Segmentation and Adaptive Assimilation for Detail-Preserving Display of High-Dynamic Range Images

Yangli Hector Yee, PDI / DreamWorks, Redwood City, California

Sumanta Pattanaik, University of Central Florida, Orlando, Florida

Descriptive Keywords: Image Processing, Image Segmentation, Signal Processing, Biological Modeling, Tone Reproduction Operator

Abstract

Realistic display of high dynamic range images is a difficult problem. Previous methods for high dynamic range image display suffer from halo artifacts or are computationally expensive. We present a novel method for computing local adaptation luminance that can be used with several different visual adaptation based tone-reproduction operators for displaying visually accurate high dynamic range images. The method uses fast image segmentation, grouping and graph operations to generate local adaptation luminance. Results on several images show excellent dynamic range compression while preserving detail, without the presence of halo artifacts. With adaptive assimilation, the method can be configured to bring out high dynamic range appearance in the display image. The method is efficient in terms of processor and memory use.

1 Introduction

Local adaptation allows the human visual system to perceive detail in an environment with a high dynamic range. For example, an observer indoors can look outside on a sunny day and perceive detail while simultaneously reading a newspaper in the shade. This represents a scene dynamic range of perhaps 1000:1 or more and yet the visual system is able to cope with the large changes in light intensity.

Recently, strides have been made towards capturing high dynamic range digital images. Modern digital cameras are capable of capturing high dynamic range images directly with wide 12-bit analog-to-digital converters. Also, software techniques pionered by Debevec et al. [3] and Nayar et al. [9] allow researchers to reconstruct high dynamic range images from multiple 8-bit images taken at different exposures. Software renderers that perform global illumination are also capable of generating images that have very high dynamic range. A high dynamic range photograph or accurate render of a real world scene can have three orders of magnitude or more of dynamic range. CRT monitors and LCD displays have only one or two orders of magnitude of dynamic range. The simplest solution for displaying a high dynamic range image would be to uniformly scale the high dynamic range image with a constant and then clamp the image to bring the pixel values into the displayable range of the display device. The problem with such uniform scaling and clamping is that the detail in the darker portions of the image are clamped to black and the detail in the brighter portions of the image are clamped to white. This represents a loss of detail in both clamped and non-clamped areas.

In this paper we present a simple, fast method for displaying high dynamic range images. We compute the local adaptation luminance for every pixel of a high dynamic range image by the use of an efficient image segmentation technique. The segmented image is then mapped onto an adjacency graph and simple graph operations are used to simplify the image by the process of assimilating small nodes into larger surrounds. The assimilation process can be configured to preserve high dynamic range appearance in the display image. The simplified image is then used in conjunction with a tone-reproduction operator to reduce the dynamic range of the input high dynamic range image for display.

2 Previous Work

Tone-reproduction operators have been used for some time to accurately display images of scenes with wide absolute range of illumination using display surfaces with low dynamic range [7][11][16][19]. In general, these authors build on psychophysical data to derive a mapping from the dynamic range of the image onto a smaller range while preserving visual appearance. These operators can be used to display high dynamic range images and some of them work by multiplying each pixel value in the high dynamic range image with local scale factors to derive the display pixel value. The local scale factors are derived from the tone-reproduction operator's model of the visual adaptation and the local adaptation luminance of the scene.

The local scale factors for local adaptation have been approached from multiple directions [2][10][13][15]. Most of these techniques compute the local adaptation luminance by finding the geometric or arithmetic mean of pixel luminance in a local neighborhood. Some of these techniques use spatially averaging linear filters or even linear filter hierarchies. These techniques invariably produce halo artifacts around regions of high contrast. Halos form on the dark side of the high dynamic range edge because the local adaptation luminance is overestimated from the presence of bright pixels. Conversely, the adaptation luminance on the bright side is underestimated due to the presence of dark pixels in the local neighborhood. This causes a gradient reversal and hence a dark or bright halo around high contrast edges in the image (see the image on the 3rd row in Figure 2 for halo artifacts). This artifact detracts from the quality of the image.

In their recent work Pattanaik and Yee [12], Ashikhmin [1] and Reinhard et al. [14] propose techniques based on local adaptation that does not produce halo artifacts. Pattanaik and Yee use the weighted mean of pixel luminances in the neighborhood as their local adaptation luminance. In their method weighting is dependent on the ratio of the luminance of neighboring pixels and the pixel under consideration. Ashikhmin's and Reinhard et al.'s methods use the mean of a circular neighborhood as the local adaptation, where the radius of the local neighborhood is adaptively computed based on the local contrast. For a high contrast neighborhood the radius is kept smaller than the radius in an uniform neighborhood. The adaptive per pixel computation used in these techniques is relatively expensive.

Our method is based on local adaptation luminance computation. It is designed to avoid the gradient reversal problem common in many local adaptation based methods and thus avoids halo artifacts. The local adaptation computation in our method is region based instead of pixel based and is hence much faster.

Ward-Larson et al. [20] invented a technique that avoids halo artifacts by performing operations on the histogram of adaptation luminance values in the image. The display contrast in the resulting display image is sometimes reduced since the scene to display luminance mapping is strictly monotonic in their method. Our method is not limited to monotonic scene to display luminance mappings and hence preserves perceived contrast in most cases.

Tumblin et al. [17] provided a technique that is halo-free, but requires that the source image be separated into different layers of lighting and reflectance. This might not always be possible, especially from images derived from photographs. Our technique is not limited to computer-generated images.

By the clever use of partial differential equations from the field of anisotropic diffusion, Tumblin and Turk [18] came up with a technique called LCIS (Low Curvature Image Simplifier) that reduces dynamic range while preserving detail. The technique works very well in making details in an image visible. However, LCIS is computationally expensive. Our technique can be up to 50 times faster than LCIS. At the same time, our method can operate in a mode that preserves detail.

Durand and Dorsey's recent work [5] proposes the use of bilateral filtering for extracting details. The resulting method is more efficient than the LCIS based method. Fattal et al.'s recent work [6] is based on computing and attenuating large luminance gradients that exist in high dynamic range images. Reinhard et al's recent work [14] proposes a non-linear compression function for compressing the dynamic range. This function is very similar to the tone mapping function proposed by Pattanaik et al. [11] in their dynamic adaptation work, and is effective within a limited dynamic range. Reinhard et al. [14] use local adaptation luminance in their compression function to handle images with larger dynamic range.

3 Method

Our method attempts to reduce the dynamic range by finding the appropriate adaptation luminance for every pixel. This luminance can be used in conjunction with psychophysically derived tone-reproduction operators for creating an accurate display image. Our method can be adjusted to extract detail similar to LCIS [18].

Our algorithm for local adaptation proceeds in four steps, segmentation, grouping, assimilation and layer averaging. Now we define some terms before we discuss the implementation. The output of the segmentation process puts pixels into *categories*. The categories are then clustered by spatial contiguity into *groups*. A process called assimilation combines different groups together. Pixels belonging to each group are set to the group's luminance value and the resulting image is called a *layer*. Multiple layers are generated by repeating the segmentation process with slightly different bin sizes and then are averaged together to obtain the local adaptation luminance image. User specified parameters are marked in **bold**.

Step 1: Segmentation. The method begins by performing a simple image segmentation on the $\log_{10}$ luminance of the high dynamic range image by binning each pixel into categories which are spaced apart in units of bin size (see Equation 1 and Equation 2).

Equation 1: bin_size = **bin_size1** + (**bin_size2** – **bin_size1**) * (layer / (**max_layers** –1));

Equation 2: category(x,y) = ($\log_{10}$ luminance(x,y) – minimum image $\log_{10}$ luminance) / bin_size;

where layer runs from 0 to **max_layers**-1, and **max_layers** is user specified. The bin size is linearly interpolated between user specified **bin_size1** and **bin_size2**.

Step 2: Grouping. A flood fill is performed in a breadth first manner so that contiguous pixels of the same category are assigned to a group. During the flood fill, the sum of pixel log luminance for the group is accumulated, as well as the list of locations of pixels belonging to the group. At the same time, if the flood fill encounters an adjacent group, the adjacency list of both groups are updated to reflect the fact that the groups are next to each other. The adjacency list is a list that keeps track of which groups are neighbors to the current group being flood filled. Group luminance is calculated as the sum of pixel log luminance values divided by the number of pixels in each group.

Step 3: Assimilation. All the groups are checked to locate singletons that are groups with only one neighbor. The larger group of the two neighbors then assimilates the smaller group if the number of pixels in the larger group exceeds a user specified **big_threshold** and if the number of pixels in the smaller group is less than a user specified **small_threshold**. Moving the pixel locations from the smaller group to the larger group and discarding the group luminance value from the smaller group summarizes the assimilation process. The list of groups is again traversed and groups with pixel count smaller than **small_threshold** have their neighbor list traversed. The largest neighbor with a pixel count larger than **big_threshold** assimilates the smaller group. Each pixel is then set to the luminance value of the group it belongs to. The image that is formed from this segmentation, grouping and assimilation process is called a layer.

Step 4: Layer Averaging. Multiple layers separated by slightly different bin sizes are then averaged together to obtain the final local adaptation luminance image which is then handed off to a tone-reproduction algorithm for processing.

Figure 1 (a) – (g) below graphically depicts each step of the algorithm in calculating the local adaptation luminance using our technique. Step (h) is the local adaptation luminance calculated using a box filter for comparison. The image segmentation via binning ensures that pixels of very different luminances are not averaged together. Figure 1 (a) and (b) show the segmentation of two layers of different bin sizes. The segmentation process prevents halos or gradient reversals from forming. A bin size of one $\log_{10}$ unit allows the capture of contrast up to 10:1. Figure 1 (c) and (d) show the grouping process, whereby spatially contiguous pixels of the same binned luminance are grouped together. The images are false colored by group. Figure 1 (e) and (f) are the result of the assimilation process. By merging small regions of pixels with very different luminances from the surround (such as specular highlights and shadows) with the surround, the assimilation process allows context dependent intensity mapping. Thus pixels with a given luminance surrounded by darker pixels are assigned the adaptation luminance of the darker pixels and are mapped to look much brighter than the same pixels surrounded by brighter pixels. For example, the letters to the right of the lamp on the wall are 'assimilated' into the luminance of the wall, which accentuates the darkness of the letters during the tone mapping process. This can be seen in the differences between image (c) and (e). The assimilation process gives a high dynamic range appearance to the display image. Finally, the averaging of luminances from different layers removes the banding caused by the image segmentation process. The result is show in Figure 1 (g) with a box filtered version in (h) as comparison.

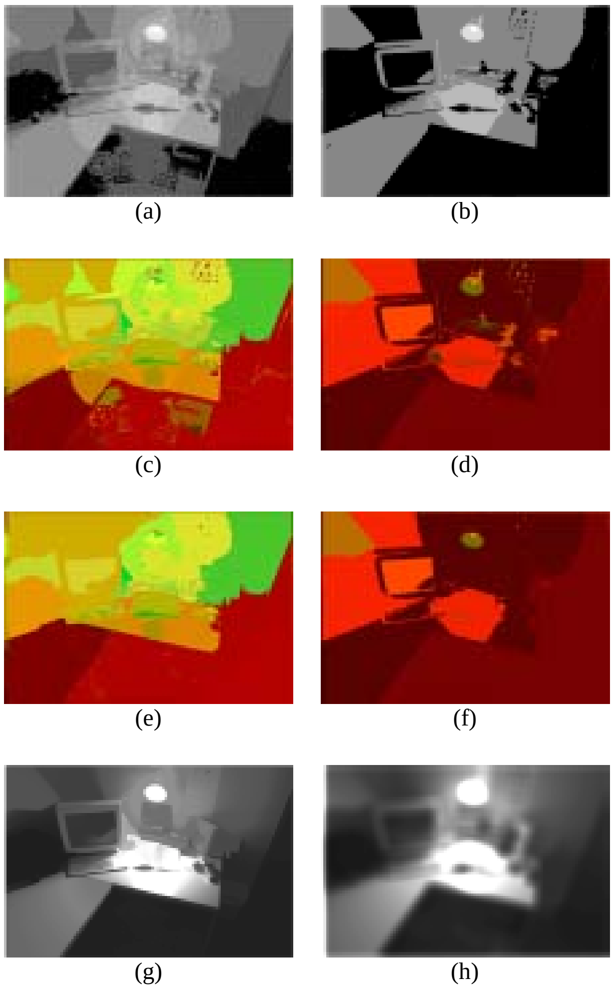


Figure 1: Graphical overview of our algorithm. (a) and (b) show the output of the segmentation process for two different bin sizes. (c) and (d) show the output after the flood fill grouping process. (e) and (f) show the output after the assimilation process. (g) is the computed local adaptation luminance from our algorithm and (h) is the local adaptation luminance computed via a box filter.

4 Implementation

We implemented the Segmentation and Adaptive Assimilation Algorithm using C++ and the Standard Template Library. See Table 1 for the data structures used. Each group maintains a vector of the coordinates of the pixels who are members of the group (MEMBERS), and a list of neighboring groups adjacent to it (NEIGHBORS). During the flood fill process, the SUM and COUNT variables are kept updated. We initially implemented the adjacency graph as a binary adjacency matrix but achieved a significant speedup by representing the graph as an adjacency list (NeighborContainer) because the flood fill process generates a sparse graph. As a preprocessing step, we replaced zero luminance values with the smallest non-zero luminance value, the minimum image luminance, in order to avoid taking the logarithm of zero.

```
struct Point { unsigned short x,y; }
typedef std::vector<Point> PointVector;
```

```
typedef std::list<unsigned short> NeighborContainer;
struct Group_Record {
        PointVector MEMBERS;
        NeighborContainer NEIGHBORS;
        double SUM;
        unsigned int COUNT;
}

typedef std::vector<Group_Record *> Groups;
```

Table 1: Data Structures used in the Adaptive Assimilation Algorithm

The five user specified paramenters are **max_layers**, **bin_size1, bin_size2, small_threshold** and **big_threshold**. The variable **max_layers** controls how many layers to generate. It is used to smooth out the boundaries generated in the segmentation process. Typical values of **max_layers** are between 16 to 96, depending on how smoothly the luminance varies in the image. A value of 16 is sufficient for images with a lot of high frequency content and a value of 96 is appropriate for images with very smooth luminance gradients. Changing the bin sizes changes the contrast captured. Larger bin sizes increase the captured contrast. Most of the time, **bin_size1** can be set to 0.5 $\log_{10}$ units while **bin_size2** can be set to 1.0. We found out through experimentation that the bin sizes should be at least 0.5 $\log_{10}$ units apart. Also, the bin sizes should not be too large or else vastly different luminance values might be averaged together. The threshold variables determine when larger groups assimilate small groups. Typical values for **small_threshold** range from 0.001% to 3% of the number of pixels in the image. Areas occupying less than one degree of visual field do not significantly influence adaptation. One could set **small_threshold** value to be less than 1 degree of visual field. In some images, the total visual field occupied by the scene is not known. In these cases, a threshold value based on a small percent of image pixels has been found to be sufficient. Setting **small_threshold** to 0.0 turns off the assimilation process and can be used to capture detail similar to LCIS. Values for **big_threshold** are between 1% and 10%.

Step 1: Segmentation. The bin size determines the maximum contrast of the detail that is reproduced in the display image. A bin size of 1.0 log10 units allows the capture of detail contrast up to 10:1. Care should be taken not to set this parameter to too large a value so that pixels of vastly different dynamic range are not grouped together. In our implementation, we allow bin size to vary between user selected **bin_size1** and **bin_size2** log10 units.

Step 2: Grouping. The flood fill algorithm was implemented by keeping a queue of neighboring pixels. The pixels are explored in a breadth first manner and pushed onto the queue if they have not been explored before.

Step 3: Assimilation. This was implemented in the following manner. Suppose group G was being assimilated by group N. Then,

N.SUM = (N.COUNT + G.COUNT) * N.SUM / N.COUNT
N.COUNT = N.COUNT + G.COUNT
N.MEMBERS = N.MEMBERS + G.MEMBERS
G.COUNT = 0

In this way, the average luminance values of pixels in group G are replaced by those in group N. The average N.SUM/N.COUNT, computed at the end of the assimilation process, was assigned to each pixel of a group.

In Step 4, Layer Averaging, was implemented as:

Local adaptation luminance (x,y) = $\sum$Layers(x,y)/**max_layers.**

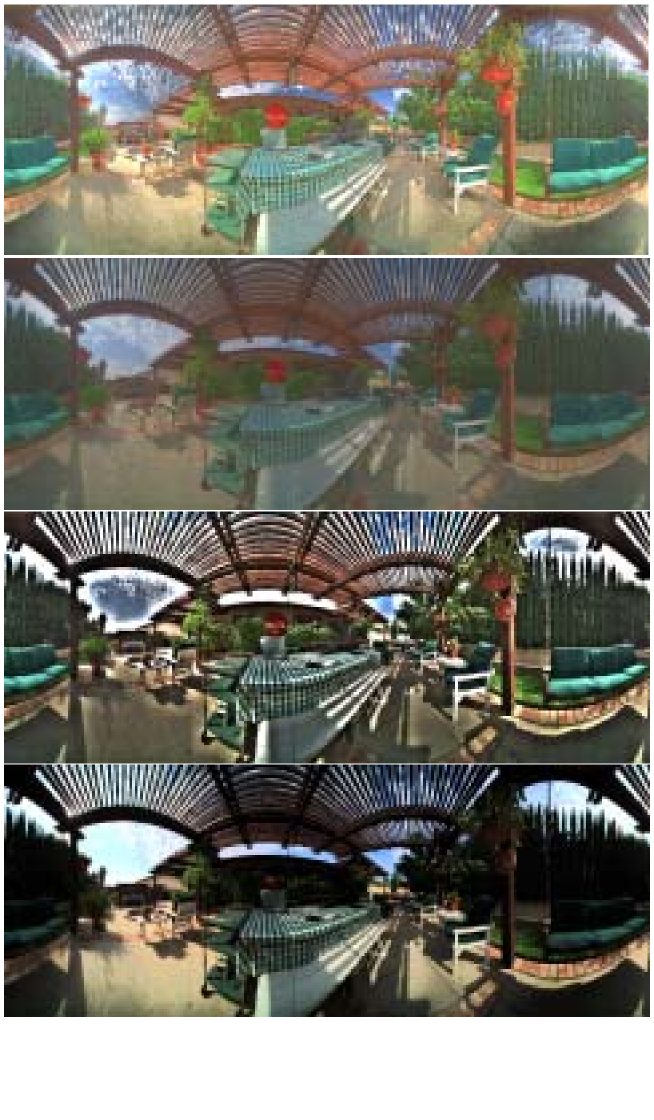
5 Results

We used the algorithm described in the previous section in conjunction with the tone-reproduction algorithms of Pattanaik et al. [11] (PTYG), Ferwerda et al. [7] (FPSG) and Tumblin and Rushmeier [16] (TR) for accurate display of high dynamic range images. Each of these tone-reproduction algorithms computes the display image pixel value from an input image pixel and the adaptation luminance for the input scene. The output of our segmentation and adaptive assimilation algorithm provides the per pixel adaptation luminance for such a display computation. The images shown in this paper are the output of the TR algorithm [17]. The TR algorithm attempts to reproduce the perceived brightness from the input pixel values and the adaptation luminance. The algorithm is very simple, and we provide the C++-code of the TR algorithm in appendix for convenience. The exact detail of the algorithm may be found in [16]. We provide the timing for computation using each of the three algorithms. Additionally, local average of the luminance of pixels in a 33×33 window around every pixel was computed using fast dynamic programming and also used in conjunction with the three tone-reproduction algorithms. This local average was used for comparative purposes. The LCIS algorithm (Tumblin et al. [18]) and WRP (Ward et al. [20]) algorithm were also implemented as points of comparison. All algorithms were re-implemented by the authors, using existing code for reference wherever possible. Refer to Table 2 for the running times of the various algorithms. Our method was found to be about two orders of magnitude faster than LCIS [18] and did not exhibit the halo artifacts that plagued the methods that use a geometric mean. The WRP technique was equally fast and did not exhibit halo artifacts. But, our algorithm is predictable in its detail preserving behavior and is able to provide context dependent mapping.

(a) Image Name & Resolution	(b) Local Averaging (in log domain)	(c) Segment and Assim-ilation	Tone mapping using local adaptation luminance from (b) or (c)				(h) LCIS
			(d) WRP	(e) PTYG	(f) FPSG	(g) TR	
Park (384x256)	0.15	2.0	0.12	1.32	0.13	0.2	178
Price Western (818x320)	0.25	9.37	0.3	3.07	0.35	0.72	479
Desk (768x512)	0.71	34	0.44	4.61	0.46	0.8	796
Bridge (512x384)	0.4	21.2	0.2	2.4	0.2	0.24	415

Table 2: Running times in seconds. Column (a) lists the image name and resolution. Columns (b) and (c) are the running times for computing the local adaptation luminance by averaging in the local window and by segmentation and adaptive assimilation (our approach) respectively. Columns (d), (e), (f) and (g) list the running times for tone mapping algorithms of WRP, PTYG, FPSG and TR respectively. The output of (b) or (c) are fed to (d), (e), (f) or (g) for the purpose of contrast reduction. The times in column (h) are for LCIS, which is a technique for contrast reduction. LCIS was run in three passes of 500 iterations each. All times are for a Pentium III 1.1 GHz with 384 MB RAM running Windows ME. Note that the adaptive assimilation algorithm is not constrained by power of two limits. The image aspect ratio can also be arbitrary.

The images in Figure 2 are the display of a high dynamic range panoramic image of the Price Western Hotel. The first row image was computed using our algorithm along with TR tone-reproduction algorithm. It demonstrates the detail preserving, range compression nature of our algorithm. Figure 2 compares the result with the output of other high dynamic range display algorithms. As can be seen from the figure, we maintain the detail LCIS provides at a fraction of the computational time. We avoid the halos that arise when using the average of a local window as the local adaptation and we have more detail than the output of WRP algorithm. An image that was multiplied with a global scale factor is supplied for comparison. At the same time, we provide user control for preserving high dynamic range appearance in the display image. The settings for Price Western are **max_layers=16, bin_size1 = 0.5, bin_size2 = 1.0, small_threshold = 0%** and **large_threshold = 3%.** These settings were used for producing the first row.



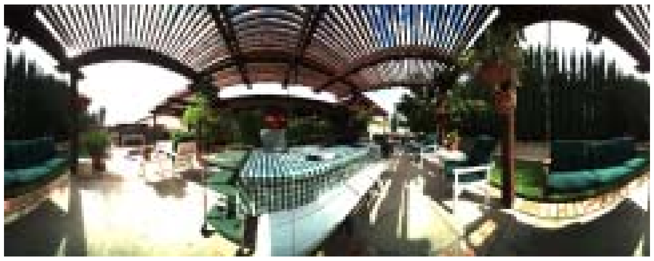
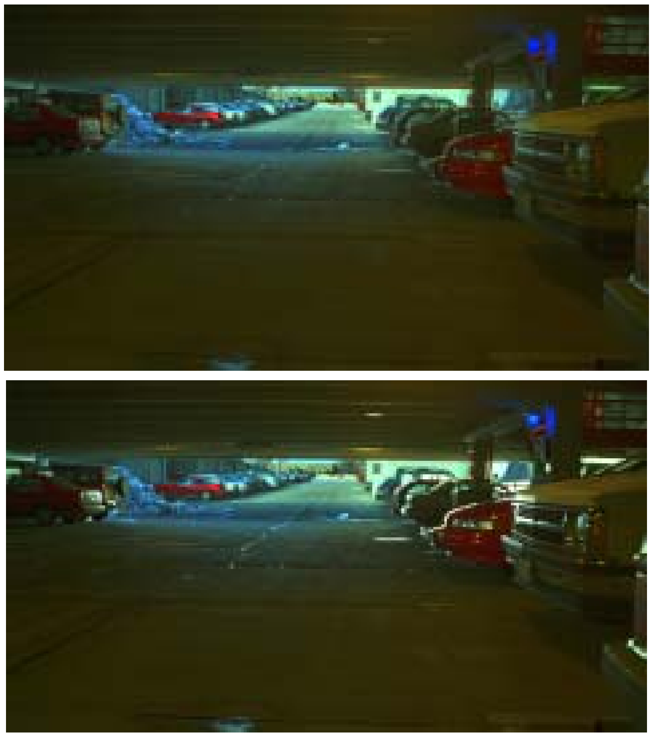


Figure 2: Price Western Hotel panorama image. Techniques in row order from top to bottom: Our Segmentation and Assimilation method with TR, LCIS, Local averaging with TR, Local averaging with WRP, simple scaling.

In another comparison test on a parking garage (see Figure 3), our technique (top 2 row images) gives superior results than local averaging with the TR tone mapper and WRP technique. Our technique shows no halos (unlike local averaging), while clearly displaying the row of cars (at a fraction of the computation time of LCIS), and has improved detail (compared to WRP). An image that was multiplied with a global scale factor is supplied for comparison. The settings for the results are **max_layers=32**, **bin_size1 = 0.5, bin_size2 = 2.0, small_threshold = 0.1%** and **large_threshold = 10%.** These settings maintain a sense of dynamic range. For only detail preserving range compression, **small_threshold** was set to 0. Note how our technique brings out the specular highlights on the yellow truck to the right.

In the Desk Image, (see Figure 4) our technique does not have the halos compared to local averaging, and has more detail than the results from WRP algorithm. For example, the logo on the light bulb is visible with our technique and the text on the book can be seen clearly. The settings for this image are: **max_layers = 96, bin_size1 = 0.5, bin_size2 = 1.25, small_threshold = 2, big_threshold = 2%.**



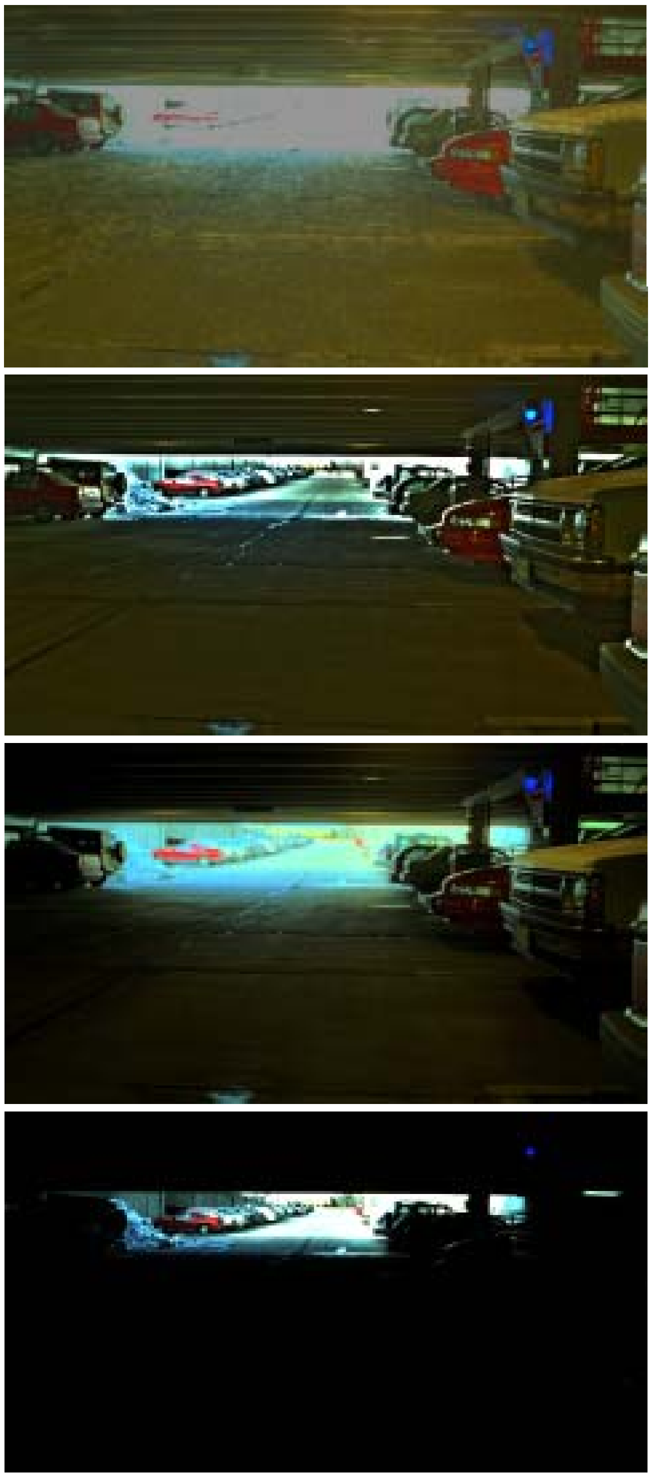


Figure 3: Parking Garage. Techniques in row order from top to bottom: Our Segmentation method with TR (detail preserving), Our Segmentation and Assimilation method with TR (detail and high dynamic range appearance), LCIS, Local averaging with TR, Local averaging with WRP, simple scaling.

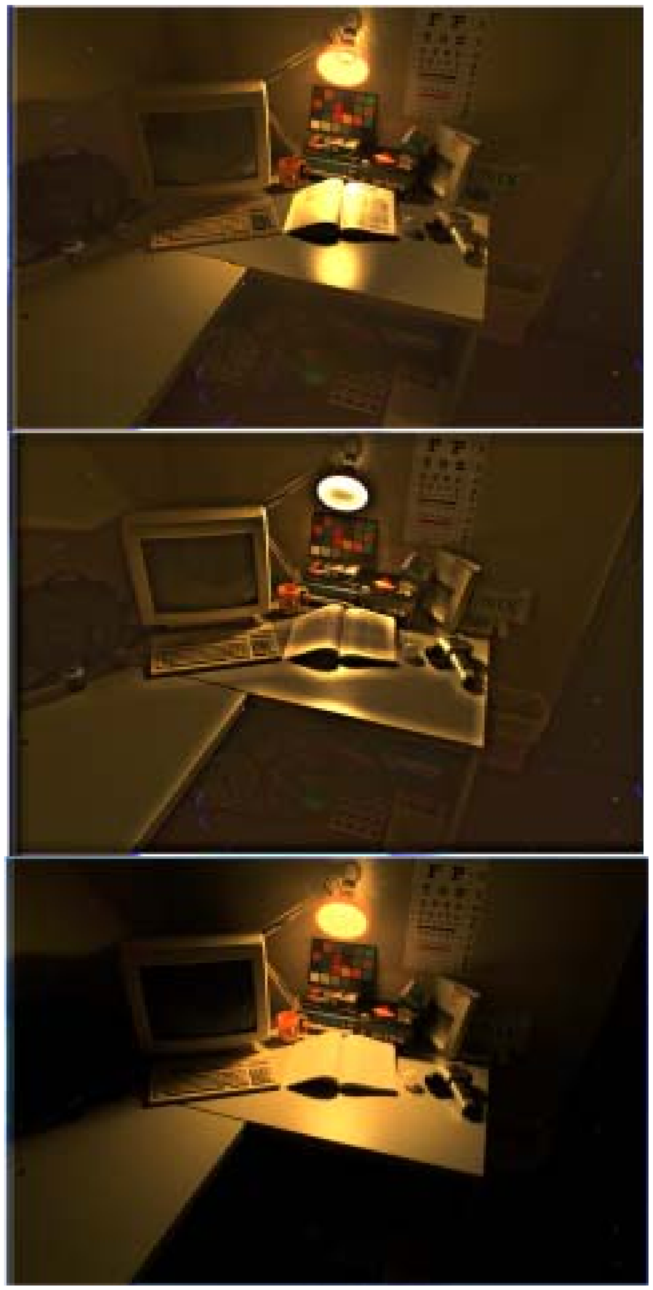


Figure 4: Desk. Top – Our method with TR. Middle – Local averaging with TR. Bottom – Local averaging with WRP.

Finally, in Figure 5 we show the Bridge Image. Our technique (top row) does a better job in dynamic range compression than WRP algorithm (bottom row) while retaining the detail. The image on the center row shows a simple scaled version of the original.

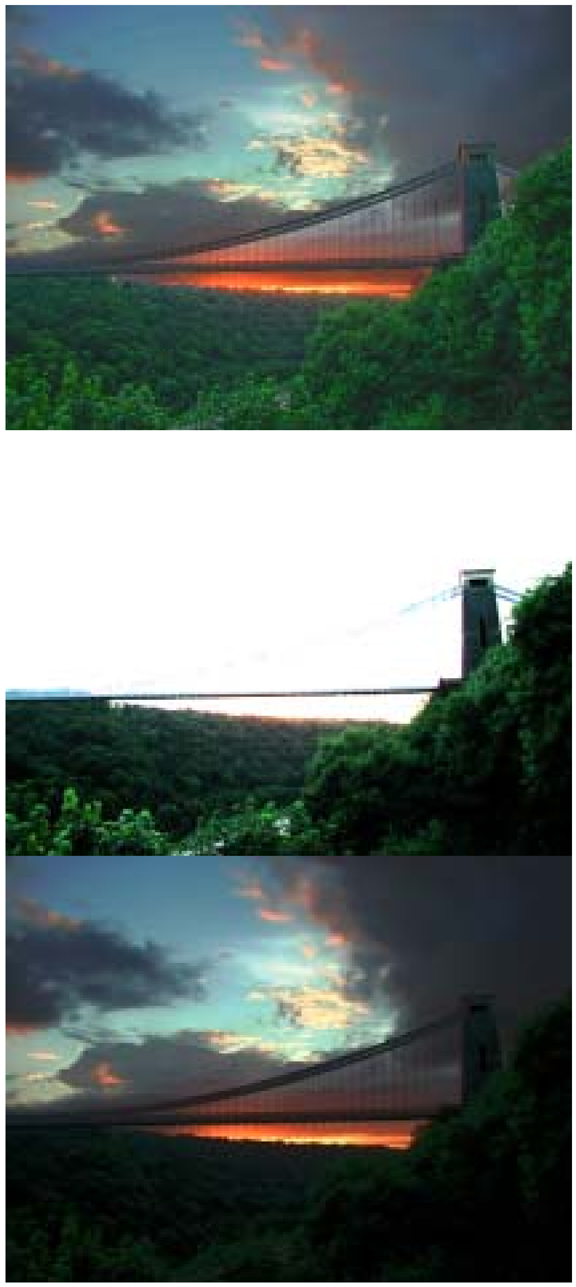


Figure 5: Bridge. Top – Our method with TR. Middle – Simple scaling. Bottom – WRP.

6 Discussion

Although our novel approach has worked successfully with several high dynamic range images, it has some weaknesses that could be addressed with further research. One problem is the number of parameters needed to fine-tune the algorithm to a particular image. For example, the number of bin sizes could perhaps be obtained automatically by some heuristic. Also, the algorithm has not been tested with user studies to see if the images produced are visually accurate.

7 Conclusions

We present a fast, efficient method of local adaptation that makes use of image segmentation, flood fills and graph operations. The method has been demonstrated with three different tone-reproduction algorithms with good results. Our method does not suffer from halo artifacts and preserves detail while improving contrast over previous techniques. For our future work we plan to implement k-means clustering [8] for the segmentation in order to relieve the user from specifying bin sizes. Also, we would like to investigate smarter and higher-level ways of assimilating groups that take into account the shape as well as the size of the groups.

Thanks go David Hart, Mahesh Ramasubramanian and Reid Gershbein for proof reading, and to our anonymous reviewers for their suggestions. Greg Ward provided the Bridge image. The Desk and Garage image were provided by Cornell University. The Price Western Hotel Panorama was from Spheron3D.

References

[1] M. Ashikhmin. A tone mapping algorithm for high contrast images. In Proceedings of Eurographics Workshop on Rendering 2002, 151- 161. June 2002.

[2] K. Chiu, M. Herf, P. Shirley, S. Swamy, C. Wang, and K. Zimmerman. Spatially nonuniform scaling functions for high contrast images. In *Proceedings of Graphics Interface '93*, 245-254. May 1993.

[3] P. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of SIGGRAPH '97*, 369-378. August 1997.

[4] P. Debevec. Image-based lighting. In *SIGGRAPH 2001 Course Notes*, 12-17. August 2001. (http://www.debevec.org/IBL2001).

[5] F. Durand, J. Dorsey. Fast bilateral filtering for the display of high dynamic range images. In *Proceedings of SIGGRAPH 2002*, 257-268. July, 2002.

[6] R. Fattal, D. Lischinski and M. Werman. Gradient domain high dynamic range compression. In *Proceedings of SIGGRAPH 2002*, 249-256. July, 2002.

[7] J. A. Fewerda, S. Pattanaik, P. Shirley and D. P. Greenberg. A model of visual adaptation for realistic image synthesis. In *Proceedings of SIGGRAPH '96*, 249-258. August, 1996.

[8] J. McQueen. Some methods for classification and analysis of multivariate observations. In Proceedings of the 5th Berkeley Symposium on Math. Stat. And Prob. Vol 1:281-296. 1967

[9] S. K. Nayar and T. Mitsunaga. High dynamic range imaging: spatially varying pixel exposures. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* June 2000.

[10] S. N. Pattanaik, J. A. Ferwerda, M. D. Fairchild and D. P. Greenberg. A multiscale model of adaptation and spatial vision for realistic image display. In *Proceedings of SIGGRAPH 98*, 287-298. July 1998.

[11] S. N. Pattanaik, J. E. Tumblin, H. Yee and D. P. Greenberg. Time-dependent visual adaptation for fast realistic image display. In *Proceedings of SIGGRAPH 2000*, 47-54. July 2000.

[12] S. N. Pattanaik, Hector Yee. Adaptive Gain Control for High Dynamic Range Image Display. In *Proceedings of Spring Conference in Computer Graphics (SCCG2002),* 83-88. April 2002.

[13] Z. Rahman, D. J. Jobson, and G. A. Woodell. Multi-scale retinex for color image enhancement. In *Proceedings, International Conference on Image Processing,*volume 4, 1003-1006. June 1996.

[14] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. In *Proceedings of SIGGRAPH 2002*, 267, 276. July, 2002.

[15] C. Schlick. Quantization techniques for visualization of high dynamic range pictures. In *Photorealistic Rendering Techniques*, Proceedings of the 5th Eurographics Rendering Workshop, 7-20. 1995.

[16] J. Tumblin and H. Rushmeier. Tone reproduction for computer generated images. In *IEEE Computer Graphics and Applications*, 13(6):42-48. November 1993.

[17] J. Tumblin, J. K. Hodgins and B. Guenter. Two methods for display of high contrast images. In *ACM Transactions on Graphics*, 18(1). January 1999.

[18] J. Tumblin and G. Turk. LCIS: A Boundary Hierarchy For Detail-Preserving Contrast Reduction. In *Proceedings of SIGGRAPH '99*, 83-90. 1999.

[19] G. Ward. A contrast-based scale factor for luminance display. In Graphics Gems IV, 415-421. 1994.

[20] G. Ward Larson, H. Rushmeier, and C. Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. IEEE Transactions on Visualization and Computer Graphics, 3(4):291-306, October-December 1997.

Appendix

```
void TR_tonemap(
  inR, inG, inB,         //Input pixel RGB values
  AdaptationLuminance,//per pixel adaptation luminance computed by our algorithm
  outR, outG, outB      //Output pixel RGB values
){
  //  Convert cd/m2 to Lambert
  double L_wa = cdm2ToLambert(AdaptationLuminance);
  double L_w = cdm2ToLambert(rgb2luminance(inR,inG,inB));
  //  Compute Color Ratio
  double f_r = inR/L_w, f_g = inG/L_w, f_b = inB/L_w;
  /*
     Stevens equation gives the Brightness in Brils.
                log10 (B) = 0.004[(S-27)(8.4-R)-108]
                where
                  S = 100 + 10 log10 (L_wa)
                  R = 10 log10 (L_wa / L_w)
  */
  double S_w = 100.0 + 10.0*log10(L_wa);
  double R_w = 10.0 * log10(L_wa/L_w);
  /*
     We want the brightness corresponding to the world luminance
     to be same as that perceived from the display
     i.e. (S_w-27)(8.4-R_w) = (S_d-27)(8.4-R_d)
  */
  double L_dMax = cdm2ToLambert(MAX_DISPLAY_LUMINANCE);
  double L_da = cdm2ToLambert(DISPLAY_ADAPTATION_LUMINANCE);
  double S_d = 100.0 + 10.0*log10 (L_da);
  //  Solve for R_d
  double R_d = 8.4 - (S_w-27)*(8.4-R_w)/(S_d-27);
  /*
     R_d = 10log10(L_da/L_d)
     L_d (in Lambers) = L_da 10^(-0.1*R_d)
  */
  double L_d = lambertToCdm2(L_da *
                    pow(10,-0.1*R_d))/MAX_DISPLAY_LUMINANCE;
  //  Compute display pixels values in the range of 0 to 1.
  outR = MIN(1.0,L_d*f_r);
  outG = MIN(1.0,L_d*f_g);
  outB = MIN(1.0,L_d*f_b);
}
```