

Coherent Lightcuts

Thomas De Bodt, Philip Dutré

Abstract—Coherent Lightcuts is an extension to Lightcuts that exploits coherence between neighbouring pixels in an image. Unlike the Reconstruction cuts algorithm, proposed by the authors of Lightcuts, we achieve this without approximating the Lightcuts solution.

We efficiently detect coherence in both large and small regions in the image using a new coherence heuristic. Though we only focus on reducing the cost of the error estimation and other overhead of Lightcuts, we achieve a significant speedup.

We believe, however, that existing techniques to exploit ray coherence can be efficiently combined with our algorithm. This is possible due to different order of computations in our algorithm. This way a much larger speedup could be achieved.

Index Terms—Computer Graphics, Rendering

I. INTRODUCTION

REALISTIC models of a real world scene often require using many or complex lights. Because most illumination can be approximated by the direct illumination from many, simple lights, this is also called the many light problem. Yet most techniques for realistic image synthesis do not handle this situation well. Lightcuts [1] is a recent algorithm proposed by Walter et al. that scales sublinearly in the number of light sources. In this algorithm light sources with similar contribution are approximated using a single, brighter light.

The Lightcuts algorithm calculates a unique clustering of lights for each point. Yet the approximations used for neighbouring pixels in an image are often very similar. This can also clearly be seen in Fig. 1. Here the regions with an identical clustering of the lights are marked for a simple scene [2]. The authors of the Lightcuts algorithm propose a technique, named Reconstruction cuts, to exploit this coherence. However, this algorithm relies on interpolation, which inevitably results in missing fine details.

The technique proposed in this paper exploits the coherence in the Lightcuts algorithm without approximating its solution. In our implementation of Lightcuts, about 50% of the total time is spent estimating the contribution of a single cluster. The most expensive part here is testing a shadow ray to determine visibility of the cluster. Because there already exist a number of techniques to exploit the similarity of several shadow rays (e.g. MLRTA by Reshetov et al. [3]), we focused on efficiently determining coherent regions and reducing the overall cost of the other calculations. An important part of these is estimating the error of a cluster ($\pm 30\%$ of the total time).

The next section of this article discusses the previous work, including a brief overview of Lightcuts. Section 3 presents our algorithm to exploit the coherence in Lightcuts. The results of this algorithm are discussed in Section 4. The conclusions are in Section 5. There is also an Appendix with details about the coherence heuristic used.

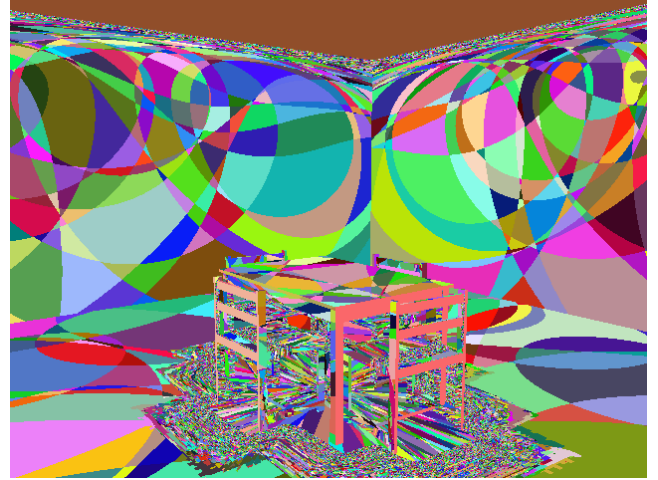


Figure 1. The regions in a simple scene where the clustering used by the Lightcuts algorithm is identical.

II. PREVIOUS WORK

A. Lightcuts

The Lightcuts algorithm handles many lights efficiently by approximating lights with similar contributions by a single, brighter light. To quickly determine a good clustering, a binary tree containing all lights is built in a preprocess step. In this light tree, a cut is determined for each visible surface point. A cut is a set of light clusters such that each light is contained in exactly one cluster. The cut for a specific point is calculated by starting with a very rough cut and iteratively refining the cluster with the largest estimated error until a perceptual heuristic is met. The heuristic, based on Weber's law, allows a constant relative error for each cluster (e.g. 2%). This avoids visible discontinuities between different pixels due to a different set of clusters being used in the approximation.

To estimate the contribution of a cluster, all lights are concentrated at the position of a single, representative light in this cluster. To estimate the error of a cluster, an upper bound on its contribution is calculated by calculating upper bounds on all individual terms. From this maximum absolute error, the maximum relative error is estimated using the estimate of the current cut.

B. Coherence

Several techniques exploit coherence by interpolation of some quantity. The most famous technique is probably the one introduced by Gouraud [4], which simply interpolates the colors calculated at the corners of a triangle. Pighin et al. [5] use Gouraud interpolation in the image plane to interpolate the colors of a sparse set of samples. The sample locations

lights in the cluster. Because the cuts for all gather points are most likely not identical in Lightcuts, we also check whether they are still coherent. If this is not the case, the block is subdivided in 4 equally sized blocks. A block is considered incoherent if the next refinement is not required for some gather point in the block. This can either be due to a higher total illumination (i.e. a higher allowed absolute error) or a lower absolute error for this cluster.

In Fig. 2, the Coherent Lightcuts algorithm is illustrated. Initially the image is subdivided into 4 blocks. Only the construction of the cut for the green block is shown in the rest of this figure. This cut is refined once (i.e. the green cut) after which the block is detected as incoherent. This results in its subdivision into 4 smaller blocks, which will each refine this cut individually. The first block satisfies the stop criterion after one refinement (i.e. the orange cut), so an accurate estimate for the illumination in each gather point of this block is now known. The cut is also refined once for the second subblock (i.e. again the orange cut). However this block is now detected again as incoherent, causing a second subdivision. The last step shows one more refinement for one of these blocks (i.e. purple block and cut) after which it also satisfies the stop criterion.

B. Block Subdivision

When a block is subdivided, the estimated error for each cluster on the cut is also re-evaluated. After all, the reason for its subdivision is that the estimated error could be below the threshold of a gather point in the block. Because estimating the error is not very cheap, unnecessary recalculations are avoided. We use the improved perceptual heuristic from Multidimensional Lightcuts, so the error for a cluster is only re-estimated if:

$$E_{\text{old}} > a \min(L_{\text{cut}}) + \frac{L_a}{10} \quad (1)$$

The estimated error for the parent block is E_{old} , the allowed relative error is a , the lowest estimated illumination is $\min(L_{\text{cut}})$ and L_a is the adaptation luminance for the image.

When a block with less than 10 gather points becomes incoherent, it is subdivided into its individual gather points. The cut for these points is completed using regular Lightcuts. This avoids blocks consisting of 1 pixel (e.g. when a block of 3×3 pixels is subdivided). Similar to regular blocks, the error for the clusters on the cut is re-estimated to avoid unnecessary refinements.

C. Coherence Heuristic

To determine whether a block is incoherent, the smallest possible upper bound on the contribution of the next cluster that will be refined is estimated. This minimal upper bound U_{min} , is used as an estimate for the lowest error in the block. So a block is considered incoherent, using the perceptual error metric from Multidimensional Lightcuts, if:

$$U_{\text{min}} \leq a \max(L_{\text{cut}}) + \frac{L_a}{10} \quad (2)$$

Where $\max(L_{\text{cut}})$ is the highest estimated illumination in a gather point of the block.

To calculate the minimum upper bound between a number of gather points and a cluster, a minimum upper bound on each term is calculated. For the geometric term of a cosine weighted point light, this is:

$$\min(G_u) = \frac{\min_{\mathcal{G}}(\max_{\mathcal{L}}(\cos \theta_G))}{\max_{\mathcal{G}}(\min_{\mathcal{L}}(d))^2} \quad (3)$$

Here \mathcal{G} is the bounding box around the gather points, \mathcal{L} the bounding box of the light cluster, θ_G the angle between the normal of the light cluster and the direction from a point in \mathcal{L} to a point in \mathcal{G} and d the distance between points in both bounding boxes. Details about the calculation of the $\min_{\mathcal{G}}(\max_{\mathcal{L}}(\theta))$ and $\max_{\mathcal{G}}(\min_{\mathcal{L}}(d))$ terms are given in Appendix A. For the material term, only the minimal upper bound on the diffuse part is estimated:

$$\min(M_u) = \rho_{\text{max}} \min_{\mathcal{G}}(\max_{\mathcal{L}}(\cos \theta_M)) \quad (4)$$

Here θ_M is the angle between the normal in a point in \mathcal{G} and the direction to a point in \mathcal{L} .

Finally the estimate of the minimal upper bound is:

$$U_{\text{min}} = \min(M_u) \min(G_u) I \quad (5)$$

Here I is the intensity of the light cluster.

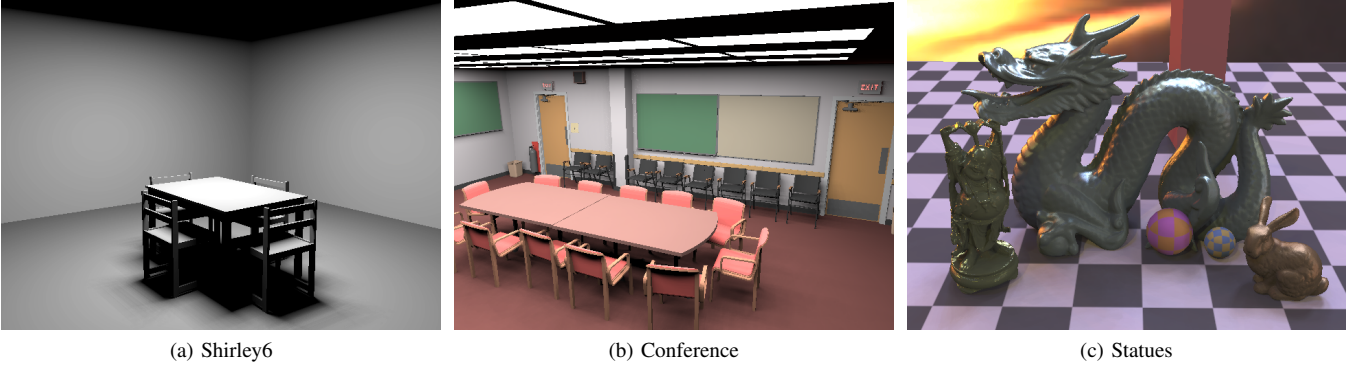
IV. IMPLEMENTATION DETAILS

Our implementation of Lightcuts has a number of small modifications compared to the original algorithm. As mentioned in the previous section, we use the improved perceptual heuristic used in Multidimensional Lightcuts. This heuristic accounts for the lower sensitivity to errors in dark regions of the human eye. A second difference is the BRDF we use to model glossy surfaces. Instead of the Phong or Ward model, we use the microfacet model proposed by Torrance and Sparrow [15], combined with the microfacet distribution proposed by Blinn [16]. The upper bound on this BRDF uses a trivial value for both the Fresnel and geometric attenuation terms. The microfacet distribution is bound using the half angle bound described by Walter [17]. A final modification is the use of a trivial lower bound on the contribution of a cluster (i.e. 0), to improve the accuracy of the error estimate.

$$|L - \hat{L}| \leq \max(\hat{L}, U_L - \hat{L}) \quad (6)$$

Here L is the contribution of the cluster, \hat{L} the estimated contribution and U_L the upper bound on its contribution. Using this upper bound reduces the cut size by 5%–10%, for the scenes used in our results, at a negligible extra cost. A consequence of this is, however, that the automatic clamping of clusters of multiple indirect lights is no longer guaranteed, but in practice this does not result in visible artifacts.

As also mentioned in the previous section, we calculate the upper bound on the contribution of a cluster to some gather point in a block using the bounds proposed for Multidimensional Lightcuts. However we do not require the material term to be normalized (the contribution of the cluster to each gather



(a) Shirley6

(b) Conference

(c) Statues

Table I

THE CUT SIZE AND RENDER TIME FOR LIGHTCUTS AND COHERENT LIGHTCUTS. THE RELATIVE DIFFERENCE TO LIGHTCUTS AND RELATIVE COST OF ESTIMATING THE CONTRIBUTION, CALCULATING THE ERROR AND OTHER CALCULATIONS FOR LIGHTCUTS IS ALSO SHOWN.

Scene	Polygons	Lights	Lightcuts		Coherent Lightcuts		Difference		Relative cost		
			Cut	Time	Cut	Time	Cut	Time	Estimate	Error	Rest
Shirley6	804	1000	113	104s	119	59s	+5%	-43%	45%	38%	17%
Conference	64534	4000	102	172s	106	136s	+4%	-21%	50%	28%	22%
Conference (d)	64534	4000	102	155s	105	107s	+3%	-31%	51%	24%	25%
Conference (d,gi)	64534	31000	234	329s	234	210s	0%	-36%	53%	25%	22%
Statues	2028597	4000	118	215s	124	228s	+5%	+6%	59%	23%	18%
Statues (d)	2028597	4000	97	158s	100	132s	+3%	-16%	66%	13%	21%

point is estimated separately) and set the strength to 1. So the upper bound is:

$$U = M_u G_u I \quad (7)$$

For calculating the upper bound on glossy components in the material term, a cubemap is required. We also use a 6×6 subdivision on each face. Building such a cubemap is quite costly (i.e. 216 evaluations of the upper bound), considering building a cut requires 100–200 evaluations of the upper bound for most points on a glossy material. Although at high resolutions the number of cubemaps can easily be reduced (in our implementation of Multidimensional Lightcuts $\pm 2\%$ of the cubemaps is computed for an image of the Conference scene with 256 samples/pixel), this is not the case for the low resolution images used in the results. For this reason no cubemaps are re-used in our implementation of Coherent Lightcuts.

Because Lightcuts calculates each cut separately, memory usage is very limited. Coherent Lightcuts on the other hand requires information about an entire image block. For medium block sizes the amount of memory required is still small but rises quickly for larger block sizes. For example the required memory for blocks of 16×16 pixels is only 3.5MB but 50MB and 200MB for respectively 64×64 and 128×128 pixels. This is mostly due to the estimated contribution of each cluster in the cut that must be stored for all gather points.

V. RESULTS AND DISCUSSION

For our results 3 scenes were used: Shirley’s 6th test scene consisting of a table lit by a single area light (Shirley6) [18], Ward’s conference room lit by 24 area light sources on the ceiling (Conference) [2] and a number of statues from the Stanford 3D scanning repository [19] lit by an

environment map (Grace cathedral) of Debevec [20]. Because the Conference and Statues scenes have some glossy materials, a diffuse version is also used (indicated by “(d)”). For the Conference scene, a version with global illumination is also used (indicated by “(gi)”). All results use a maximum allowed error of 2%, a resolution of 640×480 (1 sample/pixel) and a maximum cut size of 1000. The initial block size for the Coherent Lightcuts results is 16×16 . The render times are for a single core of a Core2 Duo T7100 processor.

The results for Lightcuts and Coherent Lightcuts are shown in Table I. Comparing the cut sizes for both algorithms shows a slight increase in cut size for Coherent Lightcuts. This is caused by a slightly different order of cut refinement for some points (i.e. for some points the refined cluster is not the one with the largest error). However the render time does decrease significantly for most scenes. On the one hand, the decrease in render time is larger for the Conference scene than the Statues scene. This is due to the trivial upper bounds for the directional light sources (i.e. 1) used in the latter, as also visible in the last columns. On the other hand, glossy materials have a negative influence on the render time. This increase is smaller for the

Table II

THE PERCENTAGE OF THE TOTAL CUT THAT IS EQUAL IN LIGHTCUTS FOR BLOCKS OF 16^2 , 8^2 , 4^2 AND 2^2 PIXELS AND THE PERCENTAGE OF THE TOTAL CUT THAT IS CONSTRUCTED BEFORE SUBDIVISION OF THE BLOCKS OF THESE SIZES IN COHERENT LIGHTCUTS.

Scene	Optimal (%)				Coherent Lightcuts (%)			
	16	8	4	2	16	8	4	2
Shirley6	85	90	95	98	76	87	94	98
Conference	66	79	88	95	61	76	86	93
Conference (d)	66	79	88	95	62	77	87	94
Conference (d,gi)	62	76	87	94	33	55	75	90
Statues	55	66	76	88	40	54	65	72
Statues (d)	63	76	86	94	52	71	86	94

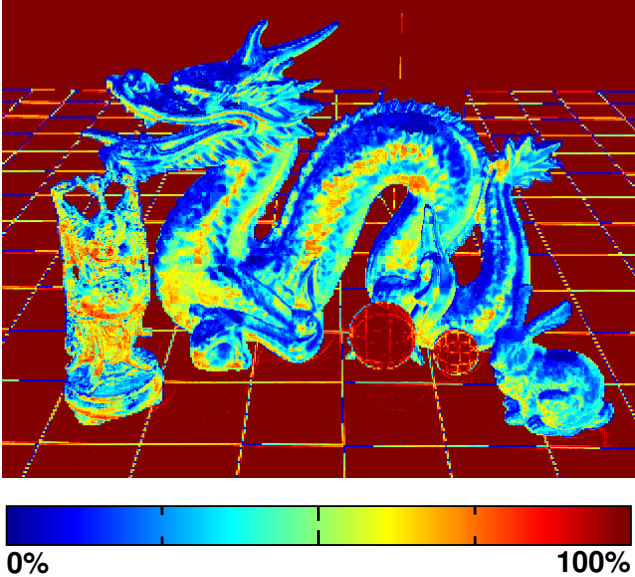


Figure 3. The percentage of the total cut that is constructed before subdivision of the 2×2 blocks in Coherent Lightcuts.

Conference scene than for the Statues scene (10% and 20% respectively) because in the first scene the diffuse component is always more important whereas in the latter some materials are mainly glossy.

The smaller advantage or disadvantage for glossy materials is caused by the usage of cubemaps and the lack of a good minimum lower bound in the coherence heuristic. The construction of the cubemaps increases the render time by $\pm 10\%$.

In Table II the available coherence between the cuts in Lightcuts and the detected coherence in Coherent Lightcuts is shown. Generally the detected coherence is close to the optimal value, which indicates the proposed coherence heuristic is good. This improves as the blocks become smaller because the minimum upper bound is estimated better due to tighter bounding boxes. The results also show less coherence is available on curved surfaces with a strong glossy component. Our algorithm also exploits this coherence less, due to the lack of a good minimum upper bound for glossy materials. This can also be seen in Fig. 3.

VI. CONCLUSION AND FUTURE WORK

We have shown that the cuts calculated by Lightcuts are very similar for neighbouring pixels and presented an algorithm to detect and exploit this coherence. We subdivide the image in a number of blocks and exploit coherence in both the entire block and its subblocks. We do not approximate the Lightcuts solution but reduce the cost of finding it (i.e. cut construction). To detect incoherence we use a heuristic based on estimating the lowest possible error.

Our algorithm reduces the render time for diffuse scenes by 15%–45%. Even though this speedup is not spectacular, we believe a much larger speed up could be achieved by exploiting the coherence between the shadow rays required for estimating the contribution of a cluster to the points in a

block. A number of techniques have already been proposed (e.g. MLRTA by Reshetov et al. [3]) which typically require a bundle of coherent rays as input. These are efficiently identified by our algorithm.

Although our coherence heuristic works very well for diffuse materials, it is not good at determining coherence on materials that are mainly glossy.

APPENDIX

CALCULATING THE MINIMUM UPPER BOUNDS

The calculation of the minimum upper bound on the cosine of the angle between the normal in a point p in a 1st bounding box \mathcal{P} and the direction to a point q in the 2nd bounding box \mathcal{Q} can be transformed to the angle w.r.t. a normal perpendicular to the z-axis. The minimal upper bound, if $\max(p_z) \leq \max(q_z)$, is:

$$\max_p(\cos \theta) \geq \frac{\min_p(\max_Q(\Delta z))}{\sqrt{D^2}} \quad (8)$$

$$D^2 = \max_p(\min_Q(\Delta x))^2 + \max_p(\min_Q(\Delta y))^2 + \min_p(\max_Q(\Delta z))^2 \quad (9)$$

The minimum upper bound and maximum lower bound on the distance along the x axis are:

$$\max_p(\min_Q(\Delta x)) \leq \max(0, \min(q_x) - \min(p_x), \max(p_x) - \max(q_x)) \quad (10)$$

$$\min_p(\max_Q(\Delta x)) \geq \max(\min(p_x) - \min(q_x), \max(q_x) - \max(p_x)) \quad (11)$$

The equations for the other axes are similar. In (11) it is assumed \mathcal{Q} is not completely inside \mathcal{P} along the x-axis.

The maximum lower bound on the distance can be calculated in a similar way:

$$\min_p(d^2) \leq \max_p(\min_Q(\Delta x))^2 + \max_p(\min_Q(\Delta y))^2 + \max_p(\min_Q(\Delta z))^2 \quad (12)$$

REFERENCES

- [1] B. Walter, S. Fernandez, A. Arbre, K. Bala, M. Donikian, and D. P. Greenberg, "Lightcuts: a scalable approach to illumination," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1098–1107, Jul. 2005.
- [2] G. Ward, "Mgf example scenes," <http://radsite.lbl.gov/mgf/scenes.html>.
- [3] A. Reshetov, A. Soupikov, and J. Hurley, "Multi-level ray tracing algorithm," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*. New York, NY, USA: ACM, 2005, pp. 1176–1185.
- [4] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Trans. Comput.*, vol. 20, no. 6, pp. 623–629, 1971.
- [5] F. H. Pighin, D. Lischinski, and D. Salesin, "Progressive previewing of ray-traced images using image plane discontinuity meshing," in *Proceedings of the Eurographics Workshop on Rendering Techniques '97*. London, UK: Springer-Verlag, 1997, pp. 115–125.
- [6] B. Guo, "Progressive radiance evaluation using directional coherence maps," in *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1998, pp. 255–266.
- [7] K. Bala, B. Walter, and D. P. Greenberg, "Combining edges and points for interactive high-quality rendering," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 631–640, 2003.
- [8] M. R. Bolin and G. W. Meyer, "A perceptually based adaptive sampling algorithm," in *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1998, pp. 299–309.

- [9] J. P. Farrugia and B. Peroche, "A progressive rendering algorithm using an adaptive perceptually based image metric," *Computer Graphics Forum (Eurographics 2004 Proceedings)*, vol. 23, no. 3, Sep. 2004.
- [10] G. Ward and P. Heckbert, "Irradiance gradients," in *Eurographics Workshop on Rendering*, 1992.
- [11] J. Křivánek, P. Gautron, K. Bouatouch, and S. Pattanaik, "Improved radiance gradient computation," in *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*. New York, NY, USA: ACM, 2005, pp. 155–159.
- [12] I. Wald, C. Benthin, and P. Slusallek, "Interactive global illumination using fast ray tracing," in *Rendering Techniques 2002 (Proceedings of the Thirteenth Eurographics Workshop on Rendering)*, Jun. 2002.
- [13] B. Walter, A. Arbre, K. Bala, and D. P. Greenberg, "Multidimensional lightcuts," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*. New York, NY, USA: ACM, 2006, pp. 1081–1088.
- [14] M. Hašan, F. Pellacini, and K. Bala, "Matrix row-column sampling for the many-light problem," in *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*. New York, NY, USA: ACM, 2007, p. 26.
- [15] K. E. Torrance and E. M. Sparrow, "Theory for off-specular reflection from roughened surfaces," *Journal of Optical Society of America*, vol. 57, no. 9, 1967.
- [16] J. F. Blinn, "Models of light reflection for computer synthesized pictures," in *SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1977, pp. 192–198.
- [17] B. Walter, "Notes on the ward BRDF," Program of Computer Graphics, Cornell University, Technical report PCG-05-06, Apr. 2005.
- [18] P. Shirley, "Tu computer science department," <http://ftp.cs.indiana.edu/pub/RW5>.
- [19] Stanford University Computer Graphics Laboratory, "The stanford 3d scanning repository," 2007, <http://graphics.stanford.edu/data/3Dscanrep/>.
- [20] P. E. Debevec, "Light probe image gallery," 2004, <http://www.debevec.org/probes/>.