Eurographics Symposium on Rendering 2011
Ravi Ramamoorthi and Erik Reinhard
(Guest Editors)

Volume 30 (2011), Number 4

Least Squares Vertex Baking

L. Kavan[†1], A. W. Bargteil[2] and P.-P. Sloan[1]

[1]Disney Interactive Studios
[2]University of Utah

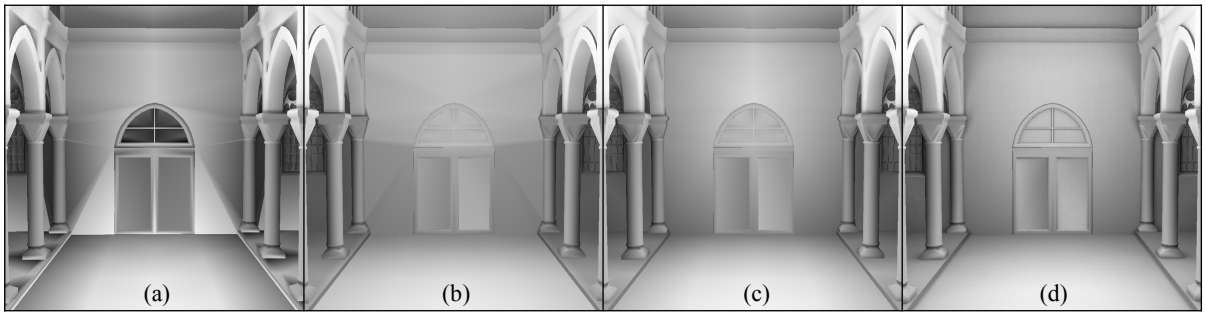

Figure 1: *Ambient occlusion in the Sibenik cathedral model represented at vertices using (a) point sampling, (b) averaging triangle samples and (c) our method. Compare to (d) ground truth per-pixel rendering.*

Abstract

We investigate the representation of signals defined on triangle meshes using linearly interpolated vertex attributes. Compared to texture mapping, storing data only at vertices yields significantly lower memory overhead and less expensive runtime reconstruction. However, standard approaches to determine vertex values such as point sampling or averaging triangle samples lead to suboptimal approximations. We discuss how an optimal solution can be efficiently calculated using continuous least-squares. In addition, we propose a regularization term that allows us to minimize gradient discontinuities and mach banding artifacts while staying close to the optimum. Our method has been integrated in a game production lighting tool and we present examples of representing signals such as ambient occlusion and precomputed radiance transfer in real game scenes, where vertex baking was used to free up resources for other game components.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Representation of signals on triangle meshes is a classic problem in computer graphics. The standard solution relies on surface parametrization and texture mapping. While this is an efficient representation for high-frequency signals, the associated memory footprints are often one of the limiting factors in game development. Especially for low-frequency signals it is common to use *vertex baking*, i.e., representation by linearly interpolated vertex attributes [Gou71]. This is much less expensive, especially for scalar signals such as ambient occlusion [ZIK98, Lan02]—with textures, the uv-coordinates require two scalars per vertex, in addition to the image data. Furthermore, no parametrization or runtime texture memory access is needed. The significantly reduced memory footprints and lower shader complexity are important not only in low-end gaming platforms and mobile devices, but also in high-end games that often combine texture mapping and vertex baking [LH09, Wan11].

The question addressed in this paper is how to determine the vertex values to optimally approximate a given signal f defined on a triangle mesh. The most straightforward solution is to point sample f at vertex locations. While under certain

† ladislav.kavan@gmail.com

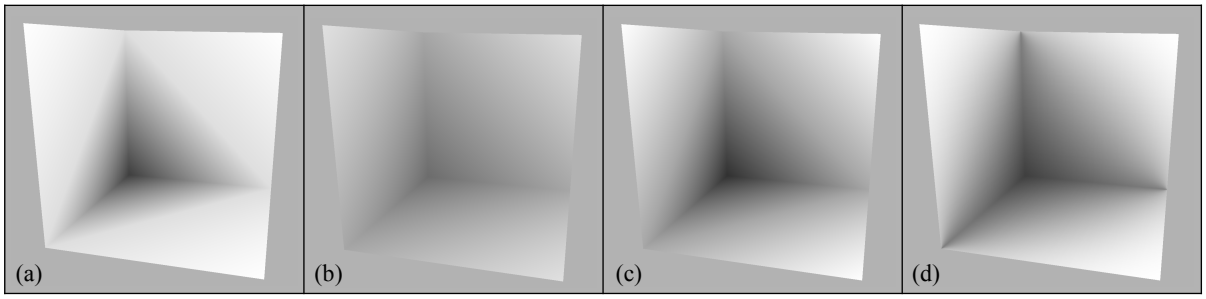

Figure 2: *Interpolated vertex samples with values determined by (a) point sampling, (b) averaging triangle samples and (c) our method. Compare with (d) ground truth per-pixel rendering.*

conditions this is sufficient for exact reconstruction [GW02], in practice we almost always must accept some inaccuracies. Error is introduced because the vertex density is often irregular, the input signal might not be band limited, and the graphics hardware supports only linear reconstruction. Unfortunately, point sampling does not minimize the approximation error and aliasing may occur, see Figure 2a. A more accurate approach is to densely sample f inside triangles and calculate the vertex values by taking barycentric-weighted averages [KL06]. This is analogous to low-pass filtering which reduces aliasing, but at the cost of blurring the result, see Figure 2b.

One possibility to improve reconstruction accuracy is to refine and/or optimize the mesh. However, game levels often push the limits of the geometry budgets and the artists are not enthusiastic to compromise on scene size or geometric detail. Moreover, geometry refinement would involve unnecessary replication of all other vertex attributes (vertices, tangent frames, texture coordinates, etc.) and would impact the whole game development pipeline, because geometry also serves purposes other than rendering, such as gameplay and physics.

In this paper, we assume a fixed mesh and we formulate and solve an optimization problem that minimizes the error in a least squares sense, see Figure 2c. However, with more complex geometry, the least squares solution is not always as visually pleasing because high variations in vertex values can be introduced, see Figure 3a. To remedy this, we study regularization techniques that trade off accuracy for smoothness, allowing us to minimize the mach band artifacts. Our experiments indicate that a small amount of regularization improves the visual quality considerably while still staying close to the optimum.

Our techniques have been implemented in a game production lighting tool and we provide examples of vertex baked ambient occlusion and precomputed radiance transfer in real game scenes, demonstrating that our method provides smoother and more accurate approximations than previous methods. From a practical standpoint, vertex baking resulted in significant memory savings compared to textures, freeing up resources for other components of the game while requiring only minimal changes in the development workflows and the game runtime.

2. Related Work

Vertex baking is common on low-end gaming platforms like the Nintendo Wii, cell phones and tablets, where memory and power consumption considerations require that compromises be made. On high-end platforms, texture memory consumption and shader complexity are still issues and the savings resulting from vertex baking can significantly improve the quality of the final products [LH09, Wan11]. Another problem is that texture mapping requires parametrization, often carefully authored to avoid undesirable uv-seams. In recent years, texturing techniques that do not require an explicit parametrization have been proposed [BL08, YKH10], but efficient implementations to date have only been presented on high-end graphics hardware.

The problem of representing signals on triangle meshes occurs in many areas of computer graphics. Much of the early work in Radiosity [CW93] deals with representing lighting signals by point sampling. More accurate results can be achieved by adapting the mesh to conform to the signal [HSA91] or by using Galerkin methods [Zat93]. Discontinuity meshing [LTG92] refines the mesh at shadow boundaries to more accurately represent the signal before or during the solution. While these methods are important, they are less useful in games, where geometry refinement or higher-order reconstruction functions are typically not an option due to resource limitations and hardware capabilities.

A related approach studied in the literature is to sample a fine mesh and then simplify it to better represent a given signal [SWH*95, Hop96]. However, the final signal used in these techniques is computed using a greedy optimization process which does not come with any optimality guarantees. In this paper, we leverage advances in linear system solvers over the past decades [BBK05] to efficiently compute optimal solutions for a fixed mesh, resulting in higher quality.

Precomputed radiance transfer [SKS02] is another technique that usually stores data at the vertices of a mesh. In the literature, these meshes are typically finely tessellated and some work has been done toward computing the solution more efficiently [HR10] or adapting the mesh [KP04]. Precomputed radiance transfer, especially if only used for indirect lighting, is another example of a signal that can be efficiently represented using our technique.

Probably the closest work to ours [KL06] reduces aliasing artifacts when storing volumetric irradiance by performing the volumetric equivalent of the barycentric-weighted averages described in this paper. Note that in spite of the aliasing issues, point sampling at vertex locations is still one of the most common solutions, and in fact is the default in many applications, including Autodesk Maya. The artifacts are either resolved during manual post-processing or are masked by another layer of high-frequency textures. Our method results in more visually pleasing approximations, eliminating the need for any post-processing or masking.

3. Vertex Baking

As input, we have a triangular mesh with vertices $\mathbf{v}_1,\ldots,\mathbf{v}_N$ representing a piecewise linear surface $\mathcal{S}\subset\mathbb{R}^3$. We assume the vertices at creases have been duplicated, i.e., each connected component of the mesh corresponds to a smooth patch (smoothing group). We further assume there is an integrable function $f:\mathcal{S}\to\mathbb{R}$ that assigns each surface point a scalar value (vector functions are handled per-component). The task is to find vertex values $\mathbf{x}=(x_1,\ldots,x_N)^T\in\mathbb{R}^{N\times 1}$ such that their piecewise linear interpolant $g_{\mathbf{x}}:\mathcal{S}\to\mathbb{R}$ is as close as possible to f in the least squares sense, i.e., minimizes

$$E(\mathbf{x})=\int_{\mathcal{S}}\left(f(\mathbf{p})-g_{\mathbf{x}}(\mathbf{p})\right)^2 d\mathbf{p} \tag{1}$$

where $\mathbf{p}$ is an arbitrary surface point. The function $g_{\mathbf{x}}$ can be formally expressed as

$$g_{\mathbf{x}}(\mathbf{p})=\sum_{i=1}^{N} h_i(\mathbf{p})x_i \tag{2}$$

where $h_i:\mathcal{S}\to\mathbb{R}$ are piecewise linear hat functions (nodal shape functions from linear finite elements) such that $h_i(\mathbf{v}_i)=1$ and $h_i(\mathbf{v}_j)=0$ for $i\neq j$. In the following we drop the subscript from $g_{\mathbf{x}}$ for improved readability.

3.1. Optimal Fit

Equation (1) represents a continuous linear least squares problem. To solve it, we substitute Equation (2) into Equation (1) and re-arrange the terms, obtaining

$$E(\mathbf{x})=\mathbf{x}^T\mathbf{A}\mathbf{x}-2\mathbf{x}^T\mathbf{b}+c \tag{3}$$

where $\mathbf{A}\in\mathbb{R}^{N\times N}$, $\mathbf{b}\in\mathbb{R}^{N\times 1}$ and $c\in\mathbb{R}$ are defined as follows

$$A_{i,j}=\int_{\mathcal{S}} h_i(\mathbf{p})h_j(\mathbf{p})d\mathbf{p} \tag{4}$$

$$b_i=\int_{\mathcal{S}} h_i(\mathbf{p})f(\mathbf{p})d\mathbf{p} \tag{5}$$

$$c=\int_{\mathcal{S}} f^2(\mathbf{p})d\mathbf{p} \tag{6}$$

The matrix $\mathbf{A}$ is sparse, symmetric positive definite and is also known as the *mass* or *Gram* matrix. Its entries can be calculated analytically: $A_{i,i}$ is the total area of the triangles incident to vertex i divided by 6 and $A_{i,j}$ is the total area of the triangles incident to edge ij divided by 12 (zeros elsewhere). The vector $\mathbf{b}$ and scalar c typically cannot be calculated analytically, because we usually do not have a closed-form expression for f and, therefore, we apply Monte Carlo integration, see Section 3.3.

Because $\mathbf{A}$ is positive definite, $E(\mathbf{x})$ is convex and we can find its global minimum by solving $\partial E(\mathbf{x})/\partial\mathbf{x}_i=0$ for $i=1,\ldots,N$, which reduces to

$$\mathbf{A}\mathbf{x}=\mathbf{b} \tag{7}$$

A common approximation is to replace $\mathbf{A}$ with a diagonal matrix $\mathbf{D}$ such that $D_{i,i}=\sum_{j=1}^{N}A_{i,j}=\int_{\mathcal{S}}h_i(\mathbf{p})d\mathbf{p}$, a technique known as *mass-lumping*. Note that the mass lumped solution $\tilde{\mathbf{x}}=\mathbf{D}^{-1}\mathbf{b}$ has components

$$\tilde{x}_i=\frac{\int_{\mathcal{S}}h_i(\mathbf{p})f(\mathbf{p})d\mathbf{p}}{\int_{\mathcal{S}}h_i(\mathbf{p})d\mathbf{p}} \tag{8}$$

which correspond to a weighted average (with h_i being the weighting function [KL06]). However, better results can be achieved by solving Equation (7) exactly, see Figure 2b,c. We discuss the numerical solution methods in Section 3.3.

3.2. Regularization

While the least squares formulation from Equation (1) works very well for simple scenes as in Figure 2, in more complex scenes the objective $E(\mathbf{x})$ becomes overly eager to exploit all available degrees of freedom. For example, Figure 3a shows the exact solution of Equation (7) applied to a game scene. The piece-wise linear approximation g exhibits large gradient discontinuities, visually exacerbated by the mach band effect.

To obtain a better behaved g, we augment the linear system from Equation (7) with a smoothness (or regularization) term $\mathbf{R}\in\mathbb{R}^{N\times N}$, resulting in

$$(\mathbf{A}+\alpha\mathbf{R})\mathbf{x}=\mathbf{b} \tag{9}$$

where $\alpha\geq 0$ is a user-tunable parameter. The simplest choice, inspired by Tikhonov regularization, i.e., $\mathbf{R}:=\mathbf{A}$, is inadequate because it corresponds to reducing brightness by pushing $\mathbf{x}$ to zero. A more suitable $\mathbf{R}$ should not affect the absolute values of $\mathbf{x}$. This motivates gradient-based regularization,

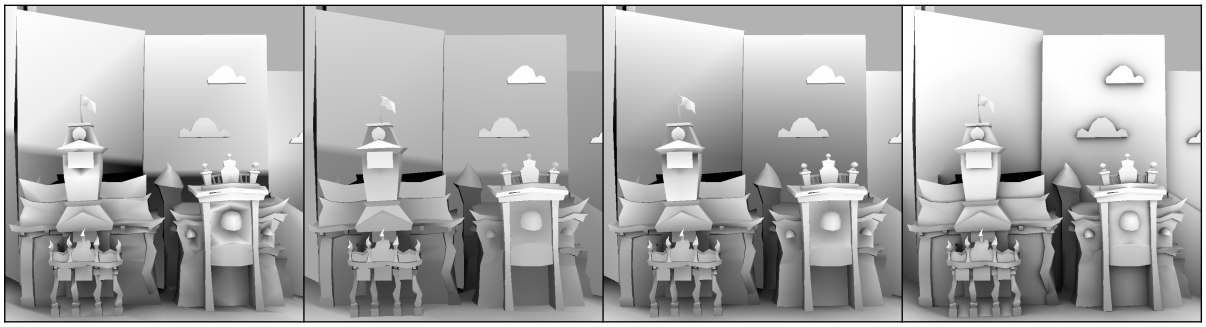

Figure 3: *Left to right: (a) least squares fit with no regularization, (b) gradient-based regularization ($\alpha = 0.1$), (c) edge-based regularization ($\alpha = 0.1$) and (d) ground truth.*

which can be expressed as

$$\underbrace{\int_{\mathcal{S}} (f(\mathbf{p}) - g(\mathbf{p}))^2 \, d\mathbf{p}}_{\text{data term}} + \alpha \underbrace{\int_{\mathcal{S}} \|\nabla g(\mathbf{p})\|^2 \, d\mathbf{p}}_{\text{smoothness term}} \qquad (10)$$

To obtain the regularization matrix, we note that ∇g is piecewise constant which allows us to split the smoothness term per triangle. For triangle t with vertices $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k$,

$$\nabla g_{|t} = (x_j - x_i) \frac{(\mathbf{v}_i - \mathbf{v}_k)^{\perp}}{2a_t} + (x_k - x_i) \frac{(\mathbf{v}_j - \mathbf{v}_i)^{\perp}}{2a_t} \qquad (11)$$

where $a_t \in \mathbb{R}$ is the area of triangle t and $\perp$ denotes a counterclockwise rotation by 90° in the plane of the triangle [BKP*10]. One subtle issue is that the units of $\nabla g_{|t}$ do not match the units of $f(\mathbf{p}) - g(\mathbf{p})$ (the former is inversely proportional to the units of distance, while the latter is independent of them). This can be fixed by scaling the gradient by the square root of the triangle area, i.e., introducing $\tilde{\nabla} g_{|t} := \sqrt{a_t} \nabla g_{|t}$. Employing $\tilde{\nabla} g_{|t}$ instead of $\nabla g_{|t}$ makes the units of the data and smoothness terms match and corresponds to giving more weight to the gradients of larger triangles. Because the gradient is linear in $\mathbf{x}$, we can write $\tilde{\nabla} g_{|t} = \mathbf{F}_t \mathbf{x}$ where $\mathbf{F}_t$ is a $\mathbb{R}^{3 \times N}$ sparse matrix. This allows us to express the regularization term as

$$\int_{\mathcal{S}} \|\tilde{\nabla} g(\mathbf{p})\|^2 d\mathbf{p} = \sum_t a_t \|\tilde{\nabla} g_{|t}\|^2 = \sum_t a_t \mathbf{x}^T \mathbf{F}_t^T \mathbf{F}_t \mathbf{x}$$
$$= \mathbf{x}^T \mathbf{R}_{grad} \mathbf{x} \qquad (12)$$

where $\mathbf{R}_{grad} \in \mathbb{R}^{N \times N}$ is symmetric positive definite. It can be shown $\mathbf{R}_{grad}$ is closely related to the standard discretization of the Laplace-Beltrami operator, i.e., the familiar cotangent formula [LZ10].

Employing the regularization term $\mathbf{R}_{grad}$ leads to a smoother g, see Figure 3b. While $\mathbf{R}_{grad}$ preserves brightness, the side effect is a loss of contrast, because as $\alpha \to \infty$, the function g converges to the average of all samples on each connected component of the mesh. This is obviously suboptimal—for example, there is no reason to reduce the gradient of an isolated triangle. Therefore, we propose a better regularization term that explicitly penalizes large gradient discontinuities across internal mesh edges. Specifically, for each pair of triangles t and u that share an internal (i.e., non-crease and non-boundary) edge, we sum

$$\sum_{(t,u)} (a_t + a_u) \|\tilde{\nabla} g_{|t} - \tilde{\nabla} g_{|u}\|^2 =$$
$$\sum_{(t,u)} (a_t + a_u) \mathbf{x}^T (\mathbf{F}_t - \mathbf{F}_u)^T (\mathbf{F}_t - \mathbf{F}_u) \mathbf{x} = \mathbf{x}^T \mathbf{R}_{edge} \mathbf{x} \qquad (13)$$

where $\mathbf{R}_{edge} \in \mathbb{R}^{N \times N}$ is a symmetric positive definite *edge-based* regularization term. Employing this new regularization term instead of $\mathbf{R}_{grad}$ results in solutions that better preserve contrast; for example, notice the effect on the background wall in Figure 3. This is because the matrix $\mathbf{R}_{edge}$ encourages smoother gradients over connected parts of the mesh without penalizing gradient magnitudes at boundaries and creases.

3.3. Numerical Solution

So far we have described how to calculate the system matrix. It remains to evaluate the signal-dependent right hand side $\mathbf{b}$ from Equation (5). Because an explicit formula for f is typically unavailable, we use Monte Carlo integration. Specifically, given approximately uniformly distributed samples $\mathbf{p}_1, \ldots, \mathbf{p}_M \in \mathcal{S}$, we estimate b_i as

$$b_i \approx \frac{\mu_i}{|X_i|} \sum_{j \in X_i} h_i(\mathbf{p}_j) f(\mathbf{p}_j) \qquad (14)$$

where X_i is the set of samples in the triangles incident to vertex i and μ_i is the sum of the areas of the triangles incident to i. For ambient occlusion, $f(\mathbf{p}_j)$ is calculated by importance sampling the hemisphere of rays starting at $\mathbf{p}_j$ and ray casting against the scene geometry using 256 directions. For precomputed radiance transfer, this process is modified to account for spherical harmonics basis functions and 1-bounce indirect lighting [SKS02]. Monte-Carlo integration can also be used to estimate the coefficient c from Equation (6), but this is only necessary if the actual numerical value of the error term $E(\mathbf{x})$ is needed.

The system matrix $\mathbf{A} + \alpha \mathbf{R}$ for $\alpha \geq 0$ is sparse, symmetric

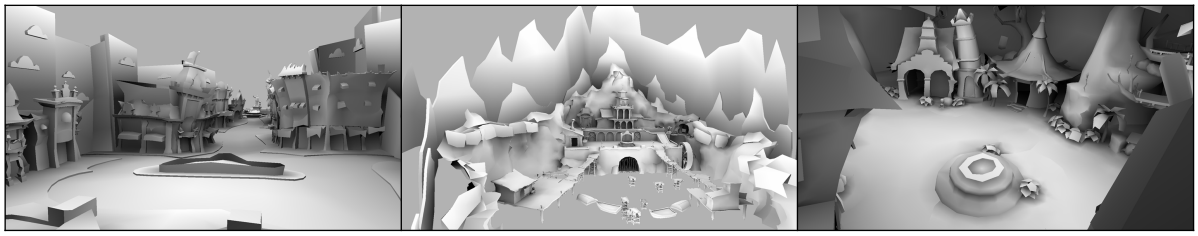

Figure 4: *Three levels from the videogame we used for testing (vertex-baked ambient occlusion computed using our method).*

Scene	Vertices	Triangles	Samples	E_{point}	E_{avg}	E_{our}, α		E_{lsqr}	Raycast	Solve time/mem	
Half Box	12	6	10^4	2.35	0.928	0.159	0	0.159	9ms	<1ms	26kB
Sibenik	71654	75987	10^6	2.1	0.203	0.136	0.05	0.103	16s	0.4s	37MB
Level A	52274	48327	10^6	4.13	0.871	0.79	0.1	0.643	15s	0.3s	26MB
Level B	66836	65042	10^6	1.37	1.06	0.567	0.1	0.47	17s	0.3s	33MB
Level C	38379	47097	10^6	3.23	0.506	0.466	0.1	0.329	14s	0.3s	20MB

Table 1: *Results for vertex baked ambient occlusion in our test scenes using point sampling (E_{point}), weighted averaging (E_{avg}) and our method (E_{our}, edge-based regularization with parameter α). E_{lsqr} is the error for $\alpha = 0$, i.e., the lowest possible.* Raycast *is the total time for calculating ambient occlusion at all samples,* Solve *is the time and memory for solving the linear system.*

positive definite and depends only on the mesh geometry. Typically, this class of linear systems can be most efficiently solved using direct sparse solvers [BBK05]. In our implementation, we used PARDISO [SG06]. Even though the numerical factorization needs to be recomputed for different values of α, the system matrix has simple structure and permits solves fast enough for interactive feedback, facilitating parameter tuning. For ambient occlusion, one complication is that while the input values are within $[0,1]$, the resulting vertex values x_i can under/overshoot (an effect similar to the Gibbs phenomena). An accurate constrained solution could be obtained with quadratic programming, but we found simple clamping to $[0,1]$ is sufficient for our purposes. In our experiments, only about 5% or fewer of the values x_i are outside $[0,1]$ and often violate the bounds only slightly. Alternatively, we could also store arbitrary values of x_i and perform the clamping after interpolation (i.e., in the pixel shader).

4. Results and Discussion

Our main practical motivation was to efficiently represent ambient occlusion. Aside from the Sibenik cathedral in Figure 1 and the didactic example in Figure 2, we tested our algorithm on three levels from a shipped game, see Figure 4 and Figure 9. To facilitate comparisons, the images are also available as supplementary materials. Note that the game meshes have occasional artifacts such as unintended penetrations or inconsistent winding orders which show up in the result but do not cause our algorithm to malfunction. We compare the results of our method to point sampling, weighted averaging, and ground truth (raytraced using 1024 rays per pixel). Point sampling requires fewer samples, but produces noticeable aliasing artifacts. Weighted averaging uses more samples and produces higher quality results, but mesh dependent mach band artifacts are apparent. Moreover, in large coarse triangles some contrast is lost (see Figure 2b) due to excessive blurring. Our method employing edge-based regularization with a small α uses the same samples as weighted averaging but produces smoother results, better preserves contrast and results in lower approximation error.

We report the quantitative results in Table 1. Note that the reported vertex counts include duplicates created on creases (i.e., vertices that have the same spatial location but different sample values). The error is calculated according to Equation (1), divided by the total mesh area and multiplied by 100. The times were measured on a 6 core dual 2.9GHz Intel X5670 CPU with 24GB memory. The number of samples is a user specified parameter; in our scenes we used the rather conservative number of 10^6 samples. Increasing the sample count further has only negligible impact on the final vertex signal (see Figure 5). While fewer than 10^6 samples would probably also suffice, the raytracer performance makes this question less practically important. The additional time and memory complexity of the linear solve required by our method is negligible, see Table 1.

A useful feature of our technique is that the amount of smoothing is controllable by the parameter α. For lower values of α, we obtain sharp gradients aggressively approximating the input signal, while higher values result in a softer look with more prominent creases, see Figure 10. For even more flexible art control, we could also consider combining the gradient- and edge-based regularization terms.

While our main motivation was efficient representation of

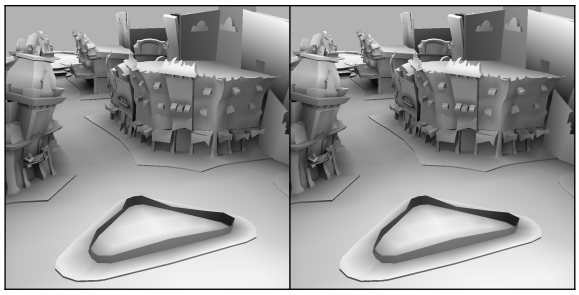

Figure 5: *Vertex signal calculated with (left)* 10^6 *samples is visually indistinguishable from using (right)* 10^8 *samples (Level A scene,* $\alpha = 0.1$ *edge-based regularization).*

ambient occlusion, our method can be successfully applied to other input signals, including vector ones. For example, we consider precomputed radiance transfer [SKS02] represented with 6th order spherical harmonics. Each of the 36 dimensions is vertex-baked separately using our method. In Figure 6, we compare the results of weighted averaging to our method ($\alpha = 0.1$ edge-based regularization). Similarly to ambient occlusion, our method produces a smoother and more accurate approximation resulting in more visually pleasing dynamic relighting.

Our technique outperforms previous methods especially with coarse and irregularly tesselated meshes, but more subtle differences can be visible even with finer geometry. We illustrate this on a simple uniformly tessellated L-shaped mesh

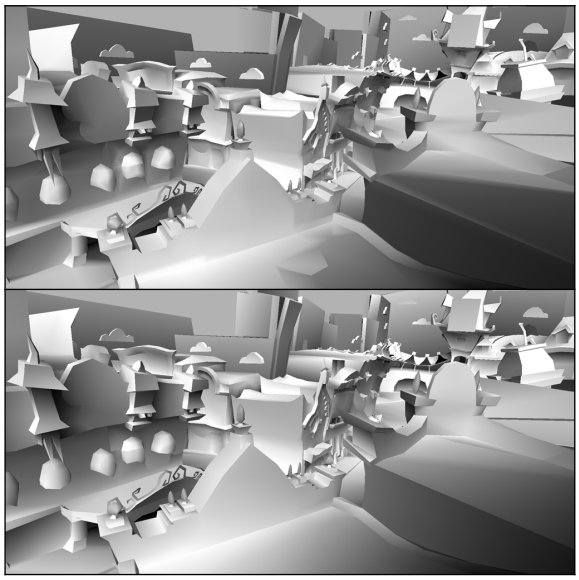

Figure 6: *Dynamic relighting with 36-dimensional precomputed radiance transfer with vertex values calculated by (top) weighted averaging and (bottom) our method.*

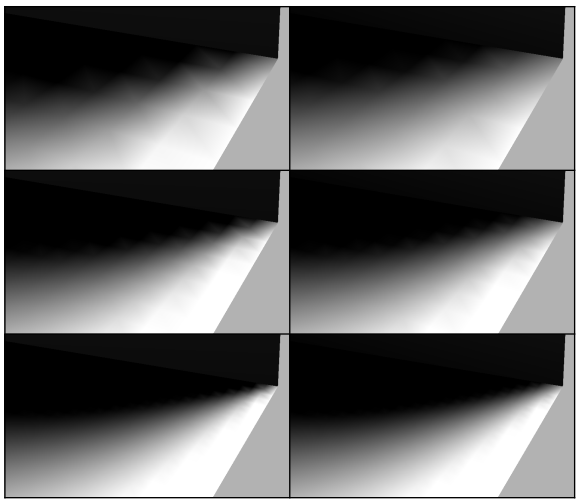

Figure 7: *Precomputed radiance transfer shadow at various mesh resolutions (top to bottom): 16×16, 32×32 and 64×64; (left) weighted averaging, (right) our method.*

with varying resolution, see Figure 7. While the deficiencies of weighted averaging begin to disappear with increased resolution, the costs of additional geometry can be prohibitive in large scenes.

While vertex baking is best suited for low-frequency lighting signals, our method is general and applicable even in the presence of sharp edges and noise. To illustrate this, we execute progressive meshes to represent a 298×298 image using a 2000 triangle mesh with a vertex baked RGB signal [Hop96]. Because progressive meshes optimize vertex attributes in a greedy way without any optimality guarantees, our method results in a more accurate approximation for the same coarse mesh, see Figure 8.

5. Conclusion and Future Work

We present a mathematical framework for vertex-level representation of signals on triangle meshes, explicitly exposing the trade-offs between accuracy and smoothness. Compared to previous methods (point sampling and weighted averaging), our technique generates more accurate results, helps to eliminate the mach band artifacts and better preserves contrast of the input signal. Compared to using textures, vertex baking requires less storage and memory bandwidth, resulting in significant savings for real-time applications, such as games. Our method is best suited to low-frequency lighting signals such as ambient occlusion or precomputed radiance transfer. While our approach is applicable to any signal, for higher frequencies and/or repeated patterns, textures are often a better choice. In future work, it would be interesting to couple our technique with mesh optimization [HDD*93] to further improve reconstruction accuracy, consider non-linear regularization terms [ROF92] and combine our method with

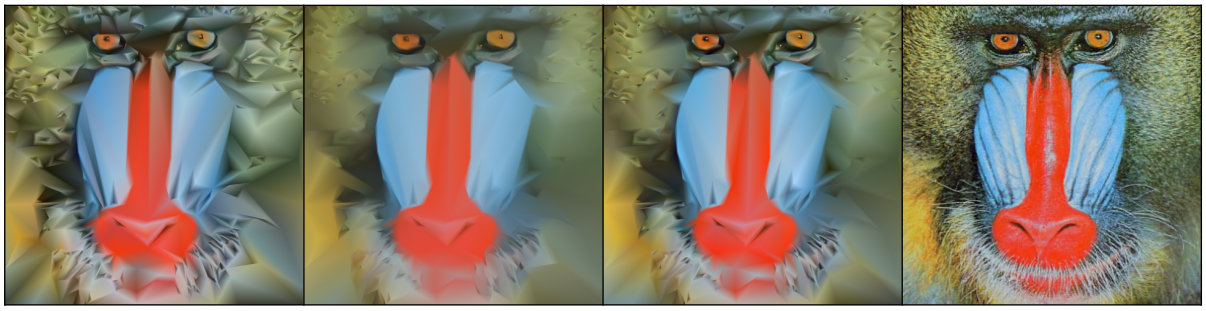

Figure 8: *An image represented by a vertex baked RGB signal on a coarse mesh (left to right): (a) the original progressive meshes result, (b) weighted averaging and (c) our method. Compare to (d) the original image.*

emerging texturing schemes [YKH10]. An extension of our method to higher order basis functions [SSD03] could be derived by replacing our piecewise linear hat functions h_i (Section 3) with higher order ones. The increased cost of runtime interpolation would be balanced by higher approximation quality for the same number of degrees of freedom, especially with strictly C^1-conforming schemes [CO01].

Acknowledgements. We thank the anonymous reviewers for their feedback and helpful comments. Many thanks to Jeff Grills, Robert Kovach and Tony Bratton for help with integration in production tools and Peter Shirley for careful proofreading.

References

[BBK05] BOTSCH M., BOMMES D., KOBBELT L.: Efficient linear system solvers for mesh processing. In *Mathematics of Surfaces XI*, Martin R., Bez H., Sabin M., (Eds.), vol. 3604 of *Lecture Notes in Computer Science*. 2005, pp. 62–83. 2, 5

[BKP*10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., LEVY B.: *Polygon Mesh Processing*. AK Peters, 2010. 4

[BL08] BURLEY B., LACEWELL D.: Ptex: Per-face texture mapping for production rendering. In *Eurographics Symposium on Rendering 2008* (2008), pp. 1155–1164. 2

[CO01] CIRAK F., ORTIZ M.: Fully C^1-conforming subdivision elements for finite deformation thin-shell analysis. *Journal for Numerical Methods in Engineering 51* (2001), 813–833. 7

[CW93] COHEN M. F., WALLACE J. R.: *Radiosity and Realistic Image Synthesis*. Academic Press Professional, San Diego, CA, 1993. 2

[Gou71] GOURAUD H.: Continuous shading of curved surfaces. *IEEE Trans. Comput. 20* (June 1971), 623–629. 1

[GW02] GONZALEZ R. C., WOODS R. E.: *Digital Image Processing*. Prentice Hall, 2002. 2

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. *Proceedings of SIGGRAPH '93* (1993), 19–26. 6

[Hop96] HOPPE H.: Progressive meshes. *Proceedings of SIGGRAPH '96* (1996), 99–108. 2, 6

[HR10] HUANG F.-C., RAMAMOORTHI R.: Sparsely Precomputing the Light Transport Matrix for Real-Time Rendering. *Computer Graphics Forum (Proc. EGSR'10) 29*, 2 (8 2010), 1335–1345. 3

[HSA91] HANRAHAN P., SALZMAN D., AUPPERLE L.: A Rapid Hierarchical Radiosity Algorithm. In *Computer Graphics (Proceedings of SIGGRAPH '91)* (1991), vol. 25, pp. 197–206. 2

[KL06] KONTKANEN J., LAINE S.: Sampling precomputed volumetric lighting. *J. Graphics Tools 11*, 3 (2006), 1–16. 2, 3

[KP04] KRIVANEK J., PATTANAIK S.: Adaptive mesh subdivsion for precomputed radiance transfer. In *Proceedings of the Twentieth Spring Conference on Computer Graphics (SCCG 2004)* (2004), pp. 106–111. 3

[Lan02] LANDIS H.: Global illumination in production. ACM SIGGRAPH 2002 Course #16 Notes, July 2002. 1

[LH09] LARSSON D., HALEN H.: The unique lighting of Mirror's Edge. In *Game Developers Conference* (2009). 1, 2

[LTG92] LISCHINSKI D., TAMPIERI F., GREENBERG D. P.: Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications 12*, 6 (Nov. 1992), 25–39. 2

[LZ10] LÉVY B., ZHANG R. H.: Spectral geometry processing. In *ACM SIGGRAPH Course Notes* (2010). 4

[ROF92] RUDIN L. I., OSHER S., FATEMI E.: Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena 60*, 1-4 (1992), 259 – 268. 6

[SG06] SCHENK O., GARTNER K.: On fast factorization pivoting methods for symmetric indefinite systems. *Elec. Trans. Numer. Anal. 23* (2006), 58–179. 5

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.* (2002). 3, 4, 6

[SSD03] SOLIN P., SEGETH K., DOLEZEL I.: *Higher-Order Finite Element Methods*. Chapman and Hall/CRC, 2003. 7

[SWH*95] SHIRLEY P., WADE B., HUBBARD P., ZARESKI D., WALTER B., GREENBERG D. P.: Global illumination via density estimation. In *Eurographics Rendering Workshop 1995* (June 1995), Eurographics. 2

[Wan11] WANG X.: Automated level of detail generation for HALO: REACH. In *Game Developers Conference* (2011). 1, 2

[YKH10] YUKSEL C., KEYSER J., HOUSE D. H.: Mesh colors. *ACM Trans. Graph. 29*, 2 (2010), 1–11. 2, 7

[Zat93] ZATZ H. R.: Galerkin radiosity: A higher order solution method for global illumination. In *SIGGRAPH 93 Conference Proceedings* (1993), pp. 213–220. 2

[ZIK98] ZHUKOV S., INOES A., KRONIN G.: An ambient light illumination model. In *Rendering Techniques '98* (1998), pp. 45–56. 1

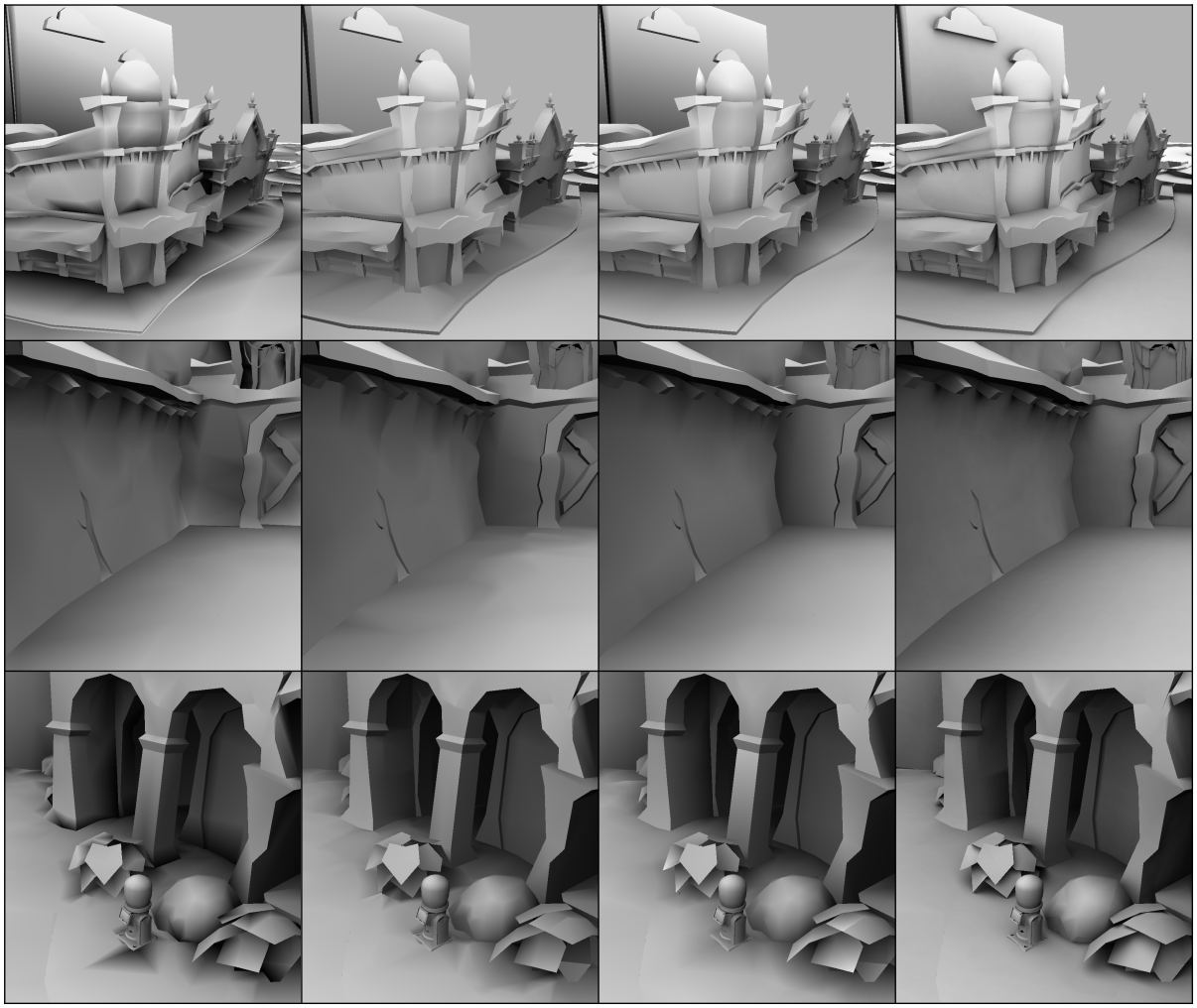

Figure 9: *Three levels from our game with vertex ambient occlusion values calculated by (left to right): (a) point sampling, (b) averaging triangle samples, (c) our method and (d) ground truth per-pixel rendering.*

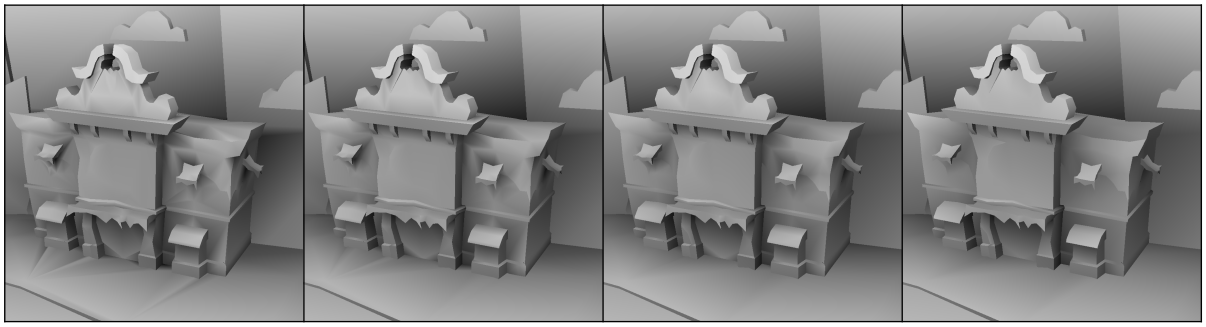

Figure 10: *Varying the regularization parameter (left to right): (a)* $\alpha = 0$ *(no regularization), (b)* $\alpha = 0.01$*, (c)* $\alpha = 0.1$ *(recommended), (d)* $\alpha = 1$.