] avoids unexpected colors by remapping blended, contrast-adjusted colors to ones that occur within the original image. However, per-channel (1D) histogram preservation makes no such attempt and unexpected coloration may persist in the result.

Of the textures shown in [Heitz and Neyret 2018] (including the supplementary results), we only found the ones shown in Figures 7 and 11 to have visible coloration artifacts; the leopard texture in Figure 7 is the most obvious and even this one requires close inspection. In Figure 8, we examine the cause of the discolorations that occur with the leopard texture and show that the nonlinearity of the Gaussianization transform is amplifying color variations in the input image which are further amplified by the contrast operator.

We find that we can preserve the contrast while avoiding introducing coloration artifacts by linearly transforming to YCbCr and performing histogram-preserving blending only on the luminance (Y), using ordinary linear blending (non-histogram-preserving) on the chrominance channels. Assuming that blending the untransformed texture would produce no objectionable colors, linear blending in a linearly transformed space produces the same result as blending in the original color space, and thus we will have no unexpected hues. Preserving luminance contrast preserves most of the apparent contrast as human vision is more sensitive to luminance differences



Neutral Red decreased from 0.95 to 0.90

Figure 8. Illustration of discoloration cause. A pair of input colors from Figure 7 (triangles) is Gaussianized through the input LUT (red, green, blue curves), linearly blended with equal weight (dashed lines, resulting in X's), contrast adjusted (resulting in diamonds), and histogram-restored through the inverse LUT, producing the final colors (circles). For input colors that are neutral with respect to the histogram, the blended result is also neutral (left). When a small change is made to the red value of one of the inputs (small red arrow), the nonlinearity of the histogram transformation amplifies the color difference, and the contrast adjustment further increases the difference (large red arrow), resulting in an unexpected green hue (right).

than color differences. Figures 7 and 11 show that using YCbCr avoids coloration artifacts in those examples.

Because discoloration artifacts are rare and typically subtle, and because there is a possibility of a reduction in color contrast, we recommend to use YCbCr only as necessary.

5. Reducing Ghosting Artifacts

Even when the contrast is restored using histogram-preserving blending, image ghosting can still occur with randomized tiling, with details from overlapping tiles showing through in blended regions. We find that using exponentiated blending weights provides a simple and efficient means to reduce ghosting and preserve more structured detail from the source texture. To perform exponentiated blending we raise each weight to a power and renormalize the resulting weights:

$$w_i' = \frac{w_i^\gamma}{\sum_{i=1}^{N} w_i^\gamma}. \tag{5}$$

A plot of Equation (5) is shown in Figure 9, and an illustration of the effect for a range of exponent values is shown in Figure 10. Further results are shown in Figure 13.

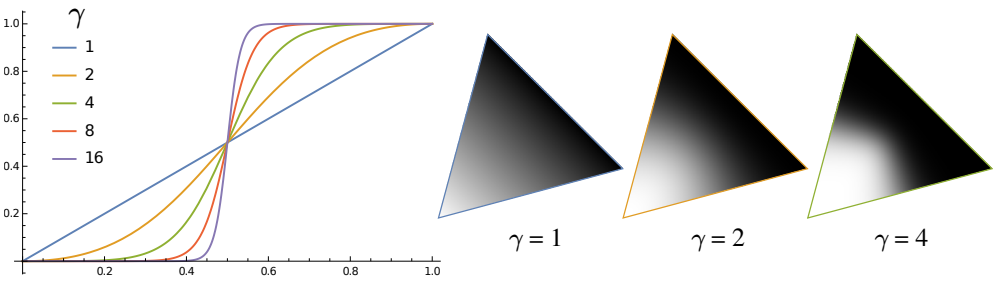


Figure 9. Exponentiated blending weight plotted for various values of γ. Left: w_1' plotted versus w_1 (with w_2=1−w_1 and w_3=0). Right: w_1' evaluated over triangle tile.

Figure 10. Blending using exponentiated blend weights with different values of γ. Ghosting is visible at $\gamma \leq 2$, and tile structure is visible at $\gamma \geq 8$. Blending with $\gamma = 4$ provides good structure preservation for this texture without revealing tile structure.

Algorithm 1 Gaussianizing a texture. (All operations are per-channel.)

for each texel x **do** ▷ compute histogram
  histogram$[x] \leftarrow$ histogram$[x] + 1$
end for
sum $= 0$
for i from 0 to LUTsize-1 **do** ▷ build LUT
  sum $\leftarrow$ sum + histogram$[i]$
  LUT$[i] \leftarrow$ sum
end for
for i from 0 to LUTsize-1 **do** ▷ normalize and Gaussianize LUT
  LUT$[i] \leftarrow \mathrm{CDF}^{-1}_{[G]}\left(\mathrm{LUT}[i]/\mathrm{sum}; \frac{1}{6}\right)$ ▷ Eq. (3)
end for
for each texel x **do**
  $x \leftarrow$ LUT$[x]$ ▷ apply LUT to texture
end for

6. Implementation

Algorithm 1 summarizes our Gaussianization process which we apply to the texture on load. Algorithm 2 summarizes our histogram-preserving blending process. In the equations, x is assumed to be $[0, 1]$ with appropriate scaling applied as needed (e.g., multiplying or dividing by 255 for an 8-bit texture).

If we are blending in YCbCr color space, we transform to YCbCr space before Algorithm 1, and perform Algorithm 1 only on the Y channel; in Algorithm 2 we perform contrast correction only on Y. Following Algorithm 2, we convert the blended color from YCbCr back to RGB as the final step.

Algorithm 2 Histogram-preserving blending of texels x_1, x_2, x_3 with weights w_1, w_2, w_3.

$w'_i \leftarrow w_i^\gamma / (w_1^\gamma + w_2^\gamma + w_3^\gamma)$; for $i \in \{1, 2, 3\}$ ▷ exponentiate (Eq. (5))
$\hat{x} \leftarrow w'_1 x_1 + w'_2 x_2 + w'_3 x_3$ ▷ linear blend (Eq. (1))
$W \leftarrow \sqrt{(w'_1)^2 + (w'_2)^2 + (w'_3)^2}$ ▷ compute variance scale factor (Eq. (2))
$\hat{x} \leftarrow S^*_{[\widehat{G}]}(\hat{x}; W)$ ▷ restore contrast (Eq.(4))
$\hat{x} \leftarrow \mathrm{LUT}^{-1}[\hat{x}]$ ▷ de-gaussianize

7. Results and Discussion

Clipping-free blending. With histogram-preserving blending using the infinite Gaussian distribution, clipping artifacts such as in Figure 3(b) can occur if the texture is stored in fixed-point representation, or as in Figure 4(b), if the inverse Gaussianization operator (CDF_G) is applied as a LUT. Even if CDF_G is evaluated analytically per-pixel (as in [Heitz and Neyret 2018, Eq. 22]), artifacts such as Figure 4(c) may still occur. Though the histogram is perfectly preserved globally, this is not true locally; when prominent features align on adjacent tiles, the assumption of statistically independent sampling does not hold. In the extreme, a texture may by chance align nearly exactly with itself on adjacent tiles in which case no variance will be lost during blending and the linear expansion of $S_{\widehat{G}}$ will induce either maximal clipping (if CDF_G is implemented as a LUT) or maximal detail compression (if CDF_G is implemented analytically).

Our truncated Gaussian formulation with soft-clipping avoids such clipping artifacts while permitting the use of a fixed-point texture and a LUT for inverse Gaussianization. Even when such clipping artifacts are few, using the soft-clipping operator does no harm, and the additional computation is modest, making it a robust approach. The soft-clipping operator also has a fairly simple form, and a more accurate approximation of $S_{[\widehat{G}]}$ would likely come at additional cost.

We note that clipping with the infinite Gaussian could be reduced through the use of a narrower Gaussian. For instance, $\sigma = \frac{1}{8}$ is sufficient to avoid the need for a floating-point texture for the artifact in Figure 3(b), but doing so sacrifices some dynamic range of the texture, and also does not improve the artifact in Figure 4(c).

As an aside, we found that histogram-preserving blending can also be performed using the uniform distribution. The main advantage of doing so is simplicity in that one can blend uniformly distributed samples directly, skipping the Gaussianization step. Blended values will have a non-uniform distribution, and applying an appropriate contrast operator can recover the uniform distribution; a contrast operator for blending three uniformly distributed samples is derived in the Appendix. An additional advantage of using the uniform distribution is that it intrinsically avoids generating values outside of $[0, 1]$. Though mathematically appealing, it unfortunately

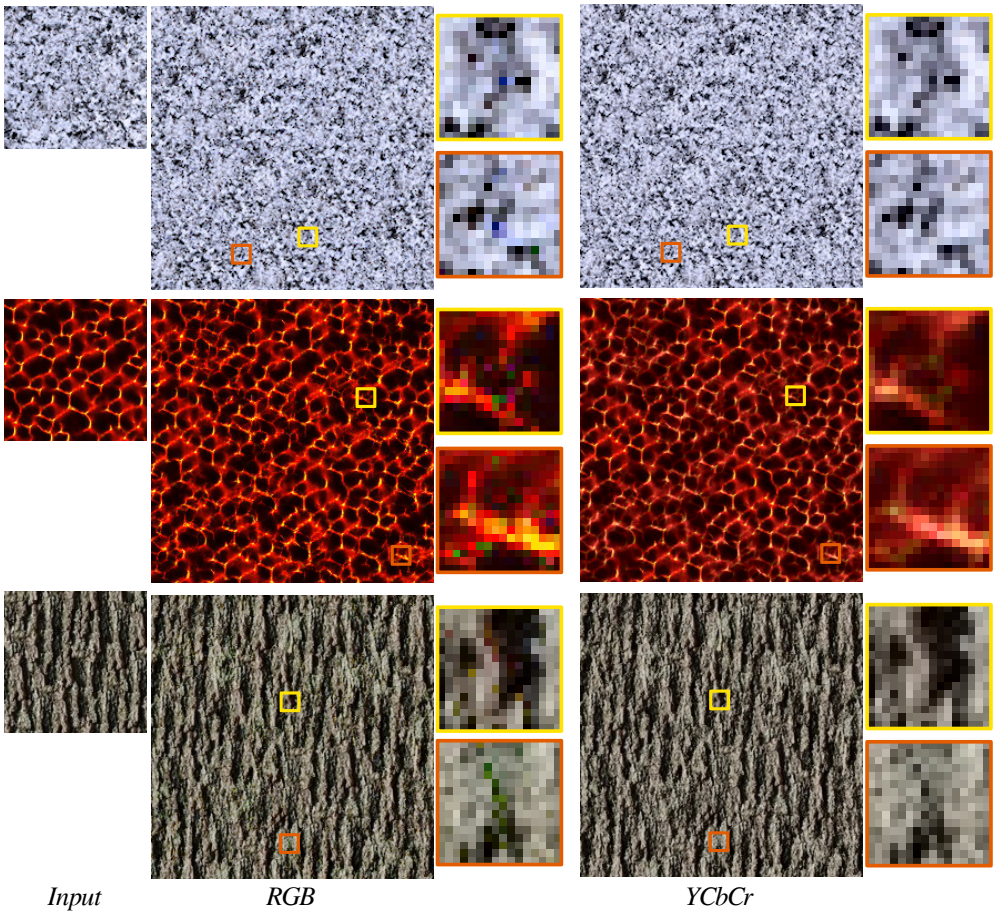


Figure 11. Examples of coloration artifacts that are avoided by blending in YCbCr color space and performing histogram-preserving blending only on the Y channel. [The input textures are from [2018] and Heitz and Neyret [2018, suppl.]]

does not improve the artifact in Figure 4(c), and the contrast operator requires more computation than our soft-clipping operator. Thus we have not found it advantageous in practice.

Color preservation. We have found that most textures do not exhibit coloration artifacts with per-channel (1D) histogram preservation, and for examples we have found, as in Figure 11, we have shown that blending in YCbCr color space and performing histogram-preserving blending on just the Y channel avoids discoloration. Many additional results comparing per-channel RGB with YCbCr luminance-only histogram preservation are shown in Figure 12; in most cases the results are indistinguishable.

Additionally, the jaguar texture in Figure 1 exhibits significant discoloration if blended as RGB. The input image in this case contains substantial color noise in the dark pixels which is presumably introduced by the digital camera sensor and not present in the subject. In that figure, YCbCr blending with histogram-preservation only on Y is used to avoid discoloration.

For our YCbCr formulation, we choose the one from the JPEG image file format [International Telecommunication Union 2011] simply because of its widespread use, but other formulations should work. There are also other potential choices for luminance-separated color spaces. However, hue-separated spaces such as HSV and HSL are problematic for linear blending given that hue values wrap around from 1 to 0 (e.g., blending a near-red of 0.99 with a near-red of 0.01 results in a cyan hue rather than the expected pure red). Color spaces with nonlinear transformations were not considered.

Another color preservation option was recently demonstrated in the concurrent work of Deliot and Heitz [2019] where the authors performed histogram-preserving blending in a colorspace defined by the eigenvectors of the RGB covariance matrix such that the color channels are naturally decorrelated. This may provide superior results in that it would not suffer color contrast loss, but at the modest expense of increased preprocessing time. We wouldn't however expect more than a marginal quality improvement as we have not found significant color contrast loss or residual coloration artifacts with our approach. The main advantage might be in robustness as their approach should never perform adversely, for instance with non-color data such as a normal map. A detailed comparison is left for future work.

Another open question is whether textures exist that can be tiled and blended and yet have such strong color correlation, or such an unusually shaped 3D histogram, that neither our YCbCr blending nor Deliot and Heitz's eigenvector blending is satisfactory. These may require 3D histogram preservation as in [Heitz and Neyret 2018], to remap blended colors to colors present in the original image, but even this may not eliminate all unwanted colors. For instance, blending a texture with red flowers in a field of green will likely result in some flowers turning green, which is unnatural.

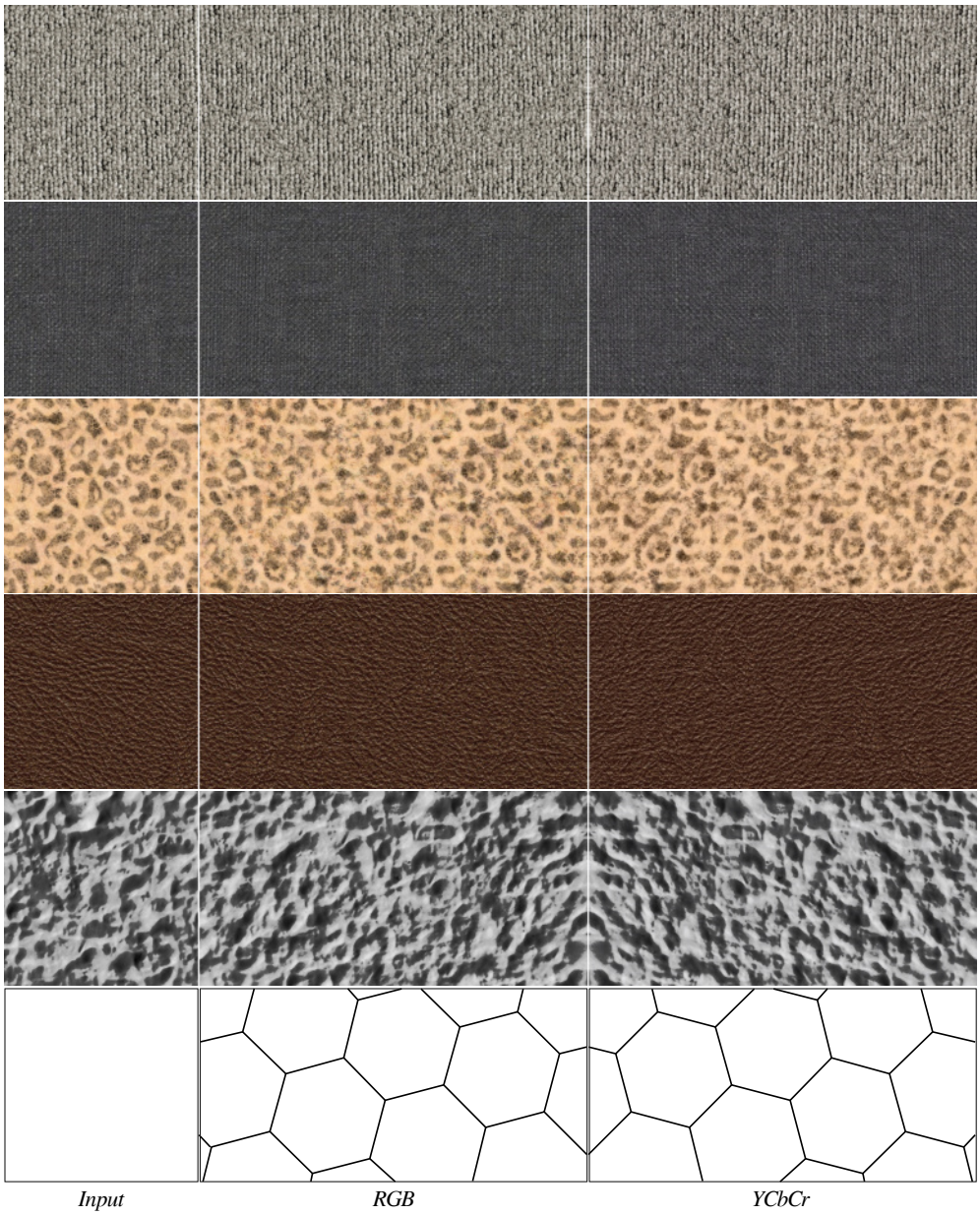
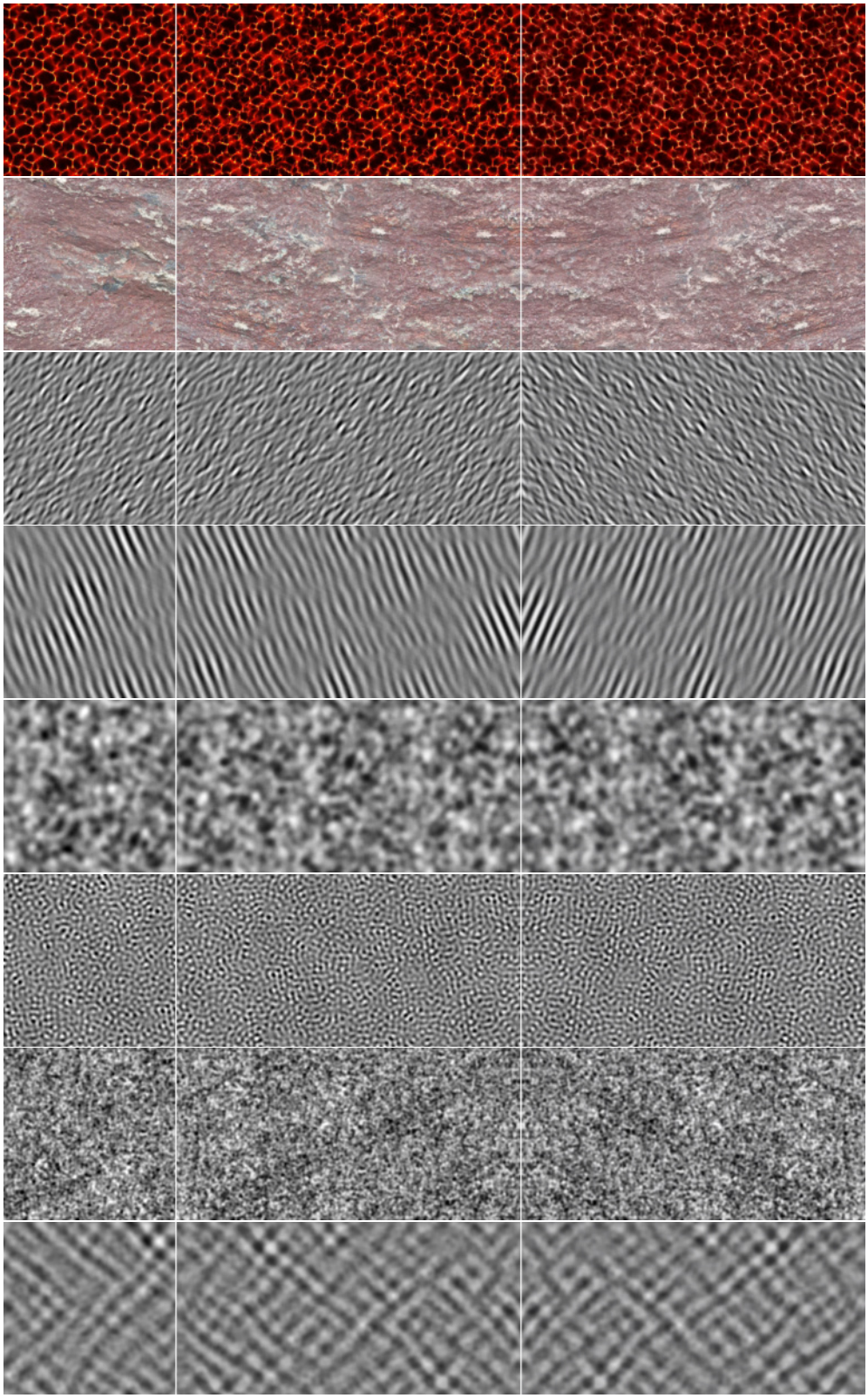
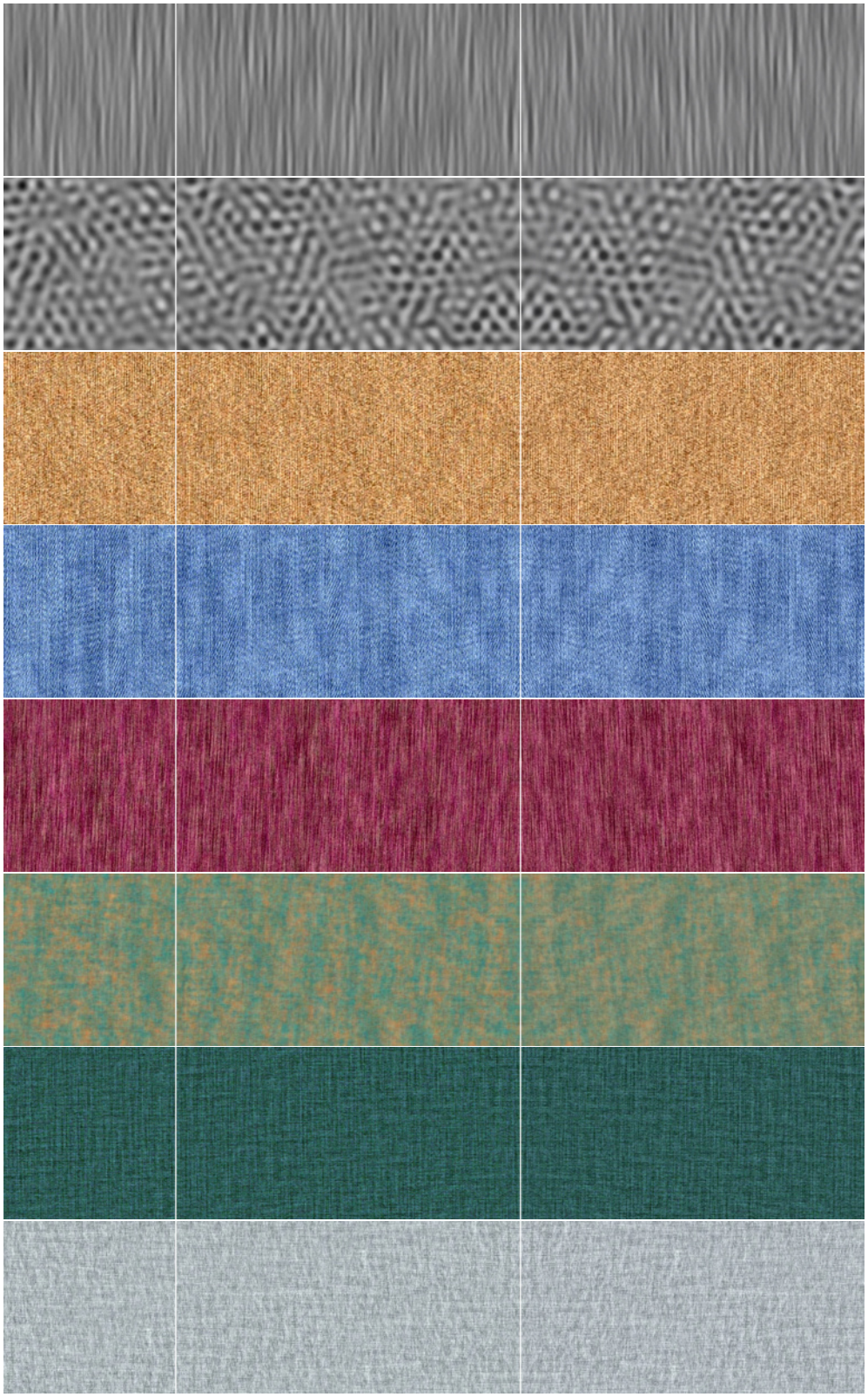
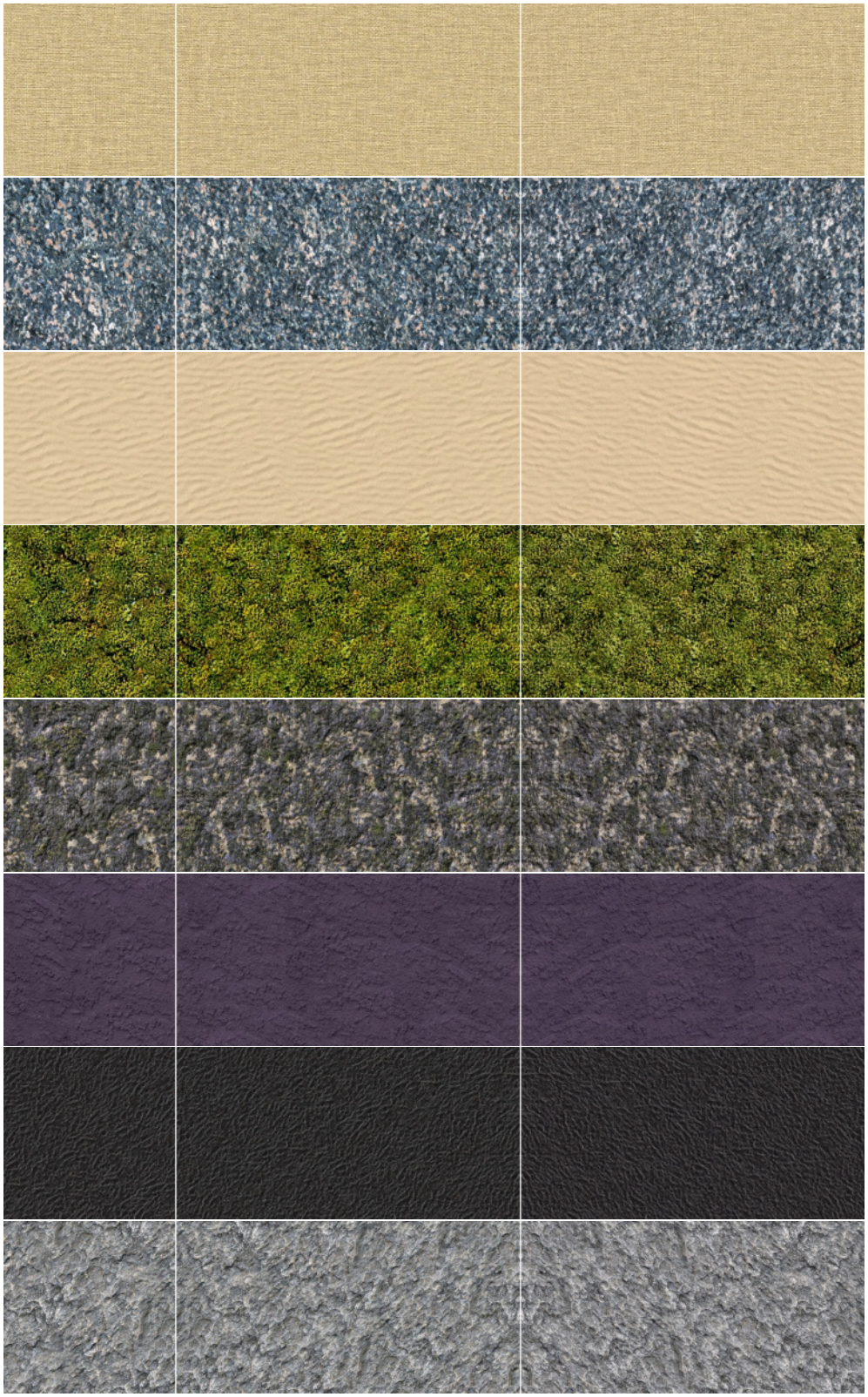
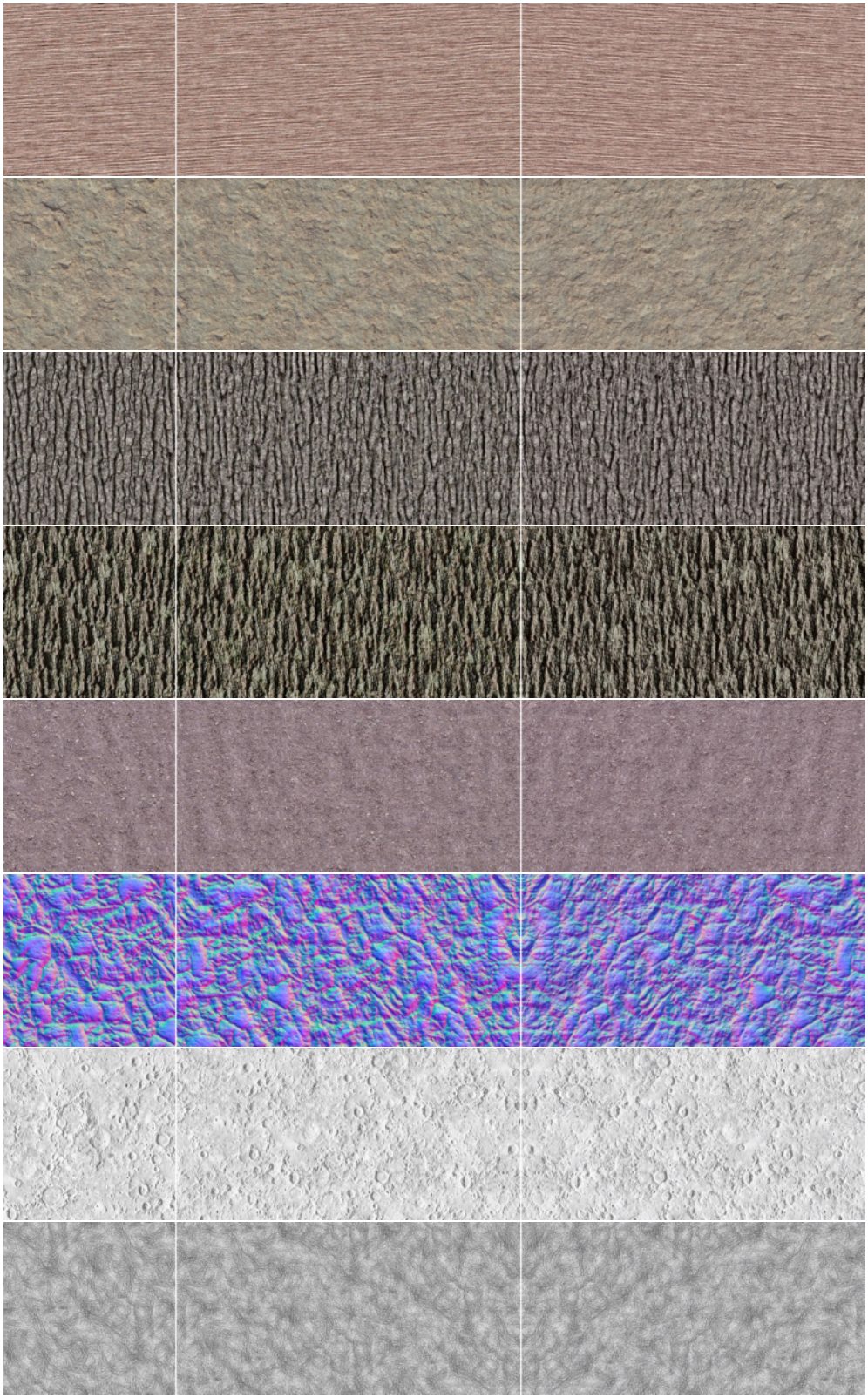


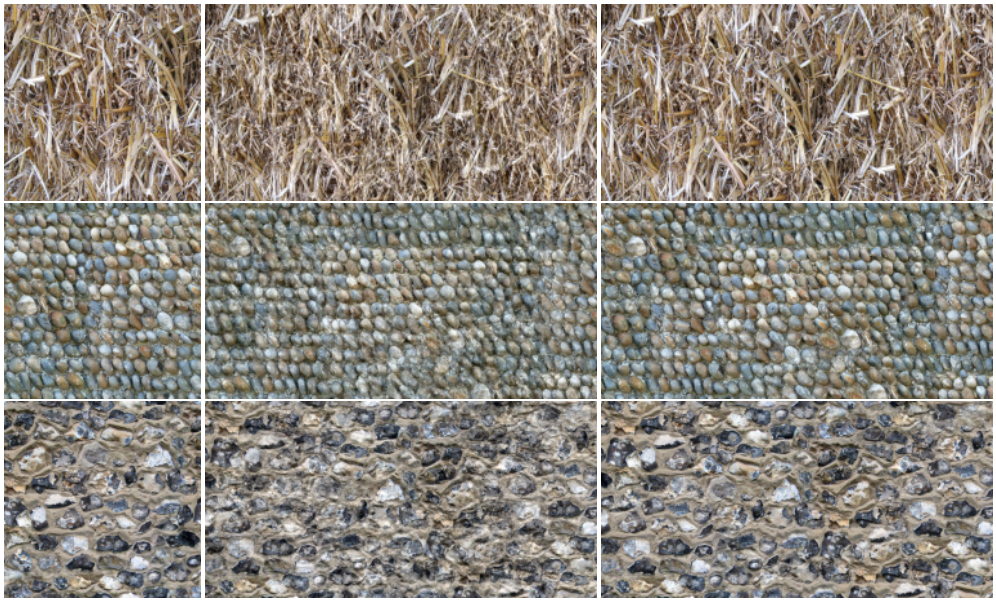
Figure 12. Randomized tiling with histogram-preserving blending using our truncated Gaussian formulation with soft-clipping. In each group, the left image is the input, the middle image uses RGB blending, and the right image (flipped for easier comparison) uses YCbCr blending with histogram-preservation only on Y. Results continue on the following pages. [Input textures from Heitz and Neyret [2018] and [2018, suppl.]]











(a) Input (b) $\gamma = 1$ (c) $\gamma = 8$

Figure 13. Input textures with strong structure shown here benefit greatly from exponentiated blending. [The input textures are from maxTextures.com, courtesy of Max Boughen.]

Exponentiated blending. Exponentiated blending is an optional feature that is independent of the blending method used (and of whether using histogram-preserving blending or not). Results benefiting from exponentiated blending are shown in Figure 13. The exponent, γ, can be any positive real value, though integer exponents typically require less computation. If floating-point exponents are used, then $\beta = \frac{1}{\gamma}$ with a range of 0 to 1 may provide a more user-friendly parameterization.

We have found that for most textures a value of $\gamma = 4$ works well, and hard-coding this value reduces each exponentiation to two multiplies. But if it can be

Stage	Time (ms)	
Texture preprocessing	< 1	
Randomized tiling	38	
Exponentiated blending	< 1	
Contrast restoration, $S^*_{[\widehat{G}]}$	5	(per channel)
(S_G, for comparison)	3	(per channel)
Inverse Gaussianization (LUT)	7	(per channel)
YCbCr to RGB (if used)	5	
Total (RGB)	75	
Total (YCbCr)	56	

Table 1. Timing breakdown for generating a 4Kx4K image from a 256x256 input texture (using 24 core Xeon @ 3.0GHz).

afforded, this is likely a value that is worth tuning for the specific texture. Given that we assume $\sigma = \frac{1}{6}$ is fixed, γ remains our only free parameter.

Timing. Timings are shown in Table 1. Using YCbCr adds overhead to transform back to RGB, but offers a net gain as only the Y channel needs contrast restoration and inverse Gaussianization.

8. Conclusion

We have shown that performing histogram-preserving blending per-channel requires trivial precomputation and provides high-quality results for most textures, and that using a truncated Gaussian formulation with our soft-clipping contrast operator avoids clipping artifacts while permitting the use of a fixed-point texture format and a LUT for inverse Gaussianization. Together these features support smooth interactive editing of large tileable textures. Coloration artifacts may occur with per-channel blending for some textures and we have shown that luminance-only blending offers a potential solution, and a modest performance advantage over per-channel RGB blending as a fringe benefit. Additionally, we have demonstrated that using exponentiated blending weights can reduce ghosting artifacts and improve the result for some textures.

Appendix - Uniform Distribution

The uniform distribution is

$$\mathcal{H}_U(x) = \Pi\left(x - \frac{1}{2}\right) = \begin{cases} 1 & 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$$

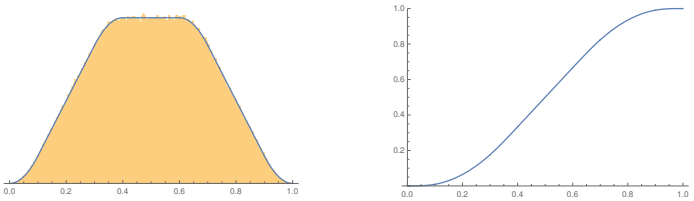
$$\mathrm{CDF}_U(x) = \mathrm{CDF}_U^{-1}(x) = \begin{cases} x & 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$$

The convolution of uniform distributions is

$$\mathcal{H}_{\widehat{U}}(\hat{x}) = \frac{\Pi(\frac{x}{w_1} - \frac{1}{2})}{w_1} * \cdots * \frac{\Pi(\frac{x}{w_N} - \frac{1}{2})}{w_N}$$

which is a piecewise polynomial with degree $N - 1$. The CDF of the blended distribution, $\mathrm{CDF}_{\widehat{U}}$, is the integral of this convolution and is therefore a piecewise polynomial with degree N. The contrast operator is equal to this CDF (noting that CDF_U^{-1} is identity):

$$\begin{aligned} S_{\widehat{U}}(\hat{x}) &= \mathrm{CDF}_U^{-1}\left(\mathrm{CDF}_{\widehat{U}}(\hat{x})\right) \\ &= \mathrm{CDF}_{\widehat{U}}(\hat{x}) \end{aligned}$$



$\mathcal{H}_{\widehat{U}}(\hat{x})$ $S_{\widehat{U}}(\hat{x})$

Figure 14. Linear blend of three sets of 2^{20} uniformly distributed samples with weights 0.1, 0.3, and 0.6, shown with predicted distribution (left) and corresponding contrast operator (right).

The contrast operator for a blend of three uniformly distributed values is

$$S_{\widehat{U}}\left(\hat{x}|\hat{x} \le \frac{1}{2}; a, b, c\right) = \begin{cases} \frac{\hat{x}^3}{6abc} & 0 \le \hat{x} < a \\ \frac{a^2-3a\hat{x}+3\hat{x}^2}{6bc} & a \le \hat{x} < b \\ \frac{a^3-3a^2\hat{x}+3a\hat{x}^2+(b-\hat{x})^3}{6abc} & b \le \hat{x} < \min(a+b, c) \\ \frac{2\hat{x}-a-b}{2c} & a+b \le \hat{x} < \frac{1}{2} \\ \frac{a^3+b^3+c^3-3\hat{x}\left(a^2+b^2+c^2\right)+3\hat{x}^2(a+b+c)-2\hat{x}^3}{6abc} & c \le \hat{x} < \frac{1}{2} \end{cases}$$

$$S_{\widehat{U}}\left(\hat{x}|\hat{x} > \frac{1}{2}; a, b, c\right) = 1 - S_{\widehat{U}}(1-\hat{x}; a, b, c)$$

where a, b, and c are the blending weights, $a \le b \le c$, and $a + b + c = 1$.

$\mathcal{H}_{\widehat{U}}(\hat{x})$ and $S_{\widehat{U}}(\hat{x})$ are illustrated in Figure 14.

Acknowledgements

I am indebted to the anonymous reviewers for their invaluable suggestions. I am also grateful to Eric Heitz for allowing the use of example textures from [Heitz and Neyret 2018] and to Max Boughen for allowing the use of the textures in Figure 13.

References

BERGEN, J., AND HEEGER, D. 1995. Pyramid-based texture analysis/synthesis. In *International Conference on Image Processing*, vol. 3. IEEE Computer Society, Los Alamitos, CA, USA, Oct, 3648. URL: https://doi.ieeecomputersociety.org/10.1109/ICIP.1995.537718. 34

DELIOT, T., AND HEITZ, E. 2019. Procedural stochastic textures by tiling and blending. In *GPU Zen 2*, W. Engel, Ed. Black Cat Publishing Inc., Encinitas, CA. 44

HEITZ, E., AND NEYRET, F. 2018. High-Performance By-Example Noise using a Histogram-Preserving Blending Operator. *Proceedings of the ACM on Computer Graphics and Interactive Techniques (issue for the ACM SIGGRAPH / Eurographics Symposium on*

High-Performance Graphics 2018) 1, 2 (Aug.), 25. URL: https://hal.inria.fr/hal-01824773, doi:10.1145/3233304. 32, 34, 36, 39, 42, 43, 44, 45, 52

INTERNATIONAL TELECOMMUNICATION UNION, 2011. Rec. T.871 – Digital compression and coding of continuous-tone still images: JPEG File Interchange Format (JFIF). URL: https://www.itu.int/rec/T-REC-T.871/en. 44

Author Contact Information

Brent Burley
Walt Disney Animation Studios
500 S. Buena Vista St.
Burbank, CA 91521
brent.burley@disneyanimation.com

Brent Burley, On Histogram-preserving Blending for Randomized Texture Tiling, *Journal of Computer Graphics Techniques (JCGT)*, vol. 8, no. 4, 31–53, 2019
http://jcgt.org/published/0008/04/02/

Received:	2018-10-16		
Recommended:	2019-02-06	Corresponding Editor:	Eric Enderton
Published:	2019-11-08	Editor-in-Chief:	Marc Olano