The LogLuv Encoding for Full Gamut, High Dynamic Range Images

Gregory Ward Larson
Silicon Graphics, Inc.
Mountain View, California

Abstract

The human eye can accommodate luminance in a single view over a range of about 10,000:1 and is capable of distinguishing about 10,000 colors at a given brightness. By comparison, typical computer monitors have a luminance range less than 100:1 and cover about half of the visible color gamut. Despite this difference, most digital image formats are geared to the capabilities of conventional displays, rather than the characteristics of human vision. In this paper, we propose a compact encoding suitable for the transfer, manipulation, and storage of high dynamic range color images. This format is a replacement for conventional RGB images, and encodes color pixels as log luminance values and CIE (u',v') chromaticity coordinates. We have implemented and distributed this encoding as part of the standard TIFF I/O library available by anonymous ftp. After explaining our encoding, we describe its use within TIFF and present some techniques for handling high dynamic range pixels, and demonstrate with an example image.

1. Introduction

Recently, there has been increased interest in high dynamic range (HDR) images, both captured and synthetic [Debevec 97] [Ward 94], which permit extended processing and higher fidelity display methods [Debevec 98] [Larson 97] [Pattanaik 98]. Conventional 24-bit RGB formats cannot encode this additional information, and simple floating point extensions require too much storage space. Some formats, used in the digital film industry, extend the dynamic range slightly using a logarithmic RGB space. Pixar has been using a 33-bit/pixel log format for years, which covers 3.5 orders of magnitude with 0.4% accuracy. Cineon has a 30-bit/pixel log format, but it only covers about 2 orders of magnitude, depending on the type of film being scanned. Another solution, employed in the *Radiance* rendering system, is to append 8 bits to each pixel to represent a common exponent for three 8-bit RGB mantissas [Ward 91]. This provides over 77 orders of magnitude in dynamic range, and works well for the majority of images, but can produce visible quantization artifacts and gamut clamping for some highly saturated colors. This is due to the imperfect separation of luminance and chrominance, and the limited gamut of a non-negative RGB color space.

In this paper, we present a new pixel encoding that uses a log representation of luminance and a CIE (u',v') representation of chrominance.[*] We call this a *LogLuv* encoding. This encoding has the following desirable properties. It:

- Covers the entire visible color gamut.
- Covers the full range of perceivable luminances (over 38 orders of magnitude).
- Uses imperceptible step sizes in a perceptually uniform space.
- May be calibrated to absolute luminance and color.
- Enables optimal visual fidelity on any output device.

In addition to these inherent features, we have the following technical goals for our format. We would like it to be:

- Compact.
- Compressible.
- Easy to convert to XYZ and RGB formats.
- Incorporated into the TIFF standard.

In this paper, we describe our LogLuv pixel encoding method, followed by a description of our extension to Sam Leffler's free TIFF library. We then discuss some practical considerations, and give an example application to HDR tone mapping, ending with a brief conclusion.

2. Encoding Method

We have actually implemented two LogLuv pixel encodings, a 24-bit encoding and a 32-bit encoding. The 24-bit encoding breaks down into a 10-bit log luminance portion and a 14-bit, indexed uv coordinate mapping. The 32-bit

[*]Chrominance is the quantity equivalent to hue plus saturation. We follow conventional CIE usage by using "color" and "chrominance" interchangeably, though the latter is properly a subset of the former [Wyszecki82].

encoding uses 16 bits for luminance and 8 bits each for u' and v'. Compared to the 24-bit encoding, the 32-bit version provides greater dynamic range and precision at the cost of an extra byte per pixel. Also, the 32-bit format is simpler and compresses better, such that most 32-bit LogLuv images end up smaller than their 24-bit counterparts. Therefore, we will not discuss the 24-bit encoding in this article, but refer the reader to the original paper for details [Larson 98].

2.1 32-bit LogLuv Pixel Encoding

The 32-bit LogLuv encoding uses 16 bits for luminance information and 16 bits for chrominance. The MSB is used to flag negative luminances, and the next 15 bits record up to 38 orders of magnitude in 0.27% relative steps, covering the full range of perceivable world luminances in imperceptible steps. The lower two bytes encode u' and v', respectively. The bit breakdown is shown in Fig. 1.

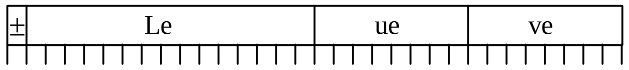

Figure 1. Bit allocation for 32-bit pixel encoding. MSB is a sign bit, and the next 15 bits are used for a log luminance encoding. The uv coordinates are separate 8-bit quantities.

The conversion to and from our log luminance encoding is given in Eq. 1. The maximum luminance using this encoding is 1.84×10^{19}, and the smallest magnitude is 5.44×10^{-20}. An L_e value of 0 is taken to be exactly 0.0. The sign bit is extracted before encoding and reapplied after the conversion back to real luminance.

$$L_e = \lfloor 256(\log_2 Y + 64) \rfloor \qquad (1a)$$

$$Y = \exp_2\left[\left(L_e + 0.5\right)/256 - 64\right] \qquad (1b)$$

Since the gamut of perceivable *u* and *v* values is between 0 and 0.62, we chose a scale factor of 410 to go between our [0,255] integer range and real coordinates, as given in Eq. 2.

$$u_e = \lfloor 410u' \rfloor \qquad (2a)$$

$$v_e = \lfloor 410v' \rfloor \qquad (2b)$$

$$u' = \left(u_e + 0.5\right)/410 \qquad (2c)$$

$$v' = \left(v_e + 0.5\right)/410 \qquad (2d)$$

This encoding captures the full color gamut in 8 bits each for u_e and v_e. There will be some unused codes outside the visible gamut, but the tolerance this gives us of 0.0017 units in uv space is already well below the visible threshold. Conversions to and from 1931 CIE (x,y) chromaticities are given in Eqs. 3 and 4.

$$u' = \frac{4x}{-2x + 12y + 3} \qquad (3a)$$

$$v' = \frac{9y}{-2x + 12y + 3} \qquad (3b)$$

$$x = \frac{9u'}{6u' - 16v' + 12} \qquad (4a)$$

$$y = \frac{4v'}{6u' - 16v' + 12} \qquad (4b)$$

where:

$x = X/(X+Y+Z)$

$y = Y/(X+Y+Z)$

3. TIFF Input/Output Library

The LogLuv encoding described has been embedded as a new `SGILOG` compression type in Sam Leffler's popular TIFF I/O library. This library is freely distributed by anonymous ftp at the site given at the end of this article.

When writing a high dynamic range TIFF image, the LogLuv *codec* (compression/decompresson module) takes floating point CIE XYZ scanlines and writes out 24-bit or 32-bit compressed LogLuv-encoded values. When reading an HDR TIFF, the reverse conversion is performed to get back floating point XYZ values. (We also provide a simple conversion to 24-bit gamma-compressed RGB for the convenience of readers that do not know how to handle HDR pixels.)

An additional tag is provided for absolute luminance calibration, named `TIFFTAG_STONITS`.* This is a single floating point value that may be used to convert Y values returned by the reader to absolute luminance in candelas per square meter. This tag is set by the application that writes out a HDR TIFF to permit prescaling of values to a reasonable brightness range for display, where values of 1.0 will be displayed at the maximum output of the destination device. This avoids the image reader having to figure out a good exposure level for absolute luminances. If the input data is uncalibrated (i.e., the absolute luminances are unknown), then there is no need to store this tag, whether the values are scaled or not.

3.1 Run-length Compression

By separating the bytes into four streams on each scanline, the 32-bit encoding can be efficiently compressed using an adaptive run-length encoding scheme [Glassner 91]. Since the top byte containing the sign bit and upper 7 log luminance bits changes very slowly, this byte-stream submits very well to run-length encoding. Likewise, the encoded u_e and v_e byte-streams compress well over areas of constant chrominance.

* STONITS stands for "samples to Nits," where "Nits" is the photometric unit for luminance, also written candelas/meter^2. Use of this tag will be discussed later in section 5.2.

3.2 Grayscale Images

For maximum flexibility, a pure luminance mode is also provided by the codec, which stores and retrieves run-length encoded 16-bit log luminance values using the same scheme as applied in the 32-bit LogLuv encoding. There is no real space savings over a straight 32-bit encoding, since the u_e and v_e byte-streams compress to practically nothing for grayscale data, but this option provides an explicit way to specify floating point luminance images for TIFF readers that care.

3.3 Raw I/O

It is also possible to decode the raw 32-bit LogLuv data retrieved from an HDR TIFF directly, and this has some advantages for implementing fast tone mapping and display algorithms. For the 32-bit format, one can simply multiply the output of a 32 Kentry L_e table and a 64 Kentry uv table to get a tone-mapped and gamma-compressed RGB result, provided the tone mapping algorithm can separate chrominance from luminance. A explanation of how this is done is given in Section 5.1.

3.4 Example TIFF Code and Images

Use of the LogLuv encoding is demonstrated and sample images are provided on our web site, which is given at the end of this article. A converter has been written to and from the *Radiance* floating point picture format [Ward 94] [Ward 91], and serves as an example of LogLuv codec usage. The web site itself also offers programming tips and example code segments.

Example TIFF images using the 32-bit LogLuv and 16-bit LogL encoding are provided on the web site. These images are either scanned from photographic negatives or rendered using *Radiance* and converted to the new TIFF format. Some images are rendered as 360° QuickTime VR panoramas suitable for experiments in HDR virtual reality.

4. Practical Considerations

There are several important considerations in applying the LogLuv image format, which we describe in this section. Issues such as conversion to and from RGB space require thought about dynamic range and gamut limitations, and calibrated versus uncalibrated color spaces. We also discuss speed and space efficiency issues, and appropriate filtering and image manipulation techniques.

4.1 Converting from RGB to XYZ

Ideally, the original image data is available in a real, XYZ color space or a spectrally sampled color space that can be converted to XYZ using standard CIE techniques [Wyszecki 82]. However, most imaging devices and renderers are based on some RGB standard. To get from RGB into XYZ space, a simple 3×3 matrix transformation may be computed from the CIE (x,y) chromaticities for the three RGB primaries and a white point [Rogers 85]. Using the standard CCIR 709 RGB primaries for computer displays and a neutral white point for optimal color balance, we derive the color transformation shown in Eq. 5.

	R	G	B
x	0.640	0.300	0.150
y	0.330	0.600	0.060

Table 1. CIE (x,y) chromaticities for CCIR 709 RGB primaries.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.497 & 0.339 & 0.164 \\ 0.256 & 0.678 & 0.066 \\ 0.023 & 0.113 & 0.864 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (5)$$

4.2 Luminance Calibration

If the image corresponds to a luminous scene (as opposed to a painting or other reflective media), it may be possible to calibrate the recorded information using the `TIFFTAG_STONITS` field mentioned earlier. This is a real multiplier that is stored with the image by whoever creates it. When reading an image, an application can retrieve this value and multiply it by each pixel's Y value to get an absolute luminance in cd/m^2. If the absolute luminance for a given pixel or subimage is known by the TIFF writer, this multiplier is also known, since it equals the absolute luminance divided by the output Y value. If, on the other hand, we are working from a scan of a photographic negative or transparency, it may be possible to approximate this multiplier from the image exposure time, f-stop and film speed. For 35mm photography, we can use the formula given in Eq. 6, borrowed from the IES Handbook [IES 93].

$$m = \frac{200}{\pi S} \cdot f^2 / t \qquad (6)$$

where:

S	= film speed (ISO ASA)
f	= aperture (f-stop)
t	= exposure time (seconds)

Eq. 6 assumes that the maximum image brightness corresponds to a Y value of 1.0. For photographic negatives, which hold about 2 orders of magnitude of extended dynamic range beyond white, we assume this "maximum" corresponds to a density of about 1.5 (3% negative transmittance).

4.3 Converting back to RGB

For the reverse conversion, we invert the matrix from Eq. 5 as shown in Eq. 7.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 2.690 & -1.276 & -0.414 \\ -1.022 & 1.978 & 0.044 \\ 0.061 & -0.224 & 1.163 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad (7)$$

Since some of the matrix coefficients are negative due to the larger gamut of the imaginary CIE primaries, the formula given in Eq. 7 may result in negative RGB values for some highly saturated colors. If the image processing software can cope with negative values, it is better to leave them that way, otherwise a gamut mapping operation may be performed to bring them back into the legal range. Much research has been devoted to this problem [Stone 88], but clamping is the most often applied solution. We have also found desaturating to a face of the RGB cube to be a simple and effective solution to limiting the color gamut.

If floating point RGB colors are maintained without gamut limiting (i.e., negative values allowed), it is possible to go from LogLuv→RGB→LogLuv without losing data. However, the opposite is not true, since the LogLuv encoding quantizes data into perceptual bins. Even in the case of integer RGB data, there will be some differences due to the binning used by the two encodings.

4.4 RGB→LogLuv→RGB Information Loss

Even though the gamut and dynamic range of the 32-bit LogLuv format is superior to that of 24-bit RGB, we will not get the exact pixel values again if we go through this format and back to RGB. This is because the quantization size of a LogLuv pixel is matched to human perception, whereas 24-bit RGB is not. In places where human error tolerance is greater, the LogLuv encoding will have larger quantization volumes than RGB and therefore may not reproduce exactly the same 24-bit values when going back and forth. Over most of the gamut, the LogLuv tolerances will be tighter, and RGB will represent lower resolution information. (This is especially true for dark areas.) In other words, the losses incurred going through the LogLuv format may be measurable in absolute terms, but they should not be visible to a human observer since they are below the threshold of perception.

We performed two tests to study the effects of going between RGB and LogLuv formats, one quantitative test and one qualitative test. In the quantitative test, we went through all 16.7 million 24-bit RGB colors and converted to 32-bit LogLuv and back, then measured the difference between the input and output RGB colors using the CIE E* perceptual error metric. We found that 17% of the colors were translated exactly, 80% were below the detectable threshold and 99.75% were less than twice the threshold, where differences may become noticeable. In our qualitative test, we examined a dozen or so images from different sources, performing the RGB→LogLuv→RGB mapping in interleaved 16 scan-line stripes that roughly corresponded to the maximum visible angular frequency. In this way, we hoped to notice the underlying pattern in cases where the translation resulted in visible differences. In all of the captured images we looked at, the details present completely obscured any differences caused by the translation, even in sky regions that were relatively smooth. Only in one synthesized image, which we designed very carefully to have smooth gradients for our tests, were we just barely able to discern the pattern in large, low detail regions. Even so, we had to really be looking for it to see it, and it sort of faded in and out, as if the eye could not decide if it was really there or not.

From these tests, we concluded that the differences caused by taking RGB data through the LogLuv format will not be visible in side-by-side comparisons.

4.5 LogLuv Image Compression

The simple adaptive run-length encoding scheme used to compress our 32-bit/pixel LogLuv format performs about as well on average as the LZW scheme used on standard TIFF RGB images. Since luminance and chrominance are placed in separate byte streams, regions where one or the other are relatively constant compress very well. This is often the case in computer graphics imagery, which will compress between 15% and 60% for reasonably complex scenes, and will often outperform LZW compression so much that the resulting file size is actually smaller than the LZW-compressed 24-bit RGB version. For scanned imagery, the performance is usually not as good using run-length encoding due to the increased complexity and image noise. However, we never grow the data, as can happen with the LZW compression algorithm, which is sometimes worse than no compression at all.

For example, compression performance for the scene shown in Fig. 2 is -8% for LZW compression, compared to 13% for our run-length encoding. The LZW file still ends up slightly smaller since it is starting from 24-bit pixels, but our compressed result has an additional advantage, which is lower entropy. Specifically, we can apply an entropy encoding scheme to our run-length encoded result, and pick up some additional savings. For the same figure, applying the gzip program to the result gains an additional 17% compression, making the resulting file smaller than the straight LZW encoding. (Applying gzip to the LZW file only reduces its size by 4%, which is not enough to make up the difference.) In most cases, applying gzip to our run-length encoded 32-bit LogLuv TIFF yields a file that is within a 10% of a gzip'ped 24-bit RGB file, which is almost always smaller than TIFF's blockwise LZW compression.

4.6 Image Processing

There are two basic approaches for processing LogLuv encoded images. One method is to convert to a floating-point RGB or XYZ color space and perform image manipulations on these quantities. This is probably the simplest and most convenient for general purposes. The other method is to work directly in a separate luminance space such as Yuv or Yxy. For certain types of image processing, such as contrast and brightness adjustment, it may actually be faster since we can work on the Y channel alone, plus we save a little time on our conversions.

In general, it is not a good idea to convert to an 8-bit/primary integer space to manipulate high dynamic range images, because the dynamic range is lost in the process. However, since most existing hardware and software is geared to work with 24-bit RGB data, it may be easiest to use this format for interactive feedback, then apply the manipulation sequence to the floating point data when writing the file. Better still, one could use the approach adopted by some software packages and store the sequence of image processing commands together with the original data. These commands may be recorded as additional TIFF tags and attached to the file without even touching the data. Besides saving time, this approach also preserves our absolute luminance calibration, if `TIFFTAG_STONITS` is present.

Compositing high dynamic range images using an alpha channel is also possible, though one must consider the effect of multiplying a floating point value by an integer value. When one is small and the other is large, the result may either be wrong or show severe quantization artifacts. Therefore, it is best to use a high dynamic range alpha channel, which may be stored as a separate 16-bit LogL TIFF layer.

An additional benefit of the 32-bit LogLuv and 16-bit LogL formats is that pixels may take on negative values, which are useful for arbitrary image math and general masking and filtering. For example, an image may be broken into separate 2D Fourier frequency layers, which when summed together yield back the original image. The coefficients for these layers may then be tuned independently or manipulated in arbitrary ways. This is usually done in memory, such that the written file format pays little part, but being able to write out the intermediate pixel data without losing information has potential advantages for subsequent processing.

5. Motivating Application: Tone Mapping

Tone mapping is the process of mapping captured image values to displayed or printed colors [Tumblin93]. With the added dynamic range and gamut provided by our new encoding, we can get a much better mapping than we could with the limited range of conventional digital images. If our data is also calibrated (i.e., `STONITS` has been set by the creating application), then we can go further to simulate visibility using what we know about human color and contrast sensitivity. Recently, a number of tone mapping techniques have been developed for images with high dynamic range [Pattanaik 98] [Spencer 95] [Chiu 93] [Jobson 97]. Most of these methods introduce image-geometric dependencies, so they cannot be applied independently to each pixel, or computed using integer math. In this section, we present an efficient tone-mapping technique for HDR images that can be applied on a pixel-by-pixel basis using integer operations [Larson97].

5.1 Efficient Tone Mapping

As we mentioned earlier in Section 3.3, we can go directly from the luminance and uv image data to an RGB tone-mapped result by multiplying the output of two lookup tables. To get the math to work out right, both the luminance and uv table values must be in the monitor's gamma-compressed space, and the RGB values must be premultiplied by their corresponding luminance coefficients. The computation for luminance and RGB table entries corresponding to specific tone-mapped display colors is given in Eq. 8.

$$L_t(L_e) = \left\lfloor 256\, L_d(L_e)^{1/g} \right\rfloor \qquad \textit{(8a)}$$

$$R_t(C_e) = \left\lfloor 256\left(S_r R_1(C_e)\right)^{1/g} \right\rfloor \qquad \textit{(8b)}$$

$$G_t(C_e) = \left\lfloor 256\left(S_g G_1(C_e)\right)^{1/g} \right\rfloor \qquad \textit{(8c)}$$

$$B_t(C_e) = \left\lfloor 256\left(S_b B_1(C_e)\right)^{1/g} \right\rfloor \qquad \textit{(8d)}$$

where:

L_t, R_t, G_t, B_t	= lookup table values
L_e	= encoded log luminance
C_e	= encoded uv color
$L_d(L_e)$	= mapped display luminance
R_1, G_1, B_1	= normalized color mapping
γ	= monitor response gamma
S_r, S_g, S_b	= red, green, blue coefficients

from middle row of Eq. 5 matrix

$S_r \times R_1(C_e) + S_g \times G_1(C_e) + S_b \times B_1(C_e) = 1$

The color coefficients guarantee that all tabulated chrominance values will be between 0 and 255. To get from the separately tabulated luminance and chrominance to display values in the 0-255 range, we apply the formulas given in Eq. 9.

$$R_d = L_t(L_e)\, R_t(C_e) \,/ \left\lfloor 256\, S_r^{\,1/g} \right\rfloor \qquad \textit{(9a)}$$

$$G_d = L_t(L_e)\, G_t(C_e) \,/ \left\lfloor 256\, S_g^{\,1/g} \right\rfloor \qquad \textit{(9b)}$$

$$B_d = L_t(L_e)\, B_t(C_e) \,/ \left\lfloor 256\, S_b^{\,1/g} \right\rfloor \qquad \textit{(9c)}$$

Since the denominators in Eq. 9 are constant, they can be precomputed, leaving only four table lookups, three integer multiplies and three integer divides to map each pixel.

We have implemented this type of integer-math tone-mapping algorithm in an HDR image viewer, and it takes less than a second to convert and display a 512×512 picture on a 180 MHz processor. The L_e table size is determined by the range of luminances present in the image, and only the colors needed are actually translated to RGB and stored in the uv lookup table. We used the high dynamic range operator described in [Larson 97], which requires a modification to our algorithm to include mesopic color correction, since it relates the luminance and color mappings. For pixels below the upper mesopic limit, we determine the color shift in uv coordinates, then do our table lookup on the result.

5.2 Example Results

Fig. 2a shows a scanned photograph as it might appear on a PhotoCD using a YCC encoding. Since YCC can capture up to "200% reflectance," we can apply a tone mapping operator to bring this extra dynamic range into our print, as shown in Fig. 3a. However, since many parts of the image were brighter than this 200% value, we still lose much of the sky and circumsolar region, and even the lighter asphalt in the foreground. In Fig. 2b, we see where 35% of the original pixels are outside the gamut of a YCC encoding.

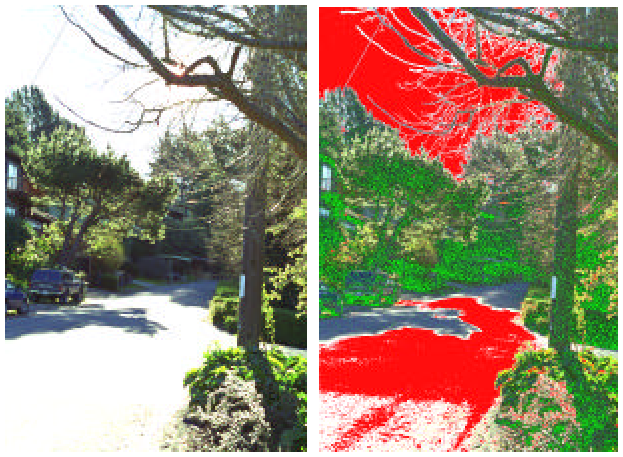

Figure 2. The left image (a) shows a PhotoYCC encoding of a color photograph tone-mapped with a linear operator. The right image (b) shows the out-of-gamut regions. Red areas are too bright or too dim, and green areas have inaccurate color.

Fig. 3b shows the same color negative scanned into our 32-bit/pixel high dynamic range TIFF format and tone mapped using a histogram compression technique [Larson 97]. Fig. 4c shows the same HDR TIFF remapped using the perceptual model of Pattanaik et al [Pattanaik 98]. Figs. 4a and 4b show details of light and dark areas of the HDR image whose exposure has been adjusted to show the detail captured in the original negative. Without an HDR encoding, this information would be lost.

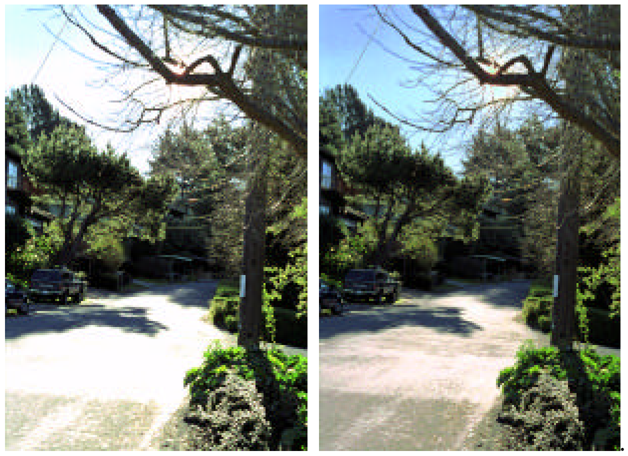

Figure 3. The left image(a) shows the YCC encoding after remapping with a histogram compression tone operator. Unfortunately, since YCC has so little dynamic range, most of the bright areas are lost. The right image (b) shows the same operator applied to a 32-bit HDR TIFF encoding, showing the full dynamic range of the negative.

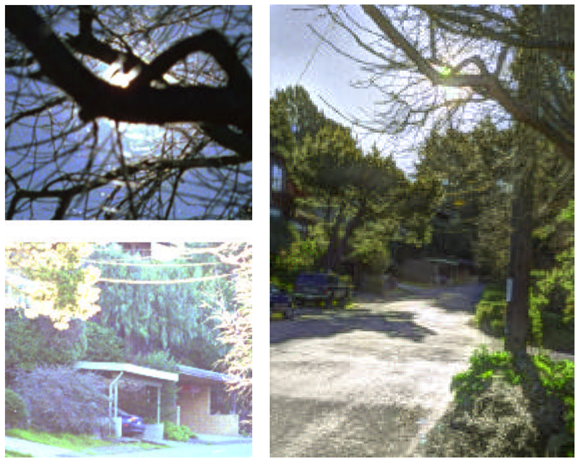

Figure 4. The upper-left image (a) shows the circumsolar region reduced by 4 f-stops to show the image detail recorded on the negative. The lower-left image (b) shows house details boosted by 3 f-stops. The right image (c) shows our HDR TIFF mapped with the Pattanaik-Ferwerda tone operator.

5.3 Discussion and Other Applications

It is clear from our example that current methods for tone-mapping HDR imagery, although better than a simple S-curve, are less than perfect. It would therefore be a mistake to store an image that has been irreversibly tone mapped in this fashion, as some film scanner software attempts to do. Storing an HDR image allows us to take full advantage of future improvements in tone mapping and display algorithms, at a nominal cost.

Besides professional film and photography, there are a number of application areas where HDR images are key. One is lighting simulation, where designers need to see an interior or exterior space as it would really appear, and evaluate things in terms of absolute luminance and illuminance levels. Since an HDR image can store the real luminance in its full-gamut coverage, this information is readily accessible to the designer. Another application is image-based rendering, where a user is allowed to move about in a scene by warping captured or rendered images [Debevec 96]. If these images have limited dynamic range, it is next to impossible to adapt the exposure based on the current view, and impossible to use the image data for local lighting. Using HDR pixels, a natural view can be provided for any portion of the scene, and the images may also be used to illuminate local objects [Debevec 98]. A fourth application area is digital archiving, where we are making a high-quality facsimile of a work of art for posterity. In this case, the pixels we record are precious, so we want to make sure they contain as much information as possible. At the same time, we have concerns about storage space and transmission costs, so keeping this data as compact as possible is important. Since our HDR format requires little more space than a standard 24-bit encoding to capture the full visible gamut, it is a clear winner for archiving applications.

Our essential argument is that we can make better use of the bits in each pixel by adopting a perceptual encoding of color and brightness. Although we don't know how a given image might be used or displayed in the future, we do know something about what a human can observe in a given scene.

By faithfully recording this information, we ensure that our image will take full advantage of any future improvements in imaging technology, and our basic format will continue to find new uses.

6. Conclusion

We have presented a new method for encoding high dynamic range digital images using log luminance and uv chromaticity to capture the entire visible range of color and brightness. The proposed format requires little additional storage per pixel, while providing significant benefits to suppliers, caretakers and consumers of digital imagery.

This format is most appropriate for recording the output of computer graphics rendering systems and images captured from film, scanners and digital photography. Virtually any image meant for human consumption may be better represented with a perceptual encoding. Such encodings are less appropriate when we either wish to store non-visual (or extravisual) information as found in satellite imagery, or output to a specific device with known characteristics such as NTSC or PAL video.

Through the use of re-exposure and dynamic range compression, we have been able to show some of the benefits of HDR imagery. However, it is more difficult to illustrate the benefits of a larger color gamut without carefully comparing hard copy output of various multi-ink printers. Also, since we currently lack the ability to capture highly saturated scenes, our examples would have to be contrived from individual spectral measurements and hypothetical scenes. We therefore leave gamut validation as a future exercise.

Future work on the format itself should focus on the application of lossy compression methods for HDR images and animations. A JPEG-like cosine transform should work very well, since LogLuv looks almost the same perceptually as the gamma-compressed YCbCr coordinates that JPEG uses. Efficient compression is needed for broad acceptance of HDR imagery and extensions for digital cinema.

7. Acknowledgments

The author would like to thank Sam Leffler for his help and cooperation in adding these routines to the TIFF i/o library, Dan Baum for his support and encouragement, Paul Haeberli, Christopher Walker and Sabine Susstrunk for their advice, and Alexander Wilkie, Philippe Bekaert and Jack Tumblin for testing the software. Finally, thanks to Ronen Barzel for his many helpful comments and suggestions on the paper itself.

8. References

[Chiu 93] K. Chiu, M. Herf, P. Shirley, S. Swamy, C. Wang and K. Zimmerman "Spatially nonuniform scaling functions for high contrast images," *Proceedings of Graphics Interface '93*, Toronto, Canada, (May 1993).

[Debevec 98] Paul Debevec, "Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography," *Computer Graphics (Proceedings of ACM Siggraph 98)*.

[Debevec 97] Paul Debevec, Jitendra Malik, "Recovering High Dynamic Range Radiance Maps from Photographs," *Computer Graphics (Proceedings of ACM Siggraph 97)*.

[Debevec 96] Paul Debevec, Camillo Taylor, Jitendra Malik, "Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach," *Computer Graphics (Proceedings of ACM Siggraph 96)*.

[Glassner 91] Andrew Glassner, "Adaptive Run-Length Encoding," in *Graphics Gems II*, edited by James Arvo, Academic Press, (1991).

[IES 93] Illuminating Engineering Society of North America, *IES Lighting Handbook, Reference Volume*, IESNA (1993).

[Jobson 97] Daniel Jobson, Zia-ur Rahman, and Glenn A. Woodell. "Properties and Performance of a Center/Surround Retinex," *IEEE Transactions on Image Processing*, Vol. 6, No. 3 (March 1997).

[Larson 98] Greg Larson, "Overcoming Gamut and Dynamic Range Limitations in Digital Images," Color Imaging Conference, Scottsdale, Arizona (1998).

[Larson 97] Greg Larson, Holly Rushmeier, Christine Piatko, "A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes," *IEEE Transactions on Visualization and Computer Graphics*, **3**, 4, (1997).

[Pattanaik 98] Sumant Pattanaik, James Ferwerda, Mark Fairchild, Don Greenberg, "A Multiscale Model of Adaptation and Spatial Vision for Realistic Image Display," *Computer Graphics (Proceedings of Siggraph 98)*.

[Rogers 85] David Rogers, *Procedural Elements for Computer Graphics*, McGraw-Hill, (1985).

[Spencer 95] G. Spencer, P. Shirley, K. Zimmerman, and D. Greenberg, "Physically-based glare effects for computer generated images," *Computer Graphics* (*Proceedings of Siggraph 95*).

[Stone 88] Maureen Stone, William Cowan, John Beatty, "Color Gamut Mapping and the Printing of Digital Color Images," *ACM Transactions on Graphics*, 7(3):249-292, (October 1988).

[Tumblin93] Tumblin, Jack and Holly Rushmeier. "Tone Reproduction for Realistic Images," *IEEE Computer Graphics and Applications*, November 1993, 13(6).

[Ward 94] Greg Ward, "The RADIANCE Lighting Simulation and Rendering System," *Computer Graphics (Proceedings of Siggraph 94)*.

[Ward 91] Greg Ward, "Real Pixels," in *Graphics Gems II*, edited by James Arvo, Academic Press, (1991).

[Wyszecki 82] Gunter Wyszecki, W.S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae*, Second Edition, Wiley, (1982).

9. Web Information

Links to Sam Leffler's TIFF library, example LogLuv images and other information may be found at: *http://www.acm.org/jgt/papers/Larson99*

Greg may be reached by e-mail at *gregl@sgi.com*.