Université de Montréal

Extended Distribution Effects for Realistic Appearance and Light Transport

par

Jean-Philippe Guertin-Renaud

Département d'informatique et de recherche opérationnelle Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.) en informatique

Orientation infographie

5 mai 2021

[©] Jean-Philippe Guertin-Renaud, 2021

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Extended Distribution Effects for Realistic Appearance and Light Transport

présentée par

Jean-Philippe Guertin-Renaud

a été évaluée par un jury composé des personnes suivantes :

Pierre Poulin (président-rapporteur)

Derek Nowrouzezahrai

(directeur de recherche)

Bernhard Thomaszewski

(membre du jury)

Jacob Munkberg

(examinateur externe)

Lyne Da Sylva

(représentant du doyen de la FESP)

Résumé

L'imagerie moderne générée par ordinateur cherche constamment à être de plus en plus représentative de la réalité physique tout autour de nous, et un de ces phénomènes clés est la notion d'effets de distribution. Les effets de distribution sont une catégorie de comportements du transport de la lumière caractérisés par leur nature distribuée selon une ou plusieurs dimension(s) donnée(s). Par exemple, le flou de mouvement est un effet de distribution dans le temps, alors que la profondeur de champ introduit le diaphragme de la caméra, ajoutant ainsi deux dimensions. Ces effets sont communs dans les films et la réalité, les rendant donc désirables à reproduire.

Dans cette thèse par articles, nous présentons quatre articles qui utilisent, étendent ou s'inspirent des effets de distribution. Premièrement, nous proposons une technique novatrice pour faire le rendu de flou de mouvement non-linéaire pour des applications en temps réel tout en conservant des caractéristiques clés d'efficacité et de mise à l'échelle. Nous tirons avantage des courbes de Bézier pour concevoir une approximation de mouvement non-linéaire depuis seulement quelques images clés et rastérisons une géométrie synthétisée pour reproduire le mouvement. Deuxièmement, nous présentons un algorithme qui fait le rendu de matériaux scintillants à haute fréquence illuminés par de grandes cartes environnementales. En utilisant une combinaison d'un système d'histogrammes de mi-vecteurs compact et des harmoniques sphériques multi échelle, nous pouvons efficacement représenter des normales de surface denses et rendre leurs interactions avec des sources de lumière filtrées de grandes dimensions. Troisièmement, nous introduisons une nouvelle méthode pour faire le rendu de dispersion sous la surface en tirant avantage de l'analyse fréquentielle et du parcours d'un arbre dual. En calculant le transport de la lumière sous la surface en espace image, nous pouvons rapidement analyser la fréquence du signal et déterminer des bandes passantes efficaces que nous pouvons alors utiliser pour limiter notre traversée dans un arbre dual d'ombrage et d'illumination. Finalement, nous démontrons un algorithme novateur d'illumination globale diffuse en temps réel qui utilise des sondes d'irradiance dynamiques. Grâce à des mises à jour efficaces de distribution de radiance, nous pouvons mettre à jour des sondes d'irradiance pendant l'exécution, prenant en compte les objets dynamiques et une illumination changeante, et nous le combinons avec une requête d'irradiance filtrée plus robuste, rendant une grille de sondes d'irradiance dense traitable en temps réel avec des artefacts minimes.

Mots-clés : effets de distribution, transport de la lumière, dispersion sous la surface, rendu temps réel.

Abstract

Modern computer generated imagery strives to be ever more faithful to the physical reality around us, and one such key physical phenomenon is the notion of distribution effects. Distribution effects are a category of light transport behaviors characterized by their distributed nature across some given dimension(s). For instance, motion blur is a distribution effect across time, while depth of field introduces a physical aperture for the camera, thus adding two more dimensions. These effects are commonplace in film and real life, thus making them desirable to reproduce.

In this manuscript-based thesis (*thèse par articles*), we present four papers which leverage, extend or inspire themselves from distribution effects. First, we propose a novel technique to render non-linear motion blur for real-time applications while conserving important scalability and efficiency characteristics. We leverage Bézier curves to approximate non-linear motion from just a few keyframes and rasterize synthesized geometry to replicate motion. Second, we present an algorithm to render glinty high-frequency materials illuminated by large environment maps. Using a combination of a compact half-vector histogram scheme and multiscale spherical harmonics, we can efficiently represent dense surface normals and render their interaction with large, filtered light sources. Third, we introduce a new method for rendering subsurface scattering by taking advantage of frequency analysis and dual-tree traversal. Computing screen-space subsurface light transport, we can quickly analyze signal frequency and determine efficient bandwidths which we then use to limit our traversal through a shading/illumination dual-tree. Finally, we show a novel real-time diffuse global illumination scheme using dynamically updated irradiance probes. Thanks to efficient spherical radiance distribution updates, we can update irradiance probes at runtime, taking into consideration dynamic objects and changing lighting, and combine it with a more robust filtered irradiance query, making dense irradiance probe grids tractable in real-time with minimal artifacts.

Keywords: distribution effects, light transport, subsurface scattering, real-time rendering.

Contents

Résumé		5
Abstract		7
List of table	es	13
List of figu	res	15
Acknowled	gements	27
Introductio	n	29
Chapter 1.	Background	37
1.1. In	tegral Calculus	37
1.1.1.	Multidimensional Integration	38
1.1.2.	Monte Carlo Integration	41
1.1.3.	Parametric Equations	43
1.1.4.	Frequency Analysis	43
1.1.5.	Spherical Harmonics	47
1.2. Li	ght Transport	49
1.2.1.	Light Transport Fundamentals and the Rendering Equation	49
1.2.2.	Foundational Rendering Techniques	50
1.2.3.	Global Light Transport	52
1.2.4.	Local Light Transport and the Microfacet Model	55
1.3. Di	istribution Effects from Realistic Camera Models	57
1.3.1.	Idealized Camera Light Transport	57
1.3.2.	Depth of Field	58
1.3.3.	Motion Blur	60
1.4. Hi	igh-performance Hardware Graphics Pipeline	62
High Perfor	rmance Non-linear Motion Blur	67

1. Introduction	69
2. Previous Work	70
 Method	71 72 72 74
4. Results 4.1. Performance	74 76
 Discussion and Limitations 5.1. Extensions 	76 78
6. Conclusion	79
Scalable Appearance Filtering for Complex Lighting Effects	81
1. Introduction	83
 Previous Work	84 85
2.2. Procedural Texture Antialiasing	85
2.3. Accurate Appearance Filtering	85 86
2.4. Past Appearance Pintering	00
3.1 Baseline Filtered Microfacet Model (G×D)	87 87
3.2. Evaluating $G \times D$	90
4. Filtering Appearance in Space and Direction	90
4.1. Multi-scale NDFs Using Histogram Accumulation	93
4.1.1. Histogram initialization	94
4.1.2. Summed Area Table Histogram Queries	94
4.2. Adaptive Basis-space Integration	94
4.2.1. Spherical Harmonic Preliminaries	95
4.2.2. Multi-scale SH NDFs and Adaptive Integration	96
5. Applications	97
5.1. Appearance Filtered Direct illumination	97
5.2. Filtered Global Illumination	98

	6. Imj	plementation and Discussion	99
	6.1.	Histogram Resolution	99
	6.2.	Gaussian Roughness Lobe Discretization	100
	6.3.	Naïve NDF SAT and Relationship to Han et al. [41]	
	6.4.	Light Source Coefficients	103
	7. Res	sults	104
	8. Co	nclusion and Future Work	104
	Acknow	wledgements	
	A. Fre	esnel and Geometric Term Approximation	108
A	Frequer	ncy Analysis and Dual Hierarchy for Efficient Rendering of Subsur	rface
		Scattering	111
	1. Inti	roduction	113
	2. Pre	evious Work	115
	2.1.	Overview	117
	3. Fou	urier Analysis	117
	3.1.	Fourier Transform of a BSSRDF	
	3.2.	Outgoing Radiance Bandwidth Computation	119
	4. Hie	erarchical Approach	
	4.1.	Hierarchical Surface Integration	120
	4.2.	Dual Hierarchy for Pixel-Surface Integration	121
	5. Imp	plementation	122
	6. Res	sults and Discussions	
	7. Co	nclusion	127
	Acknow	wledgements	
Dy	namic I	Diffuse Global Illumination with Ray-Traced Irradiance Field Probes .	129
	1. Intr	roduction	131
	2. Rel	lated Work	133
	3. Dy	namic Diffuse Global Illumination Probes: Overview	135

4. U	Jpdating Dynamic Diffuse Global Illumination Probes	137
4.1.	Probe Grid Placement	138
4.2.	Generating and Tracing Probe Update Rays	138
4.3.	Secondary Surfel Shading	139
4.4.	Probe Surfel Updates	139
5. S	Shading with Dynamic Diffuse Global Illumination Probes	140
5.1.	Direct Illumination	140
5.2.	Diffuse Indirect Illumination	141
5.3.	Multiple Bounces of Global Illumination	143
6. F	Results	145
6.1.	Probe Count and Resolution	147
6.2.	Ray Tracing Performance	148
6.3.	Probe Texel Format	149
6.4.	Quality Comparisons	149
7. C	Conclusion and Discussion	149
7.1.	Future Work	153
Ack	nowledgements	153
Chapter	2. Conclusion and Future Work	155
2.1.	Real-time Appearance Filtering	156
2.2.	Surface-distributed Ray Tracing	156
2.3.	Appearance Filtering in Time, Lens and Light	157
2.4.	Final Thoughts	158
Reference	ces	159

List of tables

1	Performance comparisons (in milliseconds) at 1920×1080 for the scenes in
	Figures 29, 30, and 32. Sample counts are provided for the linear algorithm. As expected,
	our pre-pass is almost exactly twice as costly since it requires two motion vector passes.
	Our approach is clearly slower, however the difference is not overwhelming and we have
	not yet made any optimization attempts
2	Memory and performance comparisons for our examples
3	Computation times for various parts of the algorithm

List of figures

1	Typical surface representation for a BSDF f_r with an incoming ray ω_i , an outgoing ray ω_o , and a surface normal ω_n . By convention, all vectors point away from the surface. A BSDF displayed in this way is specific to the given ω_i and indicates the attenuation factor of the surface for each given outgoing direction ω_o . For example, the present one indicates a strong specular lobe (with reflections akin to a mirror) with an additional retroreflective (that is, which reflects back towards the incident ray) component	30
2	A Computer Animated Hand: One of the first entirely computer-generated images leveraged both wireframe and simple interpolated shading, with no consideration for any potential light source. CREDITS: EDWIN CATMULL AND FRED PARKE, UNIVERSITY OF UTAH.	30
3	<i>The Adventures of André and Wally B.</i> was one of the first animated short movies entirely created in 3D and already made use of many light transport effects such as motion blur (here seen on the wings of the bee). CREDITS: PIXAR ANIMATION STUDIOS	31
4	Example of a heuristic algorithm in a videogame, here to compute a visually pleasing approximation of motion blur within mere milliseconds [37]	32
5	Real-life examples of depth of field (a) and motion blur (b). Note the blurry background (a) which resembles a typical camera lens' circular aperture, while (b) showcases the notion of "sweeping" over time, most markedly on the flapping wingtips. CREDITS: JEAN-PHILIPPE GUERTIN.	33
6	(a) Typical spherical coordinate convention. The polar angle θ defines the angle from the Cartesian $\hat{\mathbf{z}}$ axis, while the azimuthal angle φ defines the angle around the $\hat{\mathbf{z}}$ axis starting from the $\hat{\mathbf{x}}$ axis. (b) Common surface shading elements. Given a basis such that the normal \mathbf{n} is parallel to $\hat{\mathbf{z}}$ with the origin at the point of interest \mathbf{x} , we define the incoming and outgoing directions ω_i and ω_o , both pointing away from the origin by convention. The half-vector ω_h is defined as the normalized sum of the two directions. The most often used integration region in these circumstances is the hemisphere Ω centered around the normal ω_o .	30
		5)

- Example SAT procedure. Given an arbitrary 2D array (e.g., a normal map) as in (a), the SAT shown in (b) is generated by summing all values less than or equal to the cell's index. To compute the value of all cells the blue region, we sum the two extremal regions (in orange and green) and then subtract the two edge regions (in purple and red). 42

- 13 Ray tracing overview for the three-dimensional case. An arbitrary ray is first projected from the image plane into the scene, and then its corresponding pixel is processed if the ray has hit some geometry. 52
- 15 Example use of irradiance probes, here from Part 4. In this case, the probes are automatically laid out in a grid, but manual placement is possible and often desired. Each probe stores the irradiance of all visible objects from the probe's location. Note in this image that, since irradiance probes can only effectively deal with diffuse materials,

	the glasses on the tables do not exhibit the correct lighting, which is a limitation of the technique from Majercik et al. [63].	54
16	Example environment map, here of a pier at sunset. The severe distortions exhibited are due to the reprojection of a spherical map onto a 2D rectangle. CREDIT: TEXTURIFY.COM	55
17	2D projection of a microfacet model, showing the various microscopic mirrors, or facets, reflecting light. In (a), the light rays are completely unobscured. In (b), some rays are blocked from reaching the facet by a close-by facet. In (c), the reverse happens, and a neighboring facet prevents the light reflected by the facet from leaving the surface	56
18	Illustration of the four main types of BSDFs. (a) exhibits the simple case of the BRDF (with no restriction on the material's thickness, d), while (b) extends it to a BSSRDF by allowing a random walk through the material which must now have a real thickness $d > 0$. (c) and (d) are similar, but tackle transmittance through the surface, with (c) also explicitly restricting thickness d to an infinitesimal layer. A BSDF is a combination of any (or all) of the above.	57
19	Simple diagram of a pinhole camera. Objects reflect light into the pinhole, more generally known as aperture, which crosses through to the image plane some distance away, while other rays are blocked. The resulting image is rotated 180 degrees around the normal at the image plane, but otherwise replicates what an observer would see at the pinhole's location.	58
20	Simple diagram of a thin lens camera. Unlike a pinhole camera, rays can come through an aperture of a given dimension (the lens), at which point they are redirected through a lens in order to hit the image plane at the correct location. Per the thin lens equation, objects at the focal plane o_p will be replicated much as if they were seen in a pinhole camera, as shown in (a), but objects at a distance $o \neq o_p$ closer or farther away will create an image which is in front of or behind the actual image plane, creating blur, as axbibited in (b)	50
21	Exhibited in (b) Circle of confusion diagram. o is the distance between the lens and the object, o_p is the focal plane distance, f is the focal length of the lens, and A is its aperture diameter. The dotted blue lines represent an object "at infinity", which, as per the lens equation, focuses at the focal length f	59
22	Example computer-generated images with differing focal distances. In (a), the focus is set close to the camera, making nearby objects sharp, but both distant and very close objects blurry. In (b), the focus is set farther, causing the previously sharp objects to	00

	become blurry. Also note in this image that the circular shape of the camera's aperture is visible as a slightly brighter circle on the foreground cup. CREDITS: MORGAN MCGUIRE [67] AND BLEND SWAP USER WIG42	61
23	Simple motion blur example. (a) shows a moving circle in three snapshots $t \in \{0, \frac{1}{2}, 1\}$ while (b) shows the rendered blur for the same motion	63
24	Simple motion vectors interpretation. Each pixel contains (x,y) velocity for that pixel's motion since the previous frame, here represented as arrows for convenience. This example can be interpreted as follows: the blue circle is moving towards the bottom right corner of the screen, while the orange rectangle is spinning counter-clockwise while facing the camera. All motion stored here is purely linear, a notable approximation of this method.	64
25	A typical hardware graphics pipeline, here using Microsoft Direct3D terminology. Blue rectangular stages are fixed-function, which is to say they have only a limited ability to be configured or modified and cannot be directly programmed. Yellow rounded rectangles indicate programmable stages, whose behavior can be completely user-defined within some bounds. Dashed borders indicate optional stages or pathways which will not necessarily be executed unless requested. Compute shaders are typically not part of the strict graphics pipeline, but are now commonly used to replace any programmable stage as desired.	65
26	Smooth motion blur on a variety of complex, potentially non-linear motions, computed in 3.5 to $6.5 ms$ at 1920×1080	69
27	Visualizing scatter and gather operations. In scatter-based algorithms, each data point (i.e., pixel) deposits data onto neighboring points; for gather-based algorithms, each point queries its neighbors to compute its final value. Scatter-as-gather emulates the former using the latter.	70
28	Grid scatter pass: we fit Bézier curves to a grid of pixel-aligned vertices with a vertex shader, querying an object's previous and next positions. We discretize the curves into line segments in a geometry shader and then rasterize the segments. We compute the spatially-varying line color based on the originating pixel's color and distance-based weight in a pixel shader.	71
29	The helicopter scene has significant rotational motion, a common failure case for linear motion blur: linear algorithms fail to properly convey the scale of motion, they cause over- and under-blurring, and they either give the impression of pure linear motion (as	

seen on the blades) or cause a pinwheel artifact (as seen on the tail rotor). This is due to clamping to dominant linear velocity directions at different angles, across tiles...... 75

30	This teapot scene showcases chaotic movement with many superimposed velocities on each pixel of the screen. We note that most pixel motion is slightly curved, which is more faithfully represented with our non-linear approach. Moreover, there is a high variability in the motion vectors, which is only partially captured by linear tile-based algorithms, while our algorithm properly accounts for all directions and all magnitudes of motion.	77
31	Comparison between non-linear blur using the basic, single-pass algorithm (a) and using our modified two-pass rendering (b). Segmenting static objects allows us access the background color behind the blurred objects, resulting in a more accurate relative weighting between the foreground and background, completely eliminating the most distracting visual artifact of our current approach	78
32	The jumping jack scene illustrates the algorithms' behavior with animated characters. Our approach better conserves the blur on the character's knee, whereas linear algorithms can almost completely miss this effect.	79
33	We filter direct (a) and global illumination (b) with high-frequency appearance and complex emitters $\sim 30 \times$ faster than the state-of-the-art and $\sim 10\%$ the memory footprint. Unless otherwise stated, our results have converged and residual noise is due to high-frequency appearance.	83
34	While high-frequency sparkle-like appearance is highlighted by strong directional or point lighting (middle row), environmental lights (top row) still contribute significantly to sparkly appearance. Our method (right column) generates converged images with these effects in less than twice the time needed to generate converged results with a smooth microfacet model (left column), and $\sim 10 \times$ faster than Yan et al. [113] (not shown).	84
35	Previous methods only integrate over a <i>spatial</i> footprint, before evaluating filtered appearance in the direction of a point/directional source.	88
36	Our method integrates high-frequency appearance variation over both spatial footprint and incident lighting directions. To integrate over all light directions ω_i , as is needed for environment lighting, previous work requires many Monte Carlo samples, each of which involves an expensive integral over \mathcal{P} . Our method solves both integrals with a single evaluation.	89

37	Middle: Our method generates a converged filtered result in 4.89s (1spp) in a scene with a high-resolution normal map and environmental lighting. Top halves: equal time for our baseline (2spp, left) and Yan et al. [113] (1spp, right). Bottom halves : Equal quality rendering requires 16,384spp and 13m:8s with our baseline and 512spp and 2m:25s with Yan et al. [113]	91
38	We use texture-space accumulated NDF histograms to efficiently query the multi-scale NDF histogram for an <i>arbitrary filtering footprint</i> \mathcal{P} using four lookups into this SAT-like data structure (histogram radial sizes not to scale, above). Figure 43 illustrates the visual impact of different histogram resolutions.	92
39	We adapt the shading to the minimum between the max frequency bandwidth of the multi-scale NDF in a footprint (top row) and the max bandwidth of the lighting environment. We visualize environment maps bandlimited to this <i>effective shading bandwidth</i> (bottom).	93
40	We can generate converged unshadowed direct illumination with a single shading sample (left; 0m:42s) and then apply the technique of Heitz et al. [44] to compute shadowing numerically (with 4spp in 1m:08s)	98
41	We adapt the technique of Belcour et al. [15] to compute secondary path vertex footprints for filtered appearance with global illumination	100
42	When integrating with respect to ω_h our basis-space approach automatically discards NDF normals outside the shading frame (grayscale) and uses an appropriately warped incident illumination distribution.	101
43	Lower resolutions for our accumulated NDF histogram result in angular blurring of details (left), albeit a much faster rendering $(2.5 \times, here)$	101
44	When storing rotated Gaussian roughness SH coefficients below the spherical histogram resolution (left), visible re-sampling artifacts can appear. We use a conservative discretization of $4 \times$ the histogram's resolution	103
45	Cutlery scene using the same environment map and point light, shadowed direct lighting only. Our method approached the ground-truth in 50.7s while Yan et al. [113] required 7m:6s to reach the same quality at 2116 samples per pixel	105
46	Additional results for both direct and global illumination. Both (a) and (b) showcase additional models and environment maps with very high fidelity lighting while remaining temporally stable and fast to compute, 2m:27s and 1m:13s, respectively. Figures (c) and (d) are a breakdown of a frame from the Snails scene, showcasing the performance of our	

method across both direct and first-bounce indirect illumination in a global illumination context. In both cases, our results handily beat Belcour et al. [**15**] at equal time (11 spp). 106

47 We rely on a spherical histogram that discretizes the upper hemisphere into bins (see Section 4.1). The resolution of this histogram directly impacts the minimally reproducible detail size: lower resolutions progressively lose detail in the normal map, effectively smoothing it out. Our selected setting (9×32) is adequate for all scenes we 48 Impact of the resolution of the rotated ZH Gaussian roughness lobes (see Section 6.2). As is apparent, the resolution does not have a significant effect on image quality for the 49 Impact of the resolution of the environment light's SH coefficients table (see Section 6.4). As with Figure 48, the effect of a lower resolution is minor. We again conservatively use 50 Impact of the chosen band limit for the NDF or light spherical harmonics. Band limiting either is equivalent. Artifacts start to appear when either band limit is chosen below the true band limit of the respective signal (i.e., the maximum of the band limit of the NDF 51 Effect of the simplification made in Equation (46) and the proposed alternative in Equation (54). Both images were generated using our reference $G \times D$ implementation

- 52 Equal quality comparison for global illumination in the snails scene. Belcour et al. [15] took 121m:24s to render at 2048 samples per pixel while our method finished in 3m:32s.109

55	Assuming that the incoming light-field has infinite bandwidth, we estimate the bandwidth of the outgoing light-field $[B_s, B_{\theta}]$ as the bandwidth of the BSSRDF along the outgoing
	spatial positions and directions (\mathbf{a}). The interaction with the material limits the spectrum
	of the local light-field by the BSSRDF spatial and angular bandwidth (b). To estimate
	the bandwidth at the camera position, we first shear spatially the spectrum to account for
	curvature (c). Then, we scale spatially to account for foreshortening (d) and finally shear
	angularly the spectrum to account for transport (e)
56	We compare IlluminationCut (Bus et al. [17]) to the method of Jensen and Bulher [54]
	on the BUNNY scene with various ε error bound settings. The cost of computing the
	upper-bound metric (Eq. 13 of Walter et al. [106, Eq. 13]), which requires multiple
	BSSRDF evaluations, precludes the direct applicability of IlluminationCut to adaptive
	BSSRDF shading
57	First row: The sampling rate s_p computed from the screen-space bandwidth estimation.
	Second row: Pixel areas from which the sampling rate predicts an adequate
	approximation of the outgoing radiance variation
58	We compare our approach (red) to Jensen and Bulher [54] (blue) for different settings of
	ε . We highlight the $\varepsilon \in [0.01, 0.2]$ values and consistently reach equal-quality (measured
	in RMSE; y-axis) in less render time (in seconds; x-axis). The PICNIK scene challenges
	the assumptions of our work, and we only obtain equal-quality benefit at lower rendering
	times (albeit enough for visual convergence)
59	The TOAD scene has a bumpy geometry with detailed textures. We compare the
	difference images of the multiple scattering term against the ground-truth for an equal
	rendering time (196s). The difference images are scaled by 50 for visualization. Our
	approach achieves more accurate estimation than the single-tree in the same rendering
	time
60	The BUNNY scene. We compare the difference images of the multiple scattering term
	against the ground-truth for an equal rendering time (60s). The difference images are
	scaled by 200 for visualization. In this example, our approach removes artifacts under
	the tail and reduces Moiré patterns present in the single-tree approach126
61	Combined with state of the art glossy ray tracing and deferred direct shading, our method
	(left) generates full global illumination in dynamic scenes that are visually comparable
	to offline path traced results (right) but several orders of magnitude faster: 6 ms/frame,
	versus 1 min/frame in this scene (on GeForce RTX 2080 Ti at 1920×1080). Insets
	isolate the direct lighting contribution and visualize the probe locations

62	Previous interactive GI methods suffer from artifacts that often necessitate heuristic solutions, typically based on art-direction or technical constraints. Visual artifacts in these methods can manifest themselves in various forms: (in reading order) light leaking, lightmap seams, visibility/occlusion undersampling, and inter-voxel visibility mismatches. 132
63	Spherical irradiance and depth textures. We encode spherical data in an octahedral parameterization, packing all the probes in an atlas. One-pixel texture gutter/border to ensure correct bilinear interpolation, and additional padding aligns probes on 4×4 write boundaries. 136
64	Left to right: direct illumination only, direct illumination with <i>one bounce</i> of indirect diffuse illumination (computed with spherical irradiance updated by our dynamic filtered ray casting approach), and the fully converged <i>multi-bounce</i> global illumination that iteratively incorporates bounced lighting computed across probes
66	Shading of a surfel X . We sample each probe in the 8 probe cage using the surface normal n in world space. We backface weight each probe P using dir , the direction from X to P . The mean distance stored for P is represented by r . To avoid sampling visibility near the visibility function boundary (i.e. the surface), we offset from the world space position at X based on the surface normal and the camera view vector
67	Timings for the indirect light components of a single frame render using $32 \times 8 \times 32$ probes with 64 rays/probe. Probes were at 8×8 resolution using RGB10A2 format for color and RG16F format for depth. Timings were taken using glTimerQuery. We allocate time for the combined ray cast according to the proportion of rays for diffuse and glossy. We did not profile the unoptimized modular passes for parts of the system outside our contributions (shadow maps, AO, G-buffer generation) though we include unoptimized glossy ray cast and unoptimized glossy indirect shade in the final row for context.
68	Irradiance interpolation and sampling in a closed room scene (a), where light enters only from a single door opening (b). Images (b) through (i) place the camera at the back corner of the room, illustrating how light leaking artifacts from traditional irradiance probes (c) are progressively compensated for using the terms in our novel interpolant (d) - (f). We visualize a 2× error image (h) between our final result (f) and the path traced reference (g)
69	Comparison of probes with no visibility test, visibility test by a low-res ray cast, and our new variance visibility

70	Indirect shadows with increasing probe grid resolution. As the resolution of the probe grid increases, the indirect shadow approaches the pathtraced reference without the overdarkened look of SSAO
71	Ray throughput in Gigarays per second. Timings were taken on our Greek Temple Scene (876127 primitives) using an NVIDIA RTX 2080 Ti. For reference, we used $32 \times 8 \times 32$ probes with 64 rays/probe for our largest scenes
72	Quality comparison across a selection of probe resolutions and densities. Probe resolution is given as $X \times Y \times Z$, with Y increasing towards the camera. Note that even at low resolution, an image rendered with a sufficient number of probes looks identical to the path-traced reference. 146
73	Quality comparison across a selection of probe resolutions and densities for a more complex scene
74	Comparing path tracing (left) to our dynamic diffuse global illumination probes (right), with full diffuse global illumination, in a box with a dynamically translating dragon. See our video supplement for a real-time animation
75	Color precision at 128 rays/probe/frame. GL_RGB5A1 requires a low hysteresis $\alpha = 0.8$ in order to not fall below the blending threshold with many rays and suffers from flicker and oversaturation. The other formats, using $\alpha = 0.95$, are nearly indistinguishable from one another although GL_R11G11B10F has a greenish tint because it cannot represent exact grays. GL_RGB10A2 balances quality and size. Note that GL_RGB8 gives less precision but requires the same 32 bits/texel storage on modern GPUs due to word alignment. GL_RGB10A2 and GL_RGB8 are too dark because they lack the dynamic range of floating point.
76	Time-lapse images showing fully dynamic GI with moving geometry. In this example, not only are a large number of spheres animating and casting complex GI, but they are also moving <i>through</i> the probes, which would lead to objectionable shadow leaks without correct occlusion. See our supplemental video for a real-time animation 151
77	Time-lapse showing different times of day simulated with dynamic lighting. See our supplemental video for a real-time animation
78	Our pillars test scene initialized with an enclosing 8x4x8 probe grid. The grid is perfectly aligned with the box in the top left image. All other images have a rotation and translation of the grid. Some images are chosen to intentionally break the algorithm by leavin part of the scene uncovered by probes (bottom left). Others are chosen at

random. As long as there are probes covering the area being shaded, our algorithm is	
robust to rotation and translation of the probe grid. See our supplemental video for a	
demonstration of multiple random rotations, including some failure cases for positions	
outside the rotated and translated probe grid 152	

65 Direct illumination (left) versus full (diffuse, glossy & specular) global illumination (right) computed using dynamically-updated irradiance probes and ray-traced reflections. 154

Acknowledgements

J'aimerais premièrement remercier mes parents, qui m'ont supporté tout au long de ce processus et bien au-delà. Par la suite, bien évidemment, je remercie grandement Derek, qui m'a soutenu malgré les embûches et détours de cette thèse.

Je veux aussi gracieusement remercier tous ceux qui ont collaboré avec moi, tout particulièrement mais sans ordre précis: Morgan McGuire, Luis E. Gamboa, Olivier Mercier, Zander Majercik, Olivier Pomarez, Mark Meyer, Douglas Lanman, Eric Bourque, Laurent Belcour, Toshiya Hachisuka, David Milaenen, et j'en passe... Vous m'avez tous aidé à compléter ce long chemin!

Finalement, je salue et remercie le comité d'évaluation pour son travail et son attention.

Introduction

For several decades, computer graphics have been playing a growing role in cinema, television and video games. With computer-generated imagery (CGI), it is possible to create richer stories, more convincing worlds and more imaginative characters than ever before, all the while making these new possibilities more accessible to up and coming artists, filmmakers and game developers. But, computer graphics does have a cost, taking anywhere from seconds to days or even weeks to produce the spectacular imagery we associate with it today, and so it has become key to be able to reduce this cost as much as possible through algorithms and frameworks capable of optimizing or simplifying the tremendous amounts of calculations necessary to produce the aforementioned results.

However, modern computer graphics research has now reached such a level of maturity that multiple specializations exist, with each addressing specific issues, be it from a simulation, appearance modeling, or even user interaction standpoint. This thesis focuses on one of the most well-established specializations: light transport.

Light transport, at its most fundamental level, models the interaction of light with the real world. It defines everything from how light propagates from a source such as the sun, to how it interacts with surfaces or through volumes such as fog. While some interactions are fairly simple, such as travel through a vacuum (ignoring relativistic effects), others can become incredibly complex, such as light bouncing off multiple surfaces before hitting a distant object, creating intricate patterns along the way.

The fundamental behavior of light transport can be summarized elegantly in the *rendering equation* [**56**],

$$L_o(\mathbf{x}, \boldsymbol{\omega}_o, \boldsymbol{\lambda}) = L_e(\mathbf{x}, \boldsymbol{\omega}_o, \boldsymbol{\lambda}) + \int_{\Omega} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \boldsymbol{\lambda}) L_i(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\lambda}) (\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_n) \, \mathrm{d}\boldsymbol{\omega}_i, \tag{1}$$

which explicitly defines the outgoing light L_o at a surface **x** leaving in direction ω_o and with wavelength λ as a function of the emitted light L_e at the surface (with the same outgoing direction and wavelength) and of the incoming light L_i from any direction, modulated by the bidirectional scattering distribution function (BSDF) f_r , which describes the innate properties of the surface and its behavior for a pair of directions (ω_i, ω_o) relative to the surface normal ω_n (see Figure 1). A BSDF can model anything from wood to metals to plastic or even skin, with extensions such as bidirectional scattering-surface reflectance distribution functions (BSSRDF) modeling light as if it penetrated through the surface to interact within it, exiting at a different location as it bounces out of the material.



Fig. 1. Typical surface representation for a BSDF f_r with an incoming ray ω_i , an outgoing ray ω_o , and a surface normal ω_n . By convention, all vectors point away from the surface. A BSDF displayed in this way is specific to the given ω_i and indicates the attenuation factor of the surface for each given outgoing direction ω_o . For example, the present one indicates a strong specular lobe (with reflections akin to a mirror) with an additional retroreflective (that is, which reflects back towards the incident ray) component.



Fig. 2. A Computer Animated Hand: One of the first entirely computer-generated images leveraged both wireframe and simple interpolated shading, with no consideration for any potential light source. CREDITS: EDWIN CATMULL AND FRED PARKE, UNIVERSITY OF UTAH.

During its beginnings, computer graphics research did not really take into consideration light transport in a strict and physical way. For example, the novel rendering of *A Computer Generated*

Hand [81] shown in Figure 2 or even *The Adventures of André and Wally B*. [4] shown in Figure 3, one of the first works of world-renowned Pixar Animation Studios, only used very simple techniques and coarse approximations. With the growth in computing power and interest in the topic, light transport has become a fundamental component of rendering and computer graphics in general, allowing the creation of ever more realistic images, but also giving artists and authors more and more expressive and flexible tools to express themselves, whatever their objectives may be.



Fig. 3. *The Adventures of André and Wally B.* was one of the first animated short movies entirely created in 3D and already made use of many light transport effects such as motion blur (here seen on the wings of the bee). CREDITS: PIXAR ANIMATION STUDIOS.

Nowadays, the quest towards realism translates into innovations such as physically-based rendering (PBR), which is grounded in universal and meaningful parameters of matter to define its appearance, or global illumination, which simulates light transport in a scene over very large distances and along complex paths, hence modelling its real world behaviour.

Of course, many of these algorithms come with an important performance cost. Realism and precision are unfortunately not free, and therefore the computer graphics community must always work to refine these algorithms, make them more efficient, less demanding in computing or memory. To do this, many approaches are popular, from a simple heuristic based on a phenomenological analysis to create a convincing approximation all the way to using machine learning (ML) to solve complex optimization or classification problems, among others.

While scene interactions such as light scattering (e.g., when encountering fog or other such transmissive media) or surface appearance are well understood and heavily researched, realistic



Fig. 4. Example of a heuristic algorithm in a videogame, here to compute a visually pleasing approximation of motion blur within mere milliseconds [**37**].

camera models are often neglected due to their large cost and complexity versus simpler models such as the pinhole camera (see Section 1.3.2). The characteristics of the sensor and camera lens are often approximated using post-processing steps and heuristics. This thesis instead directly tackles a particular category of camera and sensor effects known as distribution effects, which involve additional dimensions beyond position (usually the Euclidean form (x,y,z)) and direction (usually the spherical form (ϕ, θ) as are typically seen in the rendering equation (Equation (1)). These additional dimensions are used to model new emergent properties of light transport which can be split in two broad categories: depth of field and motion blur. The former appends the position on the *camera lens* to the list of dimensions (usually (u,v) Euclidean coordinates on the circle). The latter adds time as a dimension, allowing the rendering equation to model camera responses to scene modifications across time, such as character animations, physical simulations, lighting changes, etc. In both cases, the final effect is various forms of blurring which are caused by a shifting of which surfaces affect which pixels on the image through specific light transport interactions in these new dimensions. These effects are vital to modeling more realistic cameras and various optimizations can be found by performing a more holistic analysis of light transport that directly includes them. More information on both depth of field and motion blur may be found in Section 1.3.

Unfortunately, these new dimensions also translate into often exponential performance costs thanks to the infamous curse of dimensionality problem, which notes that while, for instance, a



(a) Depth of field

(b) Motion blur

Fig. 5. Real-life examples of depth of field (**a**) and motion blur (**b**). Note the blurry background (**a**) which resembles a typical camera lens' circular aperture, while (**b**) showcases the notion of "sweeping" over time, most markedly on the flapping wingtips. CREDITS: JEAN-PHILIPPE GUERTIN.

factor of $2 \times$ growth in one dimension remains a factor of $2 \times$ overall, it becomes a factor of $4 \times$ in two dimensions, $8 \times$ in three, and so forth.

As a result, these distribution effects are often added separately, either as a post-process to the entire rendering pipeline, or as an approximation entirely, as with the example in Figure 4. This reduces their cost by making it possible to simply lower overall quality or defer computations until more resources are available. Unfortunately, this is also often a very limited avenue for further improvements, since much information is lost in this two-step approach. Instead, we posit that we can integrate these new dimensions very tightly into the entire rendering pipeline and improve performance and results by evaluating them earlier and in a more organic fashion with the rest of the pipeline.

Furthermore, this approach has many additional interesting applications. We have found that there are many other instances where extra dimensions are introduced at some point in the rendering pipeline which do not necessarily have anything to do with distribution effects from a narrow perspective, but which can still benefit from a similar holistic approach. We call this concept *extended distribution effects*.

In this thesis, we explore the concept of extended distribution effects in multiple algorithms to take advantage of optimizations, correlations and other such improvements that this may lead to. In Chapter 1, we briefly cover necessary prerequisites to understand light transport, mathematical and hardware concepts key to the papers presented in the other chapters. The remaining chapters comprise four papers published in high impact conferences or journals on computer graphics and rendering.

Common Distribution Effects. Our exploration begins with what one might call "traditional" distribution effects, that being depth of field and motion blur (see Sections 1.3.2 and 1.3.3 respectively). In a real-time context such as games or interactive media, distribution effects are often approximated to a high degree, producing visually acceptable results which often stray quite far from what a ground-truth implementation would create. Even in an offline rendering context such as film, motion blur is often approximated as a post-process using very limited information, discarding effects such as non-linear movement.

One such approximation found in both scenarios is rendering single frames and approximating the motion from so-called *motion vectors* (see Section 1.3.3). We have leveraged this technique in prior work [**37**] and noticed substantial weaknesses which we sought to address. The result can be found in Part 1, where we extend the notion of motion vectors by instead storing actual 3D positions for the prior *and future* frame, then leverage Bézier curves (see Section 1.1.3) to produce non-linear motion between the three interpolated locations (prior, current and future).

Appearance Filtering. Our next investigation concerns appearance filtering. Numerous articles approach this topic from a higher dimensional perspective. Typically, this is done by focusing on one of two areas: either integrating over the light source, as in e.g. Heitz et al. [43], or integrating over the surface, as in Yan et al. [112].

Our immediate intuition was to attempt to perform this integration over *both* the light source and the surface. The results of this approach can be found in Part 2, where we successfully leverage spherical harmonics (see Section 1.1.5) for fast evaluation of large scale area lights such as environment maps (see Section 1.2.3) combined with summed-area tables (see Section 1.1.2) to store and rapidly compute the normal distribution function (see Section 1.2.4) of arbitrarily large footprints.

Subsurface Scattering. There obviously exist topics where additional dimensions are an inevitability in proper analysis and rendering, and one such topic is subsurface scattering (see Section 1.2.4). In short, whereas most surfaces can be represented with a high degree of fidelity simply by considering light coming and leaving from the same location on a surface for a given ray, there are other materials, notably skin, but also milk, marble, etc., for which light can also penetrate beneath the surface and interact with the volume before exiting elsewhere on the surface. This phenomenon is called subsurface scattering, since light is scattered under the surface of a material. While this is an orthogonal topic to our main focus, we have decided to take a holistic approach to improving the realism, quality and performance of light transport in order to improve sensor and camera models.

We have approached this topic predominantly using principles of frequency analysis (see Section 1.1.4) to efficiently evaluate the shading of a surface on many points at a time. This is done by determining a reasonable maximum bandwidth (see Section 1.1.4) for the frequency of the radiance (see Section 1.2.1) signal on a per-pixel level, and then using this bandwidth to adaptively

compute the BSSRDF integration (see Section 1.2.4) through the use of a *dual-tree* (see Section 4) to traverse a combined sparse hierarchy of pixel and illumination samples. The results of this algorithm can be found in Part 3.

Dynamic Irradiance Probes. Our final project is focused on the exciting new technologies available in modern computer hardware (see Section 1.4), chiefly ray tracing acceleration. This new category of hardware allows, for the first time, real-time ray tracing in modern applications such as games without compromises as to visual fidelity or accuracy.

Real-time ray tracing currently tends to focus on a few key effects which have been lacking in games for years: accurate glossy surface reflections and global illumination (see Section 1.2.3). Our approach remains fairly conservative for glossy surface reflections, but we employ a novel idea for global illumination: rather than, as most current algorithms do, sparsely sample the scene each frame to construct a limited, view-dependent global illumination framework, we construct a dense grid of irradiance probes (see Section 1.2.3) which we update each frame, creating an efficient, scalable and stable global illumination solution which leverages the current state of the art in precomputed global illumination and applies it to dynamic GI. Our results can be found in Part 4.

Conclusion and Future Work. Finally, we go over the numerous potential future projects derived directly from all of our existing body of work, and then conclude with a summary of our findings and a discussion on the viability of extended distribution effects as a category of algorithms.
Chapter 1

Background

This chapter covers all prerequisites necessary to understand our core contributions as presented in the articles which follow. Specifically, Section 1.1 covers integral calculus and its applications to our research on global and local light transport as well as improved appearance filtering. Section 1.2 reviews fundamental concepts of light transport and appearance modelling as applied in our work. Then, Section 1.3 dives deeper into the two primary distribution effects we treat: depth of field and motion blur. Finally, Section 1.4 gives a broad overview of modern graphics rendering hardware and software frameworks.

1.1. Integral Calculus

Integration is one of the most powerful concepts of calculus and is extensively used in rendering. Indeed, from the rendering equation [56] to the basics of radiometry (see Section 1.2.1) and including notions such as distribution effects (see Section 1.3), integral calculus subtends our work. This section is dedicated to giving a brief overview of its main properties, theorems and some more advanced concepts which augment it.

First, we briefly cover some concepts of multidimensional integration which are fundamental to the analysis of light transport through the rendering equation [56] and its various components such as bidirectional reflectance/scattering distribution functions (see Section 1.2.4), weaving in some basic concepts of linear algebra such as bases and coordinate systems which are required to understand the various perspectives used throughout our analysis, especially in the surface appearance modelling of Part 2.

Second, we make a brief detour on the topic of parametric equations, which are a powerful way to represent curves and surfaces using independent variables known as *parameters*. Parametric equations can describe a large number of curves, but we concentrate here on one of the most famous – Bézier curves – which have several interesting properties making them commonplace in computer graphics and especially vector-based graphics software. As part of our light transport research, we have found that Bézier curves are an excellent representation for non-linear motion in Part 1,

giving better accuracy than linear approximations while not being brute force like a dense sampling approach might be.

Third, we dive into Fourier analysis and the notion of the Fourier transform, which can be key to the integration of particularly complex or periodic functions thanks to the transformation into a so-called *dual space* such as frequency space. In these dual spaces, some operations that were prohibitively costly or outright not solvable analytically can often become straightforward, simplifying the analysis. We extensively leverage Fourier transforms in both Parts 2 and 3.

Finally, we look at spherical harmonics, which often appear in multidimensional integrals for certain fields such as particle physics or light transport. Spherical harmonics, as a complete set of orthogonal functions defined on the surface of the sphere, can be used to represent any function on said surface, allowing us to decompose them into *harmonics*, each of which can be associated with a specific frequency, from which we can use similar tools as we found in Fourier analysis. We use them for efficient data compression by relating them with the frequency of various signals in Part 2.

1.1.1. Multidimensional Integration

One of the simplest yet most powerful notions in calculus is the idea of changing integration variables to create a more tractable integrand or simpler boundaries. We have found this especially invaluable when evaluating appearance models on complex light paths where the representation of the surface interactions was non-trivial in Euclidean world space, but much more tractable in a more specialized or local coordinate system which is achieved through a change of variables.

This process is fairly simple in practice: given an integral of the form

$$\int_{\Omega} f(\mathbf{x}) \, \mathrm{d}\mathbf{x}$$

with $\mathbf{x} \in \mathbb{R}^n$ a vector of *n* variables and $\Omega \subseteq \mathbb{R}^n$ the integration domain, it is possible to create a new formulation

$$\int_{\bar{\Omega}} \bar{f}(\bar{\mathbf{x}}) \left| \mathbf{J}_{g} \right| \mathrm{d}\bar{\mathbf{x}}$$

which transforms the variables according to an arbitrary function $\bar{\mathbf{x}} = g(\mathbf{x})$ using the *Jacobian* of the transformation,

$$\mathbf{J}_{g} = \begin{bmatrix} \frac{\partial g_{x}}{\partial x} & \frac{\partial g_{x}}{\partial y} & \cdots \\ \frac{\partial g_{y}}{\partial x} & \frac{\partial g_{y}}{\partial y} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$
(1)

to correctly account for any change in the differentials.

A common type of transformation function can convert a coordinate system from one *basis* to another. A basis is a set of *N basis vectors* which can describe any *N*-dimensional vector **v** through a

linear combination of said basis vectors. We will cover two such basis changes here. First, we define the traditional Cartesian coordinates (x,y,z) which fully describe 3D space as three orthonormal basis vectors $\hat{\mathbf{x}} = (1,0,0), \hat{\mathbf{y}} = (0,1,0), \hat{\mathbf{z}} = (0,0,1)$. A Cartesian integral will have the form

$$\int_{x} \int_{y} \int_{z} f(x, y, z) \, \mathrm{d}z \, \mathrm{d}y \, \mathrm{d}x.$$
⁽²⁾

One of the most useful alternative basis is known as *spherical* coordinates, which, as the name implies, maps 3D space on the surface of an infinite number of spheres of varying radii. Spherical coordinates are defined according to (r, θ, φ) , with *r* the distance (or radius) from the origin, θ being the *polar* angle and φ the *azimuthal* angle.



Fig. 6. (a) Typical spherical coordinate convention. The polar angle θ defines the angle from the Cartesian \hat{z} axis, while the azimuthal angle φ defines the angle around the \hat{z} axis starting from the \hat{x} axis. (b) Common surface shading elements. Given a basis such that the normal **n** is parallel to \hat{z} with the origin at the point of interest **x**, we define the incoming and outgoing directions ω_i and ω_o , both pointing away from the origin by convention. The half-vector ω_h is defined as the normalized sum of the two directions. The most often used integration region in these circumstances is the hemisphere Ω centered around the normal ω_n .

The transformation function which converts from spherical to Cartesian coordinates is

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = g(r, \theta, \varphi) = \begin{bmatrix} r\sin\theta\cos\varphi \\ r\sin\theta\sin\varphi \\ r\cos\theta \end{bmatrix},$$

giving the Jacobian

$$\mathbf{J}_{g} = \begin{bmatrix} \sin\theta\cos\varphi & r\cos\theta\cos\varphi & -r\sin\theta\sin\varphi\\ \sin\theta\sin\varphi & r\cos\theta\sin\varphi & r\sin\theta\cos\varphi\\ \cos\theta & -r\sin\theta & 0 \end{bmatrix}$$

and the determinant

$$\left|\mathbf{J}_{g}\right|=r^{2}\sin\theta$$

to finally transform Equation (2) into

$$\int_{0}^{\infty} \int_{0}^{\pi/2} \int_{0}^{2\pi} f(r\sin\theta\cos\varphi, r\sin\theta\sin\varphi, r\cos\theta) r^{2}\sin\theta \,\mathrm{d}\varphi \,\mathrm{d}\theta \,\mathrm{d}r \tag{3}$$

A more niche transformation is often used specifically in computer graphics. Given a surface to shade and incoming and outgoing directions ω_i and ω_o (e.g., a ray of light bouncing on a surface), respectively, it is fairly typical to perform an integral over some spherical subdomain (such as the hemisphere around a surface normal) along either of these directions; an example of one such setup is given in Figure 6b. This can be expressed as an integral of the form

$$\int_{\Omega} f(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \, \mathrm{d}\boldsymbol{\omega}_o$$

with **x** indicating other potential parameters (e.g., a surface's roughness, normal direction, light wavelength, etc.) and ω_i and ω_o being interchangeable. Depending on the form of f, it can be valuable to move to what is known as *half-vector space*, such that the integration domain is computed along a half-vector

$$\omega_h = g(\omega_o) = \frac{\omega_i + \omega_o}{\|\omega_i + \omega_o\|}.$$
(4)

Without loss of generality, it is possible to transform the arbitrary spherical coordinate basis used initially into one where ω_i is aligned with the \hat{z} axis. This in turn means that $\theta_h = \theta_o/2$ due to being halfway between ω_i and ω_o , and that $\varphi_h = \varphi_o$ so that all three vectors may lie on the same plane. From there, we can see that our Jacobian is of the form

$$\mathbf{J}_{g} = \begin{bmatrix} \frac{\partial \theta_{h}}{\partial \theta_{o}} & \frac{\partial \theta_{h}}{\partial \varphi_{o}} \\ \frac{\partial \varphi_{h}}{\partial \theta_{o}} & \frac{\partial \varphi_{h}}{\partial \varphi_{o}} \end{bmatrix} \left| \mathbf{J} \{ x_{o}, y_{o}, z_{o} \to \theta_{o}, \varphi_{o} \} \right| \left| \mathbf{J} \{ \theta_{h}, \varphi_{h} \to x_{h}, y_{h}, z_{h} \} \right|$$

with the additional Jacobians being required to convert from Euclidean space to solid angles and then back again. These Jacobians are already known, being

$$\left| \mathbf{J} \{ x_o, y_o, z_o \to \theta_o, \varphi_o \} \right| = \sin^{-1} \theta_o \qquad \left| \mathbf{J} \{ \theta_h, \varphi_h \to x_h, y_h, z_h \} \right| = \sin \theta_h$$

and thus,

$$\mathbf{J}_{g} = \begin{bmatrix} \frac{1/2}{2} & 0\\ 0 & 1 \end{bmatrix} \frac{\sin \theta_{h}}{\sin \theta_{o}} \qquad \qquad |\mathbf{J}_{g}| = \frac{\sin \theta_{h}}{2\sin 2\theta_{h}} = \frac{1}{4\cos \theta_{h}} = \frac{1}{4\omega_{i} \cdot \omega_{h}}.$$
 (5)

This relationship is key to many algorithms, and we have used it extensively in Part 2, where our histogram encoding uses half-vector space to more efficiently represent a distribution function of the normals over a surface.

1.1.2. Monte Carlo Integration

In many situations, a function $f(\mathbf{x})$ will need to be integrated over a certain domain Ω , but the resulting integral is not analytically computable. This is where numerical integration becomes necessary, allowing us to approximate the integral regardless of its form. One of the most commonly used methods is known as *Monte Carlo integration*. Given an integral of the form

$$I = \int_{\Omega} f(\mathbf{x}) \, \mathrm{d}\mathbf{x},$$

the approximate Monte Carlo formulation is

$$I \approx Q_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)}$$
(6)

where $p(\mathbf{x})$ is the *probability density function* (PDF) which indicates the likelihood of any given sample \mathbf{x}_i of being picked. This approximation is possible thanks to the law of large numbers, which states that $\lim_{N\to\infty} Q_N = I$. More formally, the PDF is defined for a random variable X such that

$$\Pr[a \le X \le b] = \int_{a}^{b} f_X(x) \,\mathrm{d}x,$$

where Pr is the probability of the random variable X falling between domain bounds a and b, and $f_X(x)$ is its PDF.

The selection of the PDF is key in ensuring the effectiveness of Monte Carlo methods: while using a completely uniform PDF (i.e., $p(x) \propto 1$) will eventually converge to the real value of the integral, it may take a significant amount of samples since the approximate PDF does not conform to the real (but unknown) form of the integral. By contrast, a hypothetical PDF which perfectly matches the integrand (i.e., $p(\mathbf{x}) \propto f(\mathbf{x})$ with $f(\mathbf{x})$ representing the full integrand) could provide the correct value in a single sample. In practice, the only requirement of a PDF is that it is non-zero wherever the integrand is non-zero in order to represent all potential outputs, but poorly fitting PDFs can in fact cause poor accuracy even with a substantial amount of samples.

Another convenient construct is the *cumulative distribution function* (CDF) which expresses the likelihood of the random variable X falling below a bound b,

$$\Pr[X \le b] = \int_{-\infty}^{b} f_X(x) \,\mathrm{d}x,\tag{7}$$

and directly follows from the definition of the PDF.

Summed-area Tables.

A summed-area table (SAT) can be seen as a form of precomputed integration (indeed, an alternative name for the principle is *integral image*) over an N-dimensional domain. This makes it an especially powerful tool for multidimensional integration given the phenomenon known as the curse of dimensionality, which posits that any analysis handling multidimensional data become less and less efficient as the number of dimensions increase. This is due to an exponential relationships between the number of dimensions and the scale of the data required to produce results. SATs, among others, are an excellent way of reducing the computational complexity of multidimensional integration by *caching* – that is, precomputing and then storing the results - data along an arbitrary number of dimensions (but generally limited to two) such that time complexity is reduced at the cost of increased data storage usage. The major use of SATs in our work comes from the aforementioned histogram encoding used in Part 2 which uses 2D SATs to cache normal distributions distributed in hemispherical histograms.

Given an arbitrary function f(x,y) dependent on the discrete position (x,y) of a texel, the SAT is generated by summing the values of all texels covered by the rectangle formed from the corners (0,0) and (x,y) (see Figure 7). Mathematically, this can be expressed as

$$F(x,y) = \sum_{(i,j) \le (x,y)} f(i,j).$$

Given the definition of a Riemann integral, one can see that this is equivalent, modulo the discretization error, to

$$F(x,y) = \int_{0}^{x} \int_{0}^{y} f(i,j) \,\mathrm{d}j \,\mathrm{d}i$$

and thus SATs are often used to store approximate area integrals. Indeed, one can additionally note that this is analogous to a discrete CDF as per Equation (7), but with the PDF f(i,j) restricted to the domain [(1,1), (N,M)] for an $N \times M$ texture. This exact purpose is leveraged in numerous ways in Part 2, where we use SATs to efficiently store and evaluate integrals of complex normal distribution functions (see Section 1.2.4).

-						
8	9	7	3	5	8	1
9	7	2	7	3	7	1
7	9	9	8	3	9	7
6	8	4	8	9	8	6
1	5	2	2	6	8	4

 (\mathbf{a})

			(4)	-		
8	17	24	27	32	40	41
17	33	42	52	60	75	77
24	49	67	85	96	120	129
30	63	85	111	131	163	178
31	69	93	121	147	187	206
1	1		(b)	1		

Fig. 7. Example SAT procedure. Given an arbitrary 2D array (e.g., a normal map) as in (**a**), the SAT shown in (**b**) is generated by summing all values less than or equal to the cell's index. To compute the value of all cells the blue region, we sum the two extremal regions (in orange and green) and then subtract the two edge regions (in purple and red).

1.1.3. Parametric Equations

The most common way to express a change of variables in a multidimensional integral is as a set of functions relating the new variables to the initial ones. This is also known as a *parametric equation* in a more general context and is often used to describe relationships where a singular function is inappropriate. For instance, the unit circle, which violates the right-definite property of functions (that is, given $f : X \to Y$, $\forall x \in X \ \forall y, z \in Y$: if f(x) = y and f(x) = z, then y = z), is defined by the two equations

$$x = \cos(t)$$
 $y = \sin(t)$

as a function of the polar angle t.

Parametric equations are commonplace in computer graphics thanks to their ability to describe a large variety of curves or surfaces.

Bézier Curves. A Bézier curve is a specific kind of parametric curve with a very simple formulation:

$$\mathbf{B}_{n}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} \mathbf{P}_{i} \qquad \text{where } 0 \le t \le 1,$$
(8)

where *n* is the *degree* of the curve indicating it has n + 1 control points $\{\mathbf{P}_0, \dots, \mathbf{P}_n\}$. Typically, quadratic (n = 2) and cubic (n = 3) curves are preferred, since they balance flexibility and simplicity. Bézier curves also have a few interesting properties:

- The curves always begin at P_0 and end at P_n . This makes them easy to connect and chain.
- The segments $\mathbf{P}_0\mathbf{P}_1$ and $\mathbf{P}_{n-1}\mathbf{P}_n$ are tangent at the end points. Therefore, one can make two curves \mathbf{P} and $\mathbf{Q} C^1$ continuous at $\mathbf{P}_n = \mathbf{Q}_0$ by placing $\mathbf{P}_{n-1}\mathbf{P}_n\mathbf{Q}_1$ on a line.
- The curves are C^{∞} continuous within their domain.
- A curve is always contained by the convex hull of its control points. A convex hull of a set of points **P**_n is defined as the smallest convex polygon which contains all **P**_n.
- It is possible to fit a Bézier curve of degree n onto n + 1 points uniquely, provided each point
 P has an associated location t in the Bézier curve's domain.

C-continuity, also known as *smoothness*, is related to a function's differentiability. Specifically, a function is said to be C^n continuous if its n^{th} -order derivative $\frac{d^n f(x)}{dx^n}$ exists throughout its domain.

Given those properties, we exploited Bézier curves extensively in Part 1 to represent non-linear motion across multiple frames, specifically by using the last point above to fit a quadratic curve onto three points each step.

1.1.4. Frequency Analysis

Frequency analysis is the study of the decomposition of any function into a linear combination of simple trigonometric functions, each representing a certain *frequency*. The fundamental logic

behind this can be seen as a generalization of a Fourier series,

$$f_N(x) = \frac{A_0}{2} + \sum_{n=1}^N A_n \cdot \cos\left(\frac{2\pi nx}{\mathcal{P}} - \varphi_n\right) = \sum_{n=-N}^N c_n \cdot e^{i\frac{2\pi nx}{\mathcal{P}}},\tag{9}$$

with A_n and c_n constant amplitudes for each frequency, \mathcal{P} an integrable interval over the domain of f(x) and N the number of distinct frequencies which compose the function over that interval (with $N \to \infty$ in the general case). A Fourier series allows any function to be expressed as a sum of sinusoidal factors with a given frequency over a certain period (since sinusoidal functions are inherently periodic, then the representation only works for a finite interval unless the function is itself periodic). This can be seen as a variant upon the well-known Taylor series,

$$f(x) = \sum_{n=0}^{\infty} \frac{f^n(a)}{n!} (x-a)^n$$
(10)

with $f^n(a)$ the n^{th} derivative of f(x) evaluated at a. In both cases, taking a finite sum of factors restricts accuracy; Fourier series expansions define a domain within which approximations are expected to be relatively good, while Taylor series use a point a around which the approximation will be best. However, both series can also generate finite expansions for certain functions without approximations (any trigonometric function for Fourier series, any polynomial for Taylor series).

Describing a function as a composition of basic frequencies is an extremely powerful tool. Operations such as integration, which may be intractable on the function itself, can be expressed as a sum of integrable functions. If the functions are only analyzed in a narrow frequency band, the approximation can remain very accurate with a reasonable amount of terms. This particular process is used throughout frequency analysis, for instance in Parts 2 and 3.



Fig. 8. Plot of an example function $f(x) = (x - \pi)^4 - e^{\frac{x-\pi}{2}}$ over $[-\pi,\pi]$ and its Fourier series approximations for $N \in \{3,20\}$. Note how coarse the N = 3 approximation is, whereas higher values are much more refined. Boundaries also tend to be poorly approximated due to lower weighting in the interval. Since Fourier series are periodic, the approximation repeats itself just outside of the bounds of the interval (in the gray region).

Fourier Transform. A Fourier *transform* is a general operation on any function f(x) which produces its equivalent function $\mathcal{F}[f(x)] = \hat{f}(\xi)$ in frequency space,

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i x\xi} \,\mathrm{d}x.$$
(11)

By convention, the "original" domain of the function (here, f(x)) is usually called the *time domain*, even if not actually pertaining to time, while its transform $\hat{f}(\xi)$ is in the *frequency domain*.

A Fourier transform has infinite support and is thus not in and of itself an approximation, unlike Fourier series expansions, as long as the function is integrable. Furthermore, Fourier transforms are invertible (under certain conditions we will not expand upon for brevity), that is to say,

$$f(x) = \mathcal{F}\left[\hat{f}(\xi)\right] = \int_{-\infty}^{\infty} f(\xi)e^{-2\pi i\xi x} d\xi.$$
 (12)

One of the most interesting aspects of using a Fourier transform on a function is the ability to study its frequency characteristics, notably its *bandwidth*. A function's bandwidth is the range of frequencies which it covers, which is not necessarily infinite. These frequencies can also be manipulated, for instance using a bandpass filter which can remove certain frequencies outright (a typical use of which would be blurring an image to remove high frequency noise). A *bandlimit* then is a limit on the bandwidth of a function, effectively imposing that all frequencies beyond a certain range are zero. This is something we exploit in Part 2 to limit the complexity of the lighting calculations we perform according to the maximum frequency of the lighting environment, since it is analogous to the bandlimit seen in spherical harmonics in Section 1.1.5.

Convolution Theorem. A convolution $f(t) \circ g(t)$ is the product of two functions, f(t) and g(t), where one of the two functions is reversed and both are integrated over all possible shifts τ of t,

$$(f \circ g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,\mathrm{d}\tau.$$
(13)

It is akin to "sliding" the function g(-t) (i.e., the reverse of g(t)) over the full domain of f(t) (see Figure 9) and computing the area under the curve of the product of both functions at each point.

A simple example is the convolution with a Dirac delta,

$$\int_{-\infty}^{\infty} \delta(\tau) g(t-\tau) \,\mathrm{d}\tau = g(t),$$

since, for any *t*, the integral is non-zero only when $\tau = 0$. This can be imagined as "sliding" the function g(t) along the *t* axis which is windowed by the Dirac delta.

The Fourier transform interacts with convolution in a very elegant way: as per the convolution theorem, the convolution of two functions in the time domain is proportional to their multiplication



Fig. 9. Example convolution of functions $f(x) = 2^{2-x/2} - 1$ and g(x) = 1 within $x \in [0,4]$ and zero elsewhere. Function g(-x) "slides" from the left to the right over the entire domain, computing the multiplied area under the curve of both functions.

in the frequency domain, i.e.

$$h(x) = (f \circ g)(x) \qquad \leftrightarrow \qquad \hat{h}(\xi) = k\hat{f}(\xi) \cdot \hat{g}(\xi), \tag{14}$$

where k is a constant dependent on the selected Fourier transform convention and its normalization factor.

Discrete Fourier Transform. The Discrete Fourier Transform (DFT) is an adaptation of the Fourier Transform to discrete (as opposed to continuous) inputs. This is of great value for numerical analysis, given that any function can be sampled densely and then those samples transformed. It takes the form

$$X_{k} = \sum_{n=0}^{N-1} x_{n} \cdot e^{-\frac{i2\pi}{N}kn}$$
(15)

with x_n and X_k both sequences of complex numbers of a chosen length N in the time and frequency domains, respectively. Thanks to an algorithm known as the Fast Fourier Transform (FFT), it is possible to efficiently compute DFTs on commodity computer hardware, for instance to determine the convolution of two discrete sequences.

By representing continuous signals with discrete samples and transforming them with an FFT, we can cheaply compute the signal bandwidth to determine when to terminate our search through our acceleration structures in Part 3.

1.1.5. Spherical Harmonics

Combining spherical coordinates with frequency analysis additionally enables us to use a very powerful mathematical construct: *spherical harmonics* (SH), which can represent arbitrary functions on the surface of the sphere. They are a natural extension of Fourier series, but applied in the spherical domain, with many of the same interesting characteristics. Defined as the eigenfunctions of the 3D angular Laplacian on the sphere, they are generally represented as

$$Y_{\ell}^{m}(\theta,\varphi) = \sqrt{\frac{2\ell+1}{4\pi} \frac{(\ell-m)!}{(\ell+m)!}} P_{\ell}^{m}(\cos\theta) e^{im\varphi}, \tag{16}$$

though multiple normalization factors exist depending on applications. The $P_{\ell}^{m}(\cos \theta)$ in the above formulation are the associated Legendre polynomials, which take the form

$$P_{\ell}^{m}(x) = \frac{(-1)^{m}}{2^{\ell} \ell!} \left(1 - x^{2}\right)^{m/2} \frac{\mathrm{d}^{\ell+m}}{\mathrm{d}x^{\ell+m}} \left(x^{2} - 1\right)^{\ell}.$$
(17)

Of particular note is that, much like Fourier series, given a function $f(\theta, \varphi)$ defined on the surface of the sphere, it is possible to find constant factors f_{ℓ}^m which allow a representation as a sum of SH basis functions

$$f(r,\theta,\varphi) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} f_{\ell}^{m} Y_{\ell}^{m}(\theta,\varphi).$$
(18)

This is possible because individual SH basis functions are orthonormal, that is to say,

$$\int_{\Omega} Y_{\ell}^{m}(\omega) Y_{\ell'}^{m'^{*}}(\omega) d\omega = \delta_{\ell \ell'} \delta_{mm'} \qquad \qquad \int_{\Omega} |Y_{\ell}^{m}(\omega)|^{2} d\omega = 1$$
(19)

with

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}$$

the Kronecker delta and the normalization criterion being fulfilled with the specific normalization factor chosen in Equation (16).

A spherical function $f(r,\theta,\varphi)$ is said to be *bandlimited* if, in the formulation of Equation (18), the factors $f_{\ell}^{m} = 0 \forall \ell \geq L - 1$ for some bandlimit *L*. This can be either a property of the function or as an approximation by taking all $|f_{\ell}^{m}| < \varepsilon$ as approximately zero for a small ε and determining the associated *L* from the ℓ coefficient. This notably allows for a similar process as described in Section 1.1.4, where we bandlimit *frequencies* rather than SH terms in order to constrain the complexity of the function we analyze.



Fig. 10. Plots of the absolute real part of the spherical harmonics basis functions $Y_{\ell}^{m}(\theta, \varphi)$ for some sample ℓ and m. Orange parts are positive and blue parts are negative, while the value of $Y_{\ell}^{m}(\theta, \varphi)$ is used as the radius for visualization purposes only.

Integrating the product of two spherical functions can be done through their spherical harmonics basis representation,

$$\int_{\Omega} f(\omega)g(\omega) d\omega = \int_{\Omega} \left(\sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} f_{\ell}^{m} Y_{\ell}^{m}(\theta, \varphi) \right) \left(\sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} g_{\ell}^{m} Y_{\ell}^{m}(\theta, \varphi) \right) d\omega$$
$$= \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} f_{\ell}^{m} g_{\ell}^{m} \iint_{\Omega} Y_{\ell}^{m}(\theta, \varphi) Y_{\ell}^{m}(\theta, \varphi) \sin \theta \, d\theta \, d\varphi \quad \text{by Equation (19) (left)}$$
$$= \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} f_{\ell}^{m} g_{\ell}^{m} \quad \text{by Equation (19) (right)} \tag{20}$$

which can be combined with the bandlimited formulation to quickly compute bandlimited products,

$$\int_{\Omega} f(\boldsymbol{\omega})g(\boldsymbol{\omega}) \,\mathrm{d}\boldsymbol{\omega} = \sum_{\ell=0}^{L-1} \sum_{m=-\ell}^{\ell} f_{\ell}^{m} g_{\ell}^{m}.$$
(21)

Bandlimiting is an important attribute of spherical harmonics and one of the decisive factors in our use of them in Part 2, since they allow us to dynamically scale the level of detail (and its associated memory usage requirements) according to the frequency of the light source in play.

Zonal Harmonics. Zonal harmonics are spherical harmonics which are invariant under rotation around a specified axis. For the case of the 3-dimensional sphere, the zonal harmonics are a simplification of the general spherical harmonics,

$$Z^{\ell}(\theta, \varphi) = P_{\ell}(\cos \theta), \qquad (22)$$

which can be seen as equal to $Y_{\ell}^{0}(\theta, \varphi)$ without the normalization constant. Here, $P_{\ell}(\cos \theta)$ are the Legendre polynomials (as opposed to the *associated* Legendre polynomials), which again can be seen as a simplification with m = 0,

$$P_{\ell}(x) = \frac{1}{2^{\ell} \ell!} \frac{d^{\ell}}{dx^{\ell}} \left(x^2 - 1\right)^{\ell}.$$
(23)

Basic zonal harmonics always use the \hat{z} axis for their invariant, but these can easily be rotated and are then represented as

$$Z_{\mathbf{x}}^{\ell}(\mathbf{y}) = Z^{\ell}(\mathbf{R}\mathbf{y}) \tag{24}$$

where x is the new invariant axis and R is an orthogonal transformation that reorients x as \hat{z} .

Through the use of the Funk-Hecke convolution theorem in conjunction with zonal harmonics in Part 2, we leverage an algorithm to efficiently rotate spherical harmonics (Sloan et al. [93]).

1.2. Light Transport

Light transport is the set of fundamental mechanics under which light, emitted by sources like lightbulbs or the sun, can reach a sensor, such as your eyes or a camera. It is also at the core of all of rendering and much of computer graphics research. We will now briefly look at core concepts before digging into specifics related to the articles found in this thesis.

1.2.1. Light Transport Fundamentals and the Rendering Equation

The rendering equation (Equation (1)) is the main building block of light transport: the total radiance from a surface at point **x** in direction ω_0 for wavelength λ is given by the sum of (**a**) the surface's own *emitted* radiance L_e and (**b**) the reflected *incoming* radiance L_i modulated by the *bidirectional reflectance distribution function* (BRDF) f_r for all incoming directions ω_i around the normal hemisphere Ω at the surface. This equation is also recursive, since $L_i(\mathbf{x}, \omega_i, \lambda) = L_o(\mathbf{x}', -\omega_i, \lambda)$ for some other location \mathbf{x}' in the scene.

Proper understanding of this equation requires defining a few radiometric concepts, the study of radiant energy transfer.

Radiant energy is the energy of electromagnetic radiation such as light, usually denoted Q (the e subscript used to distinguish with photometric quantities will be omitted for simplicity).

Radiant power or *radiant flux* (Φ) is the radiant energy per unit of time which is emitted, reflected or transmitted for a given system, usually a surface.

Radiance (L) is the most important and common measure in computer graphics: it is the radiant power at some point x travelling in a direction ω , per surface unit area dA and per unit solid angle $d\Omega$ along the direction.

Irradiance (*E*) is the total radiant power per unit area *incident* to a surface, which can be computed from incoming radiance L_i as

$$E = \frac{\mathrm{d}\Phi}{\mathrm{d}A} = \int_{\Omega} L_i(\mathbf{x}, \omega_i) \cos\theta \,\mathrm{d}\omega_i. \tag{25}$$

Finally, *radiosity* (*B*) can be seen as the opposite of irradiance: it is the total radiant power per unit area *exitant* to a surface:

$$B = \frac{\mathrm{d}\Phi}{\mathrm{d}A} = \int_{\Omega} L_o(\mathbf{x}, \boldsymbol{\omega}_o) \cos \theta \,\mathrm{d}\boldsymbol{\omega}_o. \tag{26}$$

Note that all prior concepts can additionally be qualified as *spectral*, which adds a dependency on a specific wavelength of light rather than covering all possible wavelengths.

While the rendering equation is, in and of itself, infinitely recursive, we generally desire to solve it for a specific location such as a camera. In practice, this means solving the integral

$$L_{c}(\mathbf{x},\boldsymbol{\lambda}) = \int_{\Omega} T(\mathbf{x},\boldsymbol{\omega}_{i},\boldsymbol{\lambda}) L_{i}(\mathbf{x},\boldsymbol{\omega}_{i},\boldsymbol{\lambda}) (\boldsymbol{\omega}_{i}\cdot\boldsymbol{\omega}_{n}) \,\mathrm{d}\boldsymbol{\omega}_{i}, \qquad (27)$$

where $T(\mathbf{x}, \omega_i, \lambda)$ is a *transfer function* describing details of how a camera processes incoming light as it hits the sensor and $L_i(\mathbf{x}, \omega_i, \lambda)$ is the incoming light described by the rendering equation.

Since this integral is impossible to compute analytically, it is usually converted into an approximate form using Monte Carlo integration (see Section 1.1.2) as per Equation (6). As a result, we get the approximation

$$L_{c}(\mathbf{x},\lambda) \approx \frac{1}{N} \sum_{n=1}^{N} \frac{1}{p(\omega_{in})} T(\mathbf{x},\omega_{in},\lambda) L_{i}(\mathbf{x},\omega_{in},\lambda) (\omega_{in}\cdot\omega_{n}), \qquad (28)$$

where $p(\omega_{in})$ is the PDF and the integrand is summed over for a large number N of samples.

1.2.2. Foundational Rendering Techniques

Broadly speaking, modern graphics rendering can be divided in two categories: *rasterization*-based approaches, used mainly in games, and *ray tracing*-based approaches, used mainly in film and television.





Fig. 11. Visualisation of the primary radiometric quantites. (a) focuses on a single photon p whose radiant energy is denoted Q. (b) shows the emitted radiant energy per unit time t, also known as radiant power, of the whole light source. (c) and (d) show related values, that being the incident (irradiance) and exitant (radiosity) sum total of radiant power per unit area dA. Finally, (e) describes the radiance at point \mathbf{x} of unit area dA exiting the surface in direction $\boldsymbol{\omega}$ and covering the solid angle $d\Omega$.

Rasterization takes vector graphics such as polygons, lines or points, and converts them into an image made up of discrete pixels. Each pixel can have arbitrary computations applied to it according to the data inputs available, for example pertaining to the point which was projected onto the pixel.



Fig. 12. Rasterization overview for the three-dimensional case. An arbitrary triangle is first projected (a) onto the screen plane, and then each pixel is tested (b) against the triangle for overlap to determine if it is covered by it.

Ray tracing takes the opposite approach: given a camera and image plane, rays are sent from the camera's origin through the plane and into the scene, where they intersect with objects to provide information for pixel processing.



Fig. 13. Ray tracing overview for the three-dimensional case. An arbitrary ray is first projected from the image plane into the scene, and then its corresponding pixel is processed if the ray has hit some geometry.

In both cases, the processing which happens on a pixel is typically called *shading*. It is where the BSDF may be evaluated, for instance, using surface and view information such as the incoming/outgoing vectors, the surface normal or albedo, etc.

1.2.3. Global Light Transport

Global light transport, often called *global illumination*, is a concept which derives directly from the rendering equation. Early computer graphics had a tendency to cut the recursive nature of the equation beyond its first or second level to reduce the exponentially increasing cost of computing it, which created something we now call *local illumination*. Under this model, objects are lit only by light directly coming from a light source, e.g., sunlight. This creates perfectly black shadow regions wherever all lights are blocked, since indirect connections (i.e., light rays bouncing on non-emissive surfaces more than once before reaching the camera or light) are not modelled.

To better describe these paths, let us introduce *path notation* [42], which uses a simple regular expression-like structure to define paths using key letters:

- L indicates the path has reached a light source. Depending on how the path is traced, this is usually the start or end of the path.
- E indicates the path has reached the eye (or camera). This will be the other endpoint of the path.
- D indicates a *diffuse* bounce, e.g., materials like raw wood, concrete, etc. Light has a tendency to scatter in many directions at random.
- S indicates a *specular* bounce, e.g., materials like metals, shiny plastics, etc. Light will bounce in a narrower range of directions.

Direct lighting, as described above, would have a path of the form L(D|S)?E, where the | (pipe) operator indicates a choice (either D or S) and the ? (question mark) operator indicates that the group or element before it is optional.

By contrast, a path which is not in any way restricted and can follow the rendering equation until it hits a natural terminator would have the path L(D|S)*E, where the * (star) operator indicates zero or more repetitions of the prior group.



Fig. 14. Some simple path examples, here drawn with the light source as the start of the path, and their respective path notation.

This path formulation is a natural fit for ray tracing, which then becomes *path tracing*, since it follows a path made up of multiple consecutive rays. However, rasterization has no such simple extension, which has given rise to many approximations and hybrid techniques.

Irradiance Probes. One such technique applies strictly for LD*E paths, since diffuse bounces can effectively be computed ahead of time with no knowledge of the final direction. This allows the



(a) Location of the probes, drawn as colored spheres, in the scene.



(b) Information stored by each probe, here laid out in a single data structure.

Fig. 15. Example use of irradiance probes, here from Part 4. In this case, the probes are automatically laid out in a grid, but manual placement is possible and often desired. Each probe stores the irradiance of all visible objects from the probe's location. Note in this image that, since irradiance probes can only effectively deal with diffuse materials, the glasses on the tables do not exhibit the correct lighting, which is a limitation of the technique from Majercik et al. [63].

diffuse information to be stored in *probes*, 2D tables which collect radiance for a discrete number of directions at a certain point in space. Using a sufficiently large number of these probes, diffuse global illumination can be approximated (see Figure 15).

Environment Maps. Another technique is used for distant light sources or complex environments which are not fully modeled in the scene. In those situations, rather than painstakingly create the full geometry and render it, a picture of the environment can be taken over the entire hemisphere around a given central location (e.g., see Figure 16). This picture can then be used to approximate lighting by projecting it at infinity and sampling it later. Such environment maps can be used for diffuse and



Fig. 16. Example environment map, here of a pier at sunset. The severe distortions exhibited are due to the reprojection of a spherical map onto a 2D rectangle. CREDIT: TEXTURIFY.COM

specular paths, and are one of the simplest ways to simulate lights with physical dimensions (rather than point lights).

1.2.4. Local Light Transport and the Microfacet Model

We have briefly mentioned the notion of BRDF in Section 1.2.1, considering it for the purpose of introduction as a black box which regulates the behavior of a surface under given lighting conditions. There exist numerous models for such functions, but one of the most popular is actually a family of algorithms all deriving from the *microfacet model*, first introduced by Torrance and Sparrow [100] and refined by Cook and Torrance [19].

The microfacet model assumes that surfaces are composed of tiny mirrors (facets) arranged in various directions. The material's macroscopic properties are dependent on the distribution and properties of those facets. For instance, a rough surface would be composed of mirrors in a wide range of directions, distributing reflected light in all directions and thus creating a duller appearance. In contrast, a perfect mirror's facets would all be aligned in the same direction, effectively becoming one large continuous surface.

Under this model, a BRDF is defined as a multiplication of terms, specifically

$$f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \boldsymbol{\lambda}) = \frac{F(\boldsymbol{\omega}_i, \boldsymbol{\omega}_h) G(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \boldsymbol{\omega}_h) D(\boldsymbol{\omega}_h)}{4(\boldsymbol{\omega}_n \cdot \boldsymbol{\omega}_i)(\boldsymbol{\omega}_n \cdot \boldsymbol{\omega}_o)}$$
(29)

where all variables are taken from the definition of the rendering equation (see Equation (1)), and the additional terms are:

• $F(\omega_i, \omega_h)$ is the Fresnel function and handles the material's behavior from the Fresnel equations, which define the material's *reflectance* (the effectiveness of a material at reflecting radiant energy, see Section 1.2.1).

- $G(\omega_i, \omega_o, \omega_h)$ is the shadow-masking term, which models the impact of neighboring facets in terms of shadowing (i.e., neighboring facets blocking incoming light from reaching a facet) and masking (i.e., neighboring facets blocking outgoing light from leaving the surface).
- $D(\omega_h)$ is the normal distribution function (NDF), which provides a statistical distribution of the microfacets' orientation vis-à-vis a given half-vector.



(a) Unobstructed microfacet (b) Shadowed microfacet (c) Masked microfacet

Fig. 17. 2D projection of a microfacet model, showing the various microscopic mirrors, or facets, reflecting light. In (**a**), the light rays are completely unobscured. In (**b**), some rays are blocked from reaching the facet by a close-by facet. In (**c**), the reverse happens, and a neighboring facet prevents the light reflected by the facet from leaving the surface.

Surface Scattering and Transmittance of Light. While we previously described the BRDF as a fundamental black box which describes the properties of a surface and its interaction with light, it is actually but a subset of a class of functions known as bidirectional *scattering* distribution functions (BSDFs). Specifically, a BRDF makes two assumptions:

- Light exits the surface from the same point it entered.
- Light does not traverse the material.

Removing the first assumption gives us bidirectional *scattering surface* reflectance distribution functions (BSSRDFs)¹, which take into account separate input and output locations, $f_{ssr}(\mathbf{x}_i, \mathbf{x}_o, \omega_i, \omega_o, \lambda)$.

Removing the second assumption gives us bidirectional *transmittance* distribution functions (BTDFs), which model thin surfaces through which light can actually pass and exit on the other side. The thin surface requirement is implied by the fact the light still exits from the same location it entered, limiting the thickness to an infinitesimal distance.

Removing both gives us bidirectional *scattering surface transmittance* distribution functions (BSSTDFs), which can model surfaces of any thickness.

¹BSSRDFs can also be called "bidirectional *subsurface scattering* reflectance distribution functions" or even "bidirectional *scattering surface* reflectance distribution functions"; we chose this particular formulation because it appears to have become the dominant terminology in recent years.

Together, all four of these functions form a BSDF, though obviously certain materials will only exhibit some properties (e.g., certain materials may not be translucent, thus eliminating any transmittance component).



Fig. 18. Illustration of the four main types of BSDFs. (a) exhibits the simple case of the BRDF (with no restriction on the material's thickness, d), while (b) extends it to a BSSRDF by allowing a random walk through the material which must now have a real thickness d > 0. (c) and (d) are similar, but tackle transmittance through the surface, with (c) also explicitly restricting thickness d to an infinitesimal layer. A BSDF is a combination of any (or all) of the above.

1.3. Distribution Effects from Realistic Camera Models

Distribution effects, as mentioned in the Introduction, were the starting point of this thesis. Being an integral part of light transport, often with a significant impact on the frequency distribution of signals affected by one such effect, they neatly combine all of the topics introduced so far into a single subject with far-reaching applications for ever-increasing photorealism.

1.3.1. Idealized Camera Light Transport

Most rendering, even today, works off a fairly simple model known as a *pinhole camera*.

Under this model, objects are all visible and in focus. In addition, we typically make the assumption that all objects are frozen in time, as though each frame was rendered from an instantaneous snapshot of the scene. These characteristics form the basis of the simplest idealized camera model.



Fig. 19. Simple diagram of a pinhole camera. Objects reflect light into the pinhole, more generally known as aperture, which crosses through to the image plane some distance away, while other rays are blocked. The resulting image is rotated 180 degrees around the normal at the image plane, but otherwise replicates what an observer would see at the pinhole's location.

1.3.2. Depth of Field

While the pinhole camera model is practical for some scenarios, it is not realistic, since typical observers such as cameras or our own eyes have a certain dimension to their *aperture*, the hole through which light may hit the image plane. As a result, more complex models attempt to replicate effects caused by a non-infinitesimal aperture, known as *depth of field*, the most popular of which is the *thin lens model*.

The thin lens model is fundamentally based on the thin lens equation,

$$\frac{1}{o} + \frac{1}{i} = \frac{1}{f} \tag{30}$$

where o is the distance of an object from the lens, i is the distance of its image from the lens, and f is the focal length of the lens. Given an image plane some distance i_p behind the lens, one can determine the distance o_p at which an object must be from the lens to be replicated perfectly on the image plane, which is typically known as being *in focus* and the plane at distance o_p is known as the *focal plane*.

It is also fairly straightforward to determine the diameter of this circular blur *c*, known as *circle* of confusion,

$$c = A \frac{\left| o - o_p \right|}{o} \frac{f}{o_p - f},\tag{31}$$

where o is the distance between the lens and the object, o_p is the focal plane distance, f is the focal length of the lens, and A is its aperture diameter.



(a) In focus



(b) Out of focus

Fig. 20. Simple diagram of a thin lens camera. Unlike a pinhole camera, rays can come through an aperture of a given dimension (the lens), at which point they are redirected through a lens in order to hit the image plane at the correct location. Per the thin lens equation, objects at the focal plane o_p will be replicated much as if they were seen in a pinhole camera, as shown in (**a**), but objects at a distance $o \neq o_p$ closer or farther away will create an image which is in front of or behind the actual image plane, creating blur, as exhibited in (**b**).

Simulating this thin lens model is extremely expensive, since it involves adding two extra dimensions to the integration problem. Taking Equation (27), we must transform it into

$$L_{cd}(\mathbf{x},\lambda) = \int_{A} \int_{\Omega} T(\mathbf{x},\omega_i,\lambda) L_i(\mathbf{x},\omega_i,\lambda)(\omega_i\cdot\omega_n) \,\mathrm{d}\omega_i \,\mathrm{d}\mathbf{a}, \tag{32}$$



Fig. 21. Circle of confusion diagram. o is the distance between the lens and the object, o_p is the focal plane distance, f is the focal length of the lens, and A is its aperture diameter. The dotted blue lines represent an object "at infinity", which, as per the lens equation, focuses at the focal length f.

where A is the 2D region of the camera's aperture and $d\mathbf{a}$ its infinitesimal section. This can be naively computed as

$$L_{cd}(\mathbf{x},\lambda) \approx \frac{1}{N} \sum_{n=1}^{N} \frac{1}{p(\mathbf{a}_n, \boldsymbol{\omega}_{in})} T(\mathbf{x} + \mathbf{a}_n, \boldsymbol{\omega}_{in}, \lambda) L_i(\mathbf{x} + \mathbf{a}_n, \boldsymbol{\omega}_{in}, \lambda) (\boldsymbol{\omega}_{in} \cdot \boldsymbol{\omega}_n),$$
(33)

where \mathbf{a}_n is a 2D location on the lens within the bounds of the aperture. This can alternatively be computed using sums of sums,

$$L_{cd}(\mathbf{x},\lambda) \approx \frac{1}{NM} \sum_{m=1}^{M} \frac{1}{p(\mathbf{a}_m)} \sum_{n=1}^{N} \frac{1}{p(\omega_{in})} T(\mathbf{x} + \mathbf{a}_m, \omega_{in}, \lambda) L_i(\mathbf{x} + \mathbf{a}_m, \omega_{in}, \lambda) (\omega_{in} \cdot \omega_n), \quad (34)$$

where *M* is another number of samples \mathbf{a}_m independent from the *N* ω_{in} samples. As a result of this interpretation, depth of field is often brute force computed as *M* pinhole renders with shifted origins.

1.3.3. Motion Blur

In addition to the aforementioned thin lens model, there exists another common effect which introduces an additional dimension to the rendering equation.

Most computer graphics these days are used for videos rather than still images, making time a significant factor in rendering. The simplest approach is to simply consider every frame of a video as an instantaneous snapshot of a scene at some point in time, which allows the rendering to proceed as though it were a still image. When enough of these frames are stitched together for a short enough duration, the appearance of motion is achieved.

However, the instantaneous nature of each frame is incorrect. A real camera must open and close its aperture using a *shutter*, which takes time. During this time, parts or the totality of the



(a) Near focus



(b) Far focus

Fig. 22. Example computer-generated images with differing focal distances. In (**a**), the focus is set close to the camera, making nearby objects sharp, but both distant and very close objects blurry. In (**b**), the focus is set farther, causing the previously sharp objects to become blurry. Also note in this image that the circular shape of the camera's aperture is visible as a slightly brighter circle on the foreground cup. CREDITS: MORGAN MCGUIRE [**67**] AND BLEND SWAP USER WIG42.

scene are visible. This introduces another dimension to our integration: time. By convention, every frame is scaled over the range $t \in [0, 1]$, regardless of the actual duration of the exposure in seconds. This *t* variable is combined with a frame index to determine the absolute point in time since the beginning of the sequence.

Taking once again Equation (27), we must transform it into

$$L_{ct}(\mathbf{x}, \lambda) = \int_{0}^{1} \int_{\Omega} W(\mathbf{x}, t) L_{i}(\mathbf{x}, t, \omega_{i}, \lambda) (\omega_{i} \cdot \omega_{n}) d\omega_{i} dt, \qquad (35)$$

where $W(\mathbf{x}, t, \lambda)$ is the windowing function at time *t* and location **x** on the sensor. A "perfect" windowing function would be simply

$$W(t) = \begin{cases} 1 & 0 \le t \le 1 \\ 0 & \text{elsewhere} \end{cases},$$

which instantly exposes or masks the entire sensor at the start and end of the frame, respectively. However, cameras have imperfect shutters, such as the common digital rolling shutter, which can also be represented in various ways.

Much like in Section 1.3.2, this modified rendering equation can be reinterpreted as a Monte Carlo process in a straightforward fashion,

$$L_{ct}(\mathbf{x},\lambda) \approx \frac{1}{NM} \sum_{m=1}^{M} \frac{1}{p(t_m)} \sum_{n=1}^{N} \frac{1}{p(\omega_{in})} W(\mathbf{x},t_m) L_i(\mathbf{x},t_m,\omega_{in},\lambda) (\omega_{in}\cdot\omega_n).$$
(36)

Motion Vectors. A compact approach to storing a limited amount of motion information is to generate *motion vectors*. Each pixel on the screen is projected to its previous location in view space, and this apparent 2D movement is stored as a vector (see Figure 24). This method is particularly popular in games, since it is lightweight to store and fast to generate, and it can additionally be used for other purposes such as temporal filtering. Unfortunately, it also severely restricts what data is available about the motion of the scene, most notably by only allowing linear motion and only storing a single instance of motion per pixel, discarding information about potential background objects in motion.

1.4. High-performance Hardware Graphics Pipeline

While graphics rendering began on general-purpose hardware, it rapidly became clear that the specific needs of graphics, such as massively parallel computations for millions of picture elements (pixels), warranted a more specialized hardware platform. Thus was born the *graphics processing unit*, or GPU, which is designed specifically for the purpose of rendering images. Historically, this was even narrower, using rasterization to render triangle geometry, but the advent of programmable GPUs has ushered in a new era of flexibility which has flourished since.

Even so, GPUs still have a basic fixed pipeline for most rendering tasks (see Figure 25), which consists of a series of *stages*:

(1) The Input Assembler stage is largely fixed in functionality and transforms the basic mesh input given to the GPU into workable geometry, such as a list of lines or triangles.



Fig. 23. Simple motion blur example. (a) shows a moving circle in three snapshots $t \in \{0, \frac{1}{2}, 1\}$ while (b) shows the rendered blur for the same motion.

- (2) The Vertex Shader stage is the first programmable stage and operates directly on individual vertices. This is where steps such as coordinate transforms or projections can take place.
- (3) The Hull Shader, Tessellator and Domain Shader stages are collectively known as the Tessellation stages and specifically have to do with hardware-accelerated tessellation, which allows the GPU to evaluate lower density meshes as input and create higher density meshes as output given data on how to break down the larger polygons into smaller ones. This can be used to add details and improve final appearance in a scalable way. Both the Hull Shader and Domain Shader stages are programmable and can be used to guide or modify the behavior of the Tessellator. All three stages are optional.
- (4) The Geometry Shader is another programmable stage which works on geometry units such as lines or triangles. It, too, is optional. In addition to having access to more information and working on a higher level than the Vertex Shader, the Geometry Shader can also emit new geometry, or equivalently not pass through all the geometry it was given as input. This gives it great flexibility for things like text rendering or procedural geometry.
- (5) The Stream Output stage is unique in that it can be used to save the results of the pipeline so far back to memory before it is rasterized. It is optional.



Fig. 24. Simple motion vectors interpretation. Each pixel contains (x,y) velocity for that pixel's motion since the previous frame, here represented as arrows for convenience. This example can be interpreted as follows: the blue circle is moving towards the bottom right corner of the screen, while the orange rectangle is spinning counter-clockwise while facing the camera. All motion stored here is purely linear, a notable approximation of this method.

- (6) The Rasterizer stage is fixed-function, but has a lot of configuration options to determine how the final geometry is rasterized, such as how vertices are transformed, parsed and in which order they are drawn.
- (7) The Pixel Shader is the last programmable stage and the most commonly used. After rasterization has been performed, the resulting pixels contain a lot of potential information interpolated from the geometry which can be used by the Pixel Shader to compute all sorts of final outputs. The most obvious use is to generate a final rendered image from a scene, using information such as surface color, material properties, etc.
- (8) The Output Merger stage takes the pixels processed by the prior stage and outputs them to the output texture, performing some final steps such as blending if necessary.

Specialized Hardware Modules. In addition, more recent graphics hardware have added specialized components which accelerate more niche tasks in a drastically more efficient manner than would be possible using general-purpose hardware, even on GPUs. Examples include:

- Hardware-accelerated video encoding and decoding, which implements specific video algorithms directly on chips to allow fast, low-power and cheap processing of compressed video footage.
- Hardware-accelerated ray tracing, which usually accelerates traversal through a scene hierarchy and ray-triangle intersections.
- Hardware-accelerated neural network computations, which can accelerate certain common inference operations.



Fig. 25. A typical hardware graphics pipeline, here using Microsoft Direct3D terminology. Blue rectangular stages are fixed-function, which is to say they have only a limited ability to be configured or modified and cannot be directly programmed. Yellow rounded rectangles indicate programmable stages, whose behavior can be completely user-defined within some bounds. Dashed borders indicate optional stages or pathways which will not necessarily be executed unless requested. Compute shaders are typically not part of the strict graphics pipeline, but are now commonly used to replace any programmable stage as desired.

A Note on Shaders. Throughout this pipeline, the word "shader" has been misused quite a few times. As per the definition given in Section 1.2.2, shaders are code which affect how a surface is shaded. While this was true historically, its use has been broadened to simply mean a piece of code which acts upon a stage in a graphics pipeline. Some shaders, notably *compute shaders*, do not even necessarily process actual graphics data, having appeared with the advent of truly programmable GPUs capable of doing arbitrary work. In this way, we have almost come full circle, with GPUs edging ever closer to the general-purpose nature of CPUs.

First Article.

High Performance Non-linear Motion Blur

by

Jean-Philippe Guertin¹, and Derek Nowrouzezahrai¹

(1) Université de Montréal

This paper was presented at the *Eurographics Symposium on Rendering* on June 26, 2015 in Darmstadt, Germany, as part of the EI&I (*Experimental Ideas & Implementations*) track. Formatting has been adapted for this thesis and minor typographical issues were corrected.

I am the primary author of this article and also presented it at the conference. My contributions are:

- Elaboration of the fundamental algorithm.
- Implementation of the algorithm in a toy renderer which was designed to represent a simple "standard" real-time renderer to showcase ease of integration.
- Validation of the results, including the creation of the various examples and scenes.

Derek Nowrouzezahrai assisted with polishing the algorithm and supervised the work.

RÉSUMÉ. Le flou de mouvement devient de plus en plus commun dans des applications tels les jeux vidéo ou les outils de prévisualisation. Dans de tels cas, une stratégie commune est d'approximer le flou de mouvement avec un post-processus en espace image, et plusieurs approches récentes démontrent des résultats très efficaces et de haute qualité [Sou13,GMN14]. Malheureusement, toutes ces approches assument un mouvement sous-jacent purement linéaire, et donc ne peuvent approximer un mouvement non linéaire sans causer des artefacts visuels sévères. Nous présentons un nouveau post-processus de flou de mouvement qui traite le cas du mouvement non linéaire (en plus du mouvement linéaire) correctement en utilisant une approche de dispersion basée sur l'échantillonnage de courbes. Nous simulons un flou de mouvement non linéaire plausible en 4ms à 1920×1080 , et notre approche démontre plusieurs propriétés souhaitables : son coût est indépendant de la complexité géométrique de la scène, elle estime de manière robuste l'étendue du flou de mouvement pour éviter les artefacts communs de sous- ou sur-floutage, elle support des déplacements de magnitude arbitrairement grande, et elle est moins bruitée que les techniques existantes. **Mots clés :** flou de mouvement, rendu en temps réel

ABSTRACT. Motion blur is becoming more common in interactive applications such as games and previsualization tools. Here, a common strategy is to approximate motion blur with an imagespace post-process, and many recent approaches demonstrate very efficient and high-quality results [Sou13,GMN14]. Unfortunately, all such approaches assume underlying linear motion, and so they cannot approximate non-linear motion blur effects without significant visual artifacts. We present a new motion blur post-process that correctly treats the case of non-linear motion (in addition to linear motion) using an efficient curve-sampling scatter approach. We simulate plausible non-linear motion blur in 4ms at 1920×1080 and our approach has many desirable properties: its cost is independent of geometric complexity, it robustly estimates blurring extents to avoid typical over- and under-blurring artifacts, it supports unlimited motion magnitudes, and it is less noisy than existing techniques. **Keywords:** motion blur, real-time rendering



Fig. 26. Smooth motion blur on a variety of complex, potentially non-linear motions, computed in 3.5 to 6.5 ms at 1920×1080 .

1. Introduction

Motion blur effects give important visual cues about the dynamics of a scene, and as such they have played an almost essential role in realistic image synthesis for visual effects. More recently, the development of high-performance post-processing techniques for approximating motion blur have led to their almost ubiquitous integration in interactive graphics applications, such as video games.

Despite these recent advances in more efficient and realistic motion blur simulation, almost every existing motion blur solution (including the majority of offline, high-fidelity solutions) assume that the underlying motion of an object is *strictly linear*. This assumption dramatically reduces the complexity of simulating motion blur effects, and is particularly important for high-performance approximations that rely on image-space post-processing.

While, in practice, this limitation can sometimes be disguised by either cleverly crafting an animation sequence, or limiting the virtual exposure to short bursts, it can still lead to very distracting visual artifacts. Avoiding these visual artifacts becomes even harder with the state of the art in interactive motion blur approximations that rely primarily on image-space post-processing. Here, camera and object motion can both very easily combine to cause very jarring visual artifacts, even in scenes with simple motions (e.g., Figure 26).

We present a high-performance motion blur approximation that gracefully handles linear and non-linear motion, supports long exposure times, does not introduce temporal artifacts under camera motion, and maintains the same advantages of existing techniques: it scales independently with the underlying scene/motion complexity, and it uses a simple post-process that integrates easily into existing engines.



Fig. 27. Visualizing scatter and gather operations. In scatter-based algorithms, each data point (i.e., pixel) deposits data onto neighboring points; for gather-based algorithms, each point queries its neighbors to compute its final value. Scatter-as-gather emulates the former using the latter.

2. Previous Work

We present recent work most related to our approach below, and we forward readers to the comprehensive survey on motion blur (Navarro et al. [74]) for a more complete view of the area.

Offline Sampling. Traditionally, motion blur (and other distribution effects) can be estimated via numerical integration, as presented by Cook in his seminal 1986 work on the topic [**20**]. Recent work on these offline solutions design more elaborate sampling and filtering schemes capable of leveraging the structure of object motion, including multi-dimensional sampling schemes (Hachisuka et al. [**40**]), or adaptive sampling schemes based on wavelet-space (Overbeck et al. [**80**]) or frequency-based (Egan et al. [**33**]) formulations of the motion blur problem. Adaptive sampling can also be combined with anisotropic spatial-temporal filtering (Lehtinen et al. [**61**]).

We are motivated by Reeves 1983's approach [85], where motion points are advected according to a (world-space) particle system to form motion segments, and a world-space blur is applied to the segments in order to approximate motion blur effects. We instead sample points on a screen-aligned grid and analytically fit motion curves, leveraging the entire programmable rasterization pipeline to efficiently implement a true motion blurring scatter operation.

We target interactive applications, where object-space sampling is not an option and motion blur effects must be computed on the order of milliseconds, not seconds/minutes.

Interactive Approximations. Apart from heuristic object- or texture-space extrusion and sorting approaches (Max and Lerner [65], Tatarchuk et al. [97], Ritchie et al. [87]), many interactive

solutions aim to approximate motion blur. Our work falls in this category, and we are most related to image-space post-processing techniques: such approaches sample, manipulate (e.g., dilate), and blur frame-buffer colors according to screen-space color and velocity information (Sousa [94], Kasyan and Schulz [58]). Recent tile-based variants segment image-space to more accurately determine blurring directions and neighborhoods, approximating the motion blur scattering operation as a localized gather. Blurring along a single, "dominant" velocity direction (Lengyel [62], McGuire et al. [69], Zioma and Green [115], Sousa [95]) is most efficient; however, we base our comparisons on the most recent "multi-direction" tile-based post-process approach of Guertin et al. [37], which is capable of resolving many of the tile- and image-space artifacts of previous "single-direction" techniques, but still maintains a very high-performance profile.

We will show that even the most robust high-performance post-process motion blur technique can fail in common scenarios, specifically when non-linear motion exists and/or large motion magnitudes (and/or large exposure times) are used. We are able to generate more accurate and more spatially/temporally coherent motion blur in these scenarios, with only a modest performance overhead: instead of requiring on the order of 1 to 3ms (as in Guertin et al. [37]), our (unoptimized) approach requires 3.5 to 6.5ms.



Fig. 28. Grid scatter pass: we fit Bézier curves to a grid of pixel-aligned vertices with a vertex shader, querying an object's previous and next positions. We discretize the curves into line segments in a geometry shader and then rasterize the segments. We compute the spatially-varying line color based on the originating pixel's color and distance-based weight in a pixel shader.

Alternative Rendering Architectures. Recent work on GPU micropolygon rendering (Akenine-Möller et al. [2]) and stochastic rasterization techniques (Akenine-Möller et al. [2], McGuire et al. [68]) provides a middle-ground between accuracy and performance: object visibility and shading are decoupled, which allows a more accurate motion blur effect compared to interactive post-processes, if at an increased cost.

3. Method

Modern interactive motion blur approaches rely heavily on the principle of *scatter-as-gather*, since algorithms designed in this manner can readily benefit from accelerated processing on

massively parallel modern GPU architectures. Specifically, a scatter operation (such as motion blur), where each pixel $p_{x,y}$ influences the value at one or more pixels $p_{x',y'}$, is implemented as a gather operation, where each pixel $p_{x,y}$ queries pixels in its neighborhood $p_{x',y'}$ to determine their potential contributions (see Figure 27). In the general case, the gather solution would require a neighborhood size equal to the image resolution in order to perfectly simulate the scatter, but a common acceleration strategy reduces this neighborhood size heuristically in exchange for introducing some approximation error. In the motion blur setting, this restriction constrains both the form (i.e., linear vs. non-linear) and the length of motion blur features.

Our method instead directly implements the scatter solution to motion blur, but in a manner that completely avoids its principal disadvantage: scattering on a GPU is inherently inefficient since it reduces thread coherence by performing unordered buffer writes. Moreover, unordered buffer writing operations are typically unoptimized at the driver- and hardware-levels since they break the SIMD processing model of GPUs, further increasing their cost in practice. In contrast, *primitive rasterization* is perhaps the most optimized set of routines on a GPU; we exploit the fact that primitive rasterization reduces to a series of unordered buffer writes, and build our optimized GPU solution atop it.

We discuss our rendering approach below. It is divided into three stages: a motion pre-pass, a *grid scatter* pass, and a normalization pass.

3.1. Motion Pre-pass

Similarly to most post-processed interactive motion blur algorithms, we first use a set of prepasses to output the necessary motion data. Specifically, we store two buffers with the location of each pixel's geometry in screen-space at two different time steps. Each pixel has an associated 3D position and 1D time coordinate, $p(t) = [x_t, y_t, z_t, t]$, with the current frame's pixel p(0) at t = 0, and the "previous" and "next" buffers storing p(-1) at t = -1 and p(1) at t = 1. At render-time, given the current pixel's screen coordinates (x,y), we can retrieve its full screen-space position at these three different points in time.

3.2. Grid Scatter

Given the motion data, the first pass of our algorithm requires a set of pixel-aligned vertices, at the same $M \times N$ resolution of the final image, with positions $p_{x,y}$ where $1 \le x \le M, 1 \le y \le N$. We generate (and render) this data as a pre-generated point list on the GPU.

Bézier Curve Computation. We first fit a Bézier curve B(s) at each pixel from the three positions we have at times $t = \{-1,0,1\}$. To do so, we perform vertex position texture fetches in the vertex shader and set the necessary fitting constraints for a Bézier curve *B* as

$$B(0) = p(-1)$$
 $B(1/2) = p(0)$ $B(1) = p(1)$
to solve for the curve control points

$$B_0 = p(-1)$$
 $B_2 = p(1)$ $B_1 = p(0) - \frac{p(-1) - p(1)}{2}$.

We then output these three points, as well as the (u,v) coordinates of the vertex, which will be used later on. We chose $s = \{0, 1/2, 1\}$ since we uniformly sampled the time steps, but different sample location points could also be used.

Discretization and Generation. We leverage a geometry shader during the second step of our algorithm. Using the three points outputted from the vertex shader above, we generate geometry to represent the Bézier curve as faithfully as possible. Given our performance targets, we have found that discretizing the curves into line lists balances accuracy and performance, especially since lines are efficient to compute and rasterize across a limited number of pixels (scaling with O(N), versus $O(N^2)$ for a polygonal approximation, for *N* curves). The geometry shader generates a list of fixed length Δ lines as follows:

$$\left\{ (1-s^2)B_0 + 2(1-s)sB_1 + s^2B_2 \middle| s \in S \right\}$$
(37)

with S defined as

$$S = \frac{1}{\Delta - 1} \{0, 1, 2, \dots, \Delta - 2, \Delta - 1\}.$$
(38)

We discard motionless pixels prior to segment generation.

Rasterization. The final step in our first pass rasterizes the line segments into an accumulation buffer. Given many line segments, the potential for significant overdraw is high and so our shading routine must remain as simple as possible. To do so, we can simply use the (u,v) coordinates stored in the first step to query the color of the original pixel and output this constant color for the line, effectively implementing the scattering operation. While this works, we can improve the visual quality of the blur by additionally weighting the sampled color according the pixel's interpolated *s* coordinate value. We use a simple 1D Gaussian blur kernel with $(\mu = 1/2, \sigma)$, and we also output this weight to the alpha channel.

We render every pixel with fully additive alpha blending so that the final buffer stores the sum of all scatter operations. We additionally enable depth tests but disable depth writes: this means that each line also is rendered using proper z-order tests with the other objects in the scene, correctly accounting for objects moving behind other objects, even through heavily non-linear motion. This strategy also has the benefit of reducing the number of processed pixels processed with early depth testing. Due to the high variability of the values written in the accumulation buffer, we recommend using a 32-bit floating-point buffer.

3.3. Normalization

The second pass of our algorithm is a "traditional" fullscreen post-processing pass, with the rendered scene and accumulation buffer as input. For each pixel, we wish to compute the weighted average of every line rasterized onto the pixel. Concretely, we wish to compute the color $c'_{x,y}$ of the pixel at (x,y) according to the contribution of all of the other pixels on the screen:

$$c'_{x,y} = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} w_{i,j,x,y} c_{i,j} + w_b c_{x,y}}{\sum_{i=1}^{M} \sum_{j=1}^{N} w_{i,j,x,y} + w_b}$$
(39)

where $c_{x,y}$ is the original color of the pixel at (x,y) before any blurring, $w_{i,j,x,y}$ is the weight of the contribution of pixel (i,j) to pixel (x,y), and w_b is the (constant) background weight.

The accumulated values in buffer A, generated during the previous pass, effectively stores the first term of the numerator in Equation (39) and its alpha channel α stores the first term of the denominator and so we can trivial compute Equation (39) in a pixel shader as

$$c'_{x,y} = (A_{x,y} + w_b c_{x,y}) / (\alpha_{x,y} + w_b) .$$
(40)

We additionally output $\alpha_{x,y}$ in the alpha channel to support transparency. Note that the explicit background contribution is required, since we discard pixels without motion: without it, all motionless pixels would render as black.

4. Results

All results were computed on a Core i7-3770K with 16GB of RAM and a GTX780. Unless noted otherwise, we render at 1920×1080 and with $\{w_b, \Delta, \sigma\} = \{5, 11, 2\}$. We compare against an optimized implementation of Guertin et al.'s efficient tile-based motion blur post-process (Guertin et al. [37]) using the parameters listed in the paper, as well as comparing against ground-truth computed using brute-force accumulation (with temporal samples distributed according to a Halton sequence to minimize banding). We adjusted motion magnitudes in each scene in order to produce similar blurring effects for each of the three algorithms, and we chose a linear blur sample count *N* for each scene that reduces noise or outright eliminates it wherever possible. We do not apply any antialiasing.

Helicopter Scene. The first scene is the simplest, but highlights an important feature of non-linear blur: the ability to correctly motion blur spinning objects. While linear algorithms can approximate very low velocity rotations, they quickly fall apart as soon as faster motions (and/or longer exposures) are used. The helicopter's spinning blades are a simple representative example. As illustrated in Figure 29, linear algorithms are unable to represent the arcing motion of the blades and tail rotor, and instead approximate it as patches of discrete linear velocity blurs (in wildly different directions). For thin objects such as the blades, the effect is incorrect but relatively acceptable. For



Fig. 29. The helicopter scene has significant rotational motion, a common failure case for linear motion blur: linear algorithms fail to properly convey the scale of motion, they cause over- and under-blurring, and they either give the impression of pure linear motion (as seen on the blades) or cause a pinwheel artifact (as seen on the tail rotor). This is due to clamping to dominant linear velocity directions at different angles, across tiles.

round objects such as rotor, a distracting "pinwheel artifact" is glaringly obvious. Compared to the reference image, we note that apart from the additional presence of the unblurred image, the shape and appearance of the blur surrounding the blade is very accurate (see Section 5), closely matching ground-truth.

Teapot and Cubes Scene. The teapot scene represents a more chaotic animation, with many objects moving along different (often curved) trajectories. This scene highlights the stability of our approach, which more accurately follows all motion vectors for every object; approximating blurs per-tile, on the other hand, can lead to directionless blurs and blur effects that are difficult to visually parse. Specifically, previous approaches have a tendency to over-blur the top of the teapot, where chaotic motion is highest and thus where it is extremely likely for a few highly mobile pixels to cause an entire tile neighborhood's blur estimate to deviate. These approaches also have difficulty rendering the motion blur of the movement at the bottom, where motion is largely linear and parallel, but of a higher magnitude due to gravity. Our non-linear algorithm manages to accurately represent both scenarios, once again achieving a result that is very close to the reference, aside from the overlaid presence of the unblurred objects (see once again Section 5).

Jumping Jack Scene. The last scene illustrates the algorithm's behavior with rigged characters. Character animations are an excellent example of non-linear motion, since limbs generally perform rotational movements rather than purely rectilinear ones. As with previous scenes, the blur's magnitude is more accurate with our approach, and it varies smoothly depending on the actual

velocity of the limb at any given point. Details are better preserved and shading is closer to the reference, ground-truth accumulated image.

4.1. Performance

Due to our algorithm's design, computation time tends to be higher on average versus linear techniques, but not significantly so (see Table 1). The pre-pass cost is easy to quantify: it is roughly double the cost of the linear algorithm's pre-pass, as it requires two buffers instead of one. The post-process cost is more complex, since it depends on the scene, including the area of the screen which is blurred as well as the magnitude of the blur. Even so, it appears that a good experimental estimate is roughly double the cost of the linear algorithm's post-process. As such, it is fair to say that our non-linear approach is roughly twice as expensive as the state of the art linear motion blur post-process. This may seem significant, but in practice many modifications could be applied to limit the impact of our post-process (see Section 5.1); even then, it is important to note that we generate higher quality results in only **4 to 6.5ms** of compute time.

5. Discussion and Limitations

While our approach generates good results in only a handful of milliseconds, especially in scenes where the state of the art fails, it still has some drawbacks which we discuss below and will address in future work (see Section 5.1).

Performance. Due to our very straightforward blurring approach treatment, we also impose some additional constraints on our input. These two properties lead to variance in the rendering cost: a scene with little to no blur can easily be cheaper than existing, linear algorithms, since our shader code is comparatively simpler; however, a scene with large amounts of blur can cause significant overdraw and reduce performance. On average our approach has modest performance characteristics requiring between 4 and 6.5ms of compute time, but this is still 50 to 100% slower than the state of the art. More complex scenes are penalized more by the requirement of our second motion pre-pass.

Magnitude Constraints. Unlike existing techniques, our algorithm does not impose any constraint on the magnitude of the motion (or the exposure time), but extremely large and high-frequency motions are unlikely to be captured accurately: any lack of motion information between our sampled time steps, as well as the inherent limitations of simple quadratic Bézier curve fits, allow our approach to handle motions roughly a few times the magnitude of current algorithms, which is still a significant improvement but is not completely general.



Fig. 30. This teapot scene showcases chaotic movement with many superimposed velocities on each pixel of the screen. We note that most pixel motion is slightly curved, which is more faithfully represented with our non-linear approach. Moreover, there is a high variability in the motion vectors, which is only partially captured by linear tile-based algorithms, while our algorithm properly accounts for all directions and all magnitudes of motion.

Masked Information. The post-processing nature of our approach adopts the same limitations as existing techniques: due to the limited scene information available per-pixel, moving objects occluded by other objects (whether stationary or not) will not produce any motion blur, which can cause vanishing motion trails when objects move in or out of view.

Depth Ordering. Our algorithm processes all pixels simultaneously and without any constraint as to depth ordering. Since we use purely additive blending, this does not affect the final result, but a more accurate blending would require proper depth ordering. It is possible that improvements such as rasterizer ordered views [73] may allow fast and accurate sorting, therefore presenting more opportunities for accurate blending.

	Linear			Non-linear	
	Pre-pass	Post-process	Samples	Pre-pass	Post-process
Helicopter	0.45	1.4	35	0.88	3.1
Teapot	0.25	5.2	61	0.51	5.8
Jumping Jack	0.076	2.0	35	0.15	3.8

Table 1. Performance comparisons (in milliseconds) at 1920×1080 for the scenes in Figures 29, 30, and 32. Sample counts are provided for the linear algorithm. As expected, our pre-pass is almost exactly twice as costly since it requires two motion vector passes. Our approach is clearly slower, however the difference is not overwhelming and we have not yet made any optimization attempts.



a) Single-pass rendering

b) Two-pass rendering

Fig. 31. Comparison between non-linear blur using the basic, single-pass algorithm (**a**) and using our modified two-pass rendering (**b**). Segmenting static objects allows us access the background color behind the blurred objects, resulting in a more accurate relative weighting between the foreground and background, **completely eliminating** the most distracting visual artifact of our current approach.

5.1. Extensions

We will outline the avenues of future work we will pursue in order to address some of the limitations listed above.

Cubic Bézier Curves and Circular Arcs. We can alleviate some of the limitations caused by the use of quadratic Bézier curves by moving to higher-order Bézier curves: a cubic curve would require a third motion pre-pass (i.e., at either t = -2 or t = -1/2; see Section 3), but would produce a much more accurate blur, especially for larger movements. Neither quadratic nor cubic curves would support conic motions however, and so a third option would be to instead fit circular arcs to the motions. Transitioning between these different representations is an interesting idea we are exploring as well. This is of particular interest in scenes with fast spinning objects.

Multi-pass Rendering. In order to address artifacts that result from a lack of background information (which is a limitation inherent to post-process approaches), we require some important changes to the rendering pipeline. One possible trade-off would be to segment and render the scene twice, storing two color and depth buffers: one for objects in motion, and one for static objects. In this case, camera motion would have to be handled separately. Now, we can apply our approach to the moving objects, whereby they exclusively access information from "moving" color/depth buffers *except* when computing the background color contribution, where they use the "static" buffer's colors. This significantly improves the appearance of the blur with only moderate additional overhead, effectively removing the most prevalent visual artifacts of our approach (see Figure 31). An open problem in this extension is how to correctly handle very low velocities, where the background weight must tend toward zero.



Fig. 32. The jumping jack scene illustrates the algorithms' behavior with animated characters. Our approach better conserves the blur on the character's knee, whereas linear algorithms can almost completely miss this effect.

Optimizations. Our current algorithm is designed with clarity in mind, however several avenues for optimization are available: firstly, it should be possible to determine the magnitude of the movement and clamp generated curves' lengths according to the maximum; secondly, small motions can be handled using a cheaper approximation, with our approach toggled on a *per-pixel* basis to handle complex motion; lastly, rendering our blur at a reduced resolution and using, e.g., a bilateral upsampling technique (Sloan et al. [92]) is an obvious direction to explore.

6. Conclusion

We propose a new approximate motion blur post-processing approach capable of more accurately capturing blur effects caused by non-linear motion. Our approach reduces spatial and temporal noise (see the supplemental videos) and is designed to leverage the shader pipeline in order to implement a true scatter operation. As such, we are able to handle longer motion trails, our algorithm is simple to implement atop existing rendering engines, it scales independently of the underlying geometric complexity, and it generates spatially- and temporally-smooth results in scenes where the state of the art fails. We discuss the limitations imposed by our approach, and propose several avenues for future work. We have begun exploring these directions, and a preliminary result (Figure 31) shows significant promise, eliminating the most significant visual artifact of our approach.

Second Article.

Scalable Appearance Filtering for Complex Lighting Effects

by

Luis E. Gamboa¹, Jean-Philippe Guertin¹, and Derek Nowrouzezahrai²

- (1) Université de Montréal
- (2) McGill University

This article has been published in volume 37, issue 6 of the *ACM Transactions on Graphics*, in November 2018. It has then been presented at *ACM SIGGRAPH Asia* 2018 in Tokyo. Formatting has been adapted for this thesis and minor typographical issues were corrected.

I am the second author of this article. My main contributions are:

- Creation of multiple test scenes and demonstrations to best distinguish the algorithm with prior work, e.g., the cutlery scene used throughout, the donut scene of Figure 37 and the kettle scene of Figure 46.
- Implementation of parts of the algorithm in Mitsuba [49].
- Real-time implementation of the algorithm which unfortunately did not make the cut in the final publication.

Luis E. Gamboa formulated the theoretical foundation of the algorithm, devised many tests and implemented the algorithm. Derek Nowrouzezahrai contributed to the algorithm's design and supervised the work.

The real-time implementation was to be a significant element of our results, showing that our algorithm was highly scalable to workload and, given some smart tradeoffs, could perform interactively. After facing several technical issues with the implementation of the algorithm on the GPU, we had to publish without this contribution. However, these challenges indicated that the real-time implementation has merit as a standalone contribution, as we detail in Section 2.1.

RÉSUMÉ. Le rendu réaliste avec des matériaux qui présentent des variations spatiales à haute fréquence demeure un défi, puisqu'éliminer le crénelage spatial et temporel requiert un nombre prohibitif d'échantillons. Des travaux récents ont rendu le problème plus traitable, mais les méthodes présentées restent extrêmement coûteuses lorsque des lumières environnementales et/ou de l'illumination globale (correctement filtrée) sont utilisées. Nous présentons un modèle d'apparence avec des variations explicites à haute fréquence des micronormales, et une approche de filtrage d'intégrales d'ombrage multidimensionnelles qui reste efficace à plusieurs échelles. En combinant un système novateur et compact d'histogrammes de mi-vecteurs avec une expansion de bases directionnelle, nous calculons précisément l'intégrale de la réflectance haute fréquence filtrée sur des lumières étendues avec des émissions variables angulairement. Notre approche fonctionne à plusieurs échelles, rendant des images indistinguables de la référence à plus de $10 \times$ la vitesse de l'état de l'art et avec seulement 15% d'usage de mémoire. Lorsque nous filtrons l'apparence en considérant l'illumination globale, nous sommes $\sim 30 \times$ plus rapides que l'état de l'art.

Mots clés : cartes de normales, lueurs, harmoniques sphériques, images intégrales

ABSTRACT. Realistic rendering with materials that exhibit high-frequency spatial variation remains a challenge, as eliminating spatial and temporal aliasing requires prohibitively high sampling rates. Recent work has made the problem more tractable, but existing methods remain prohibitively expensive when using large environmental lights and/or (correctly filtered) global illumination. We present an appearance model with explicit high-frequency micronormal variation, and a filtering approach that scales to multi-dimensional shading integrals. By combining a novel and compact half-vector histogram scheme with a directional basis expansion, we accurately compute the integral of filtered high-frequency reflectance over large lights with angularly varying emission. Our approach is scalable, rendering images indistinguishable from ground-truth at over $10 \times$ the speed of the state-of-the-art and with only 15% the memory footprint. When filtering appearance with global illumination, we outperform the state of the art by $\sim 30 \times$.

Keywords: normal maps, glints, spherical harmonics, summed area tables



(a) Direct illumination

(b) Global illumination

Fig. 33. We filter direct (a) and global illumination (b) with high-frequency appearance and complex emitters $\sim 30 \times$ faster than the state-of-the-art and $\sim 10\%$ the memory footprint. Unless otherwise stated, our results have converged and residual noise is due to high-frequency appearance.

1. Introduction

Microfacet reflectance models are a powerful tool for expressing the behavior of real-world appearance. Traditionally, these models relied on aggregate statistical formulations of the normal distribution function (NDF); however, many real-world objects exhibit features at scales visible to a viewer. Meso- and macro-scale scratches, flakes and bumps all produce visually rich "*sparkle*" effects.

One challenge in simulating these effects lies in resolving aliasing in the *effective* reflectance inside a pixel's projected footprint (or the footprint of a path vertex, for indirect bounces), requiring prohibitively large sampling rates. Previous approaches (e.g., Jakob et al. [50], Yan et al. [113], Belcour et al. [15]) resolve this issue by hierarchically culling normals and positions when *evaluating* an appropriately filtered effective reflectance towards a *fixed lighting direction*.

These methods are tailored to sharp directional or point lighting, where sparkle effects can be quite pronounced; however, scenes with larger area lights or environmental sources can also exhibit sparkly behavior; here, resolving final antialiased images additionally requires *integrating* the *evaluation* of these previous models over the domain of the extended light source (Figure 34). Even with their efficient importance sampling schemes this integration over extended lights becomes prohibitively expensive, and the problem is compounded if we seek to simulate additional bounces of global illumination.

We present a filtered appearance model that admits an efficient numerical integration of incident radiance over a shading footprint from, e.g., all-frequency environmental light sources. We allow an explicit specification of the underlying normal variation and present a simple, efficient *double filtering* algorithm that adapts to *both* the frequency-content of the underlying lighting *and* the effective NDF within an arbitrary filter footprint. Our representation has modest memory needs, we



Smooth MF: 1m:23s

Our method: 2m:36s

Fig. 34. While high-frequency sparkle-like appearance is highlighted by strong directional or point lighting (middle row), environmental lights (top row) still contribute significantly to sparkly appearance. Our method (right column) generates converged images with these effects in less than twice the time needed to generate converged results with a smooth microfacet model (left column), and $\sim 10 \times$ faster than Yan et al. [113] (not shown).

easily incorporate it into standard offline and real-time rendering engines, and we demonstrate its ability to scale to scenes with complex lighting and global illumination.

Specifically, we present the following technical contributions:

- a novel spherical histogram to query scale-dependent NDFs in time independent of the normal map or footprint size,
- an efficient basis-space half-vector integrator that adapts to the frequency of *both* the incident radiance *and* multi-scale NDF, and
- applications to direct lighting and correctly filtered secondary bounces in global illumination, both with complex lighting.

We generate alias-free animations in a fraction of the time ($\sim 2-10\%$) of the state-of-the-art (Yan et al. [113], Belcour et al. [15]).

2. Previous Work

We aim to efficiently render alias-free ground-truth-quality images of scenes with microfacet BRDFs, high-frequency normal variation, all-frequency lighting, both with and without global illumination. While no current method can efficiently handle these scenarios, we discuss the most relevant prior work: specifically, we draw upon work in both the interactive and offline rendering communities.

2.1. Microfacet Models

A long history of work on appearance models for reflection from rough surfaces, using microfacet theory (Beckmann and Spizzichino [10], Cook and Torrance [19], Walter et al. [108]), lies at our foundation. As with most appearance filtering techniques, our model focusses on the representation and treatment of *continuous multi-scale* NDFs in the microfacet model, remaining agnostic to the choice for the remaining Fresnel and shadowing-masking terms in the model . We will outline the specific instance of these terms we employ in our model (Section 3), acknowledging the large body of work on both parametric (e.g., Ngan et al. [75], Bagher et al. [7], Dupuy et al. [31]) and non-parametric microfacet models (e.g., Bagher et al. [8]). We use the same framework but with a discrete set of normals rather than a continuous distribution.

2.2. Procedural Texture Antialiasing

Prior to most work on NDF-based appearance filtering, high-frequency and/or procedural textures were employed extensively as a means of introducing high-fidelity spatial variation in simple shading models (see Perlin [82]). Approaches for antialiasing these procedural textures include directly filtering out high spatial frequency content, or imposing simple geometric models atop the textures in order to facilitate more flexible filtering schemes (Cook and DeRose [21], Lagae et al. [59], Heitz et al. [45]). While suitable for albedo filtering, these approaches cannot be applied to filtering multi-scale NDFs.

2.3. Accurate Appearance Filtering

Brute force numerical integration is a simple and prohibitively costly method to accurately render sparkle effects that arise from high-frequency normal map variation (Yan et al. [112], Jakob et al. [50]). This problem is compounded if, in addition to having to resolve spatial aliasing within a pixel footprint, variation from incident radiance need also be integrated numerically. Neither efficient (multiple) importance sampling schemes (Yan et al. [112], Jakob et al. [50], Veach and Guibas [103]) nor more efficient pruning strategies (Yan et al. [113], Atanasov and Koylazov [6]) help, due to the nature of the full integrand, which includes a *product* of the filtered NDF with the incident lighting. Our filtered appearance model explicitly treats the fact that both a spatial *and* angular *integral* must be computed (Figure 36), as opposed to an *evaluation* of a spatial integral (Figure 35).

Most recently, several methods approach the appearance filtering problem with solutions to efficiently prune only the normals in an NDF that will contribute non-negligibly to the final shading, for a given view and lighting direction pair (Yan et al. [112], Jakob et al. [50], Yan et al. [113], Atanasov and Koylazov [6]). These work provide significant improvements over brute force

integration in the presence of strong point or directional lights, but they become prohibitively costly when large lights or global illumination are considered.

Two notable exceptions are the work of Raymond et al. [84] and Belcour et al. [15]. Raymond and colleagues present a multi-scale appearance model tailored to scratch microstructures, and they demonstrate its flexibility in the context of complex lighting. We instead treat arbitrary user- and procedurally-generated high-resolution normal variations, and explicitly compute both the spatial and angular integrations with a single method. Belcour et al. [15] integrate an existing appearance filtering model (Yan et al. [112]) atop a covariance tracing-based global illumination framework in order to correctly filter indirect bounces off of, and on to, sparkly materials. This approach demonstrated orders of magnitude performance improvements over brute force Monte Carlo (at the time, this was the state-of-the-art for accurately simulating *indirect bounces* of sparkly materials). We similarly integrate our appearance model into a filter-aware global illumination algorithm, and demonstrate a $30 \times$ performance improvement over the approach of Belcour et al.

Recently, appearance models that treat wave optics for fine-scale microstructures have demonstrated the ability of simulating subtle iridescence effects (Werner et al. [110], Yan et al. [114]). We rely on geometric optics and instead focus on accelerating lighting integration for multi-scale appearance in direct and indirect illumination. Incorporating wave optics in our model is an interesting, complementary direction of future work.

2.4. Fast Appearance Filtering

Initial work on normal antialiasing was led by the interactive graphics community. Here, smooth and compact normal distributions were used to leverage graphics MIP hardware to perform fast multi-scale filtering (Toksvig [99], Olano and Baker [79], Dupuy et al. [30]). These single-lobe approximations of the NDF are suitable for interactive applications, but they cannot capture the details and anisotropies of NDFs (across scales) with high spatial resolution (Yan et al. [112]).

Han et al. [**41**] notably use spherical harmonics (SH) and spherical Gaussian mixture models of the multi-scale NDF, allowing for multimodal NDFs. We too rely on SH, but only when performing our final shading (i.e., directional integration) and *not* when computing the form of the multi-scale NDF (i.e., spatial normal map filtering). Combined with an efficient half vector-space shading formulation (Section 4.2), this allows us to correctly integrate environmental lighting with both all-frequency underlying BRDFs *and* all-frequency environmental lighting, generating spatially-and temporally-antialiased image sequences.

Finally, real-time approximations of existing accurate appearance filtering approaches for glints (Zirr and Kaplanyan [116]) and iridescent scratches (Velinov et al. [104]) have shown promise towards pushing these important visual effects into an interactive context. The approach of Zirr and Kaplanyan [116] is effective but limited to simple lighting, whereas Velinov et al. [104] derive analytic approximations for spherical and polygonal area lights. Their model, however, only treats

scratch-like microstructures, whereas we consider both arbitrary normal maps *and* arbitrary, i.e., environmental incident illumination. While we primarily target offline, fully-accurate simulations, we also discuss a proof-of-concept interactive renderer implementation in Section 6.

3. Preliminaries & Baseline Appearance Model

As with recent work on filtered appearance we build atop a standard microfacet BRDF model of reflection from smooth surfaces

$$f_r(x, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \boldsymbol{\rho}_d + \boldsymbol{\rho}_s \left(\frac{D(x, \boldsymbol{\omega}_h) F(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_h) G(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)}{4(n \cdot \boldsymbol{\omega}_i)(n \cdot \boldsymbol{\omega}_o)} \right) , \tag{41}$$

but with a critical deviation that the *normal distribution function* (NDF) $D(x, \omega_h)$ is a scaledependent, potentially high-frequency and anisotropic spherical distribution in the space of halfvectors $\omega_h = (\omega_i + \omega_o)/|\omega_i + \omega_o|$. Here, ω_i and ω_o are the incident and outgoing lighting direction at the shading point *x* (with geometric normal *n*), ρ_d and ρ_s are diffuse and specular reflection coefficients, $F(\omega_i \cdot \omega_h)$ is the Fresnel term and a geometry term $G(\omega_i, \omega_o)$ captures micro-scale shadowing, masking and inter-reflection.

Techniques that target ground-truth-quality renderings of microfacet models with high-frequency normal variation can be categorized by whether they allow **implicit** or **explicit** specification of the underlying normal variation. Explicit approaches (e.g., the "glints" methods of Yan et al. [**112**, **113**]) are flexible in that they allow an arbitrary high-resolution normal map texture as input, but generally have large memory requirements (see Section 7). Implicit approaches (e.g., the "discrete microfacet" method of Jakob et al. [**50**]) instead rely on statistical processes to describe the underlying high-frequency normal variation (typically using lazy, on-the-fly evaluation), leading to more compact run-time algorithms but at the cost of reduced control over appearance variations.

We will efficiently compute filtered appearance integrals *in the presence of* complex incident lighting, and we propose our solution in Section 4. First, however, we establish a *baseline* filtered appearance model with which we can generate ground-truth results and compare to in the context of prior work. As with our final model (Section 4), our baseline model is an explicit model and, so, allows for user-controllability through arbitrary normal map inputs. Our baseline (and final) models will require low memory footprints, in order to facilitate comparisons on complex scenes. In the context of prior work, our baseline model combines the *glint* (as in Yan et al. [112, 113]) and *discrete microfacet* models (as in Jakob et al. [50]), and so we refer to it as the G×D model.

3.1. Baseline Filtered Microfacet Model (G×D)

The $D(x, \omega_h)$ term in Equation (41) can be expressed as a hemispherical probability distribution (over half-vectors) of the angle θ_h formed by every microfacet normal with the underlying macro-scale/geometric normal *n* at a shade point *x*. Unlike smooth microfacet models, the NDF within



Fig. 35. Previous methods only integrate over a *spatial* footprint, before evaluating filtered appearance in the direction of a point/directional source.

a filtering footprint \mathcal{P} (e.g., subtended by the projection of a pixel onto a shading surface; see Figure 35) can be modelled according to the individual normals that lie inside it.

We borrow and combine previous notation from Jakob et al. 2014 and Yan et al. [50, 112] and define the normalized NDF as the set of normals ω_x at shading locations x inside footprint \mathcal{P} , each of which represents the mean direction of a spherical Gaussian G_s that models the roughness of an equivalently-smooth underlying microsurface (Figure 35),

$$\widehat{D}(\mathcal{P}, \boldsymbol{\omega}_h) = \int_{\mathcal{P}} G_s(\boldsymbol{\omega}_h; \boldsymbol{\omega}_x, \boldsymbol{\Sigma}_s) dx, \qquad (42)$$

where the Gaussian roughness centered about each micronormal is parameterized by its covariance matrix Σ_s . In practice, we apply isotropic roughness, and so a diagonal Σ_s with elements σ_s .

While Equation (42) assumes an arbitrary underlying spatial distribution of micronormals, in practice we use *discrete* micronormal distributions defined in normal map textures. That is, our normalized NDF is the finite set of *explicitly specified* normals ω_x located at shading locations x inside footprint \mathcal{P} . For this reason, it can be convenient to rewrite Equation (42) in terms of normal map texels (dropping the dependence on σ_s , for brevity), as

$$\widehat{D}(\mathcal{P},\boldsymbol{\omega}_h) = \frac{1}{N_{\mathcal{P}}} \sum_{\boldsymbol{\omega}_x \in \mathcal{P}} G_s\left(\boldsymbol{\omega}_h; \boldsymbol{\omega}_x\right), \qquad (43)$$

where $N_{\mathcal{P}}$ is the number of texels that the footprint \mathcal{P} projects onto in texture-space (Figure 35, right), and we assume equally weighted (i.e., box-filtered) averaging of these micronormals.

Note that as $\sigma_s \to 0$ the roughness approaches mirror-like (delta) reflection and $G_s \to \delta_{\Omega}$, our baseline approaches the model of Jakob et al. [50], but with the important difference that micronormals are defined *explicitly*.



Fig. 36. Our method integrates high-frequency appearance variation over both spatial footprint and incident lighting directions. To integrate over all light directions ω_i , as is needed for environment lighting, previous work requires many Monte Carlo samples, each of which involves an expensive integral over \mathcal{P} . Our method solves both integrals with a single evaluation.

Incorporating the multi-scale NDF (i.e., for arbitrary pixel footprint size) described in Equation (42) into Equation (41), involves changing the notion of a point-wise BRDF f_r (dependent on intersection point *x*) into a "multi-scale" BRDF \hat{f}_r that depends on pixel footprint \mathcal{P} . Having done this, we can evaluate for a *single* view and light direction, arriving at the formal definition of our baseline G×D filtered appearance model:

$$\widehat{f}_{r}(\mathcal{P},\boldsymbol{\omega}_{i},\boldsymbol{\omega}_{o}) = \rho_{s} \int_{\mathcal{P}} \frac{D(x,\boldsymbol{\omega}_{h})F(\boldsymbol{\omega}_{i}\cdot\boldsymbol{\omega}_{h})G(\boldsymbol{\omega}_{i},\boldsymbol{\omega}_{o})}{4(n\cdot\boldsymbol{\omega}_{i})(n\cdot\boldsymbol{\omega}_{o})} dx$$

$$= \rho_{s} \frac{\widehat{D}(\mathcal{P},\boldsymbol{\omega}_{h})}{\int G_{s}(\boldsymbol{\omega}_{h};\boldsymbol{\omega}_{x})dx}F(\boldsymbol{\omega}_{i}\cdot\boldsymbol{\omega}_{h})G(\boldsymbol{\omega}_{i},\boldsymbol{\omega}_{o})$$

$$= \rho_{s} \frac{\mathcal{P}}{4(n\cdot\boldsymbol{\omega}_{i})(n\cdot\boldsymbol{\omega}_{o})}, \qquad (44)$$

where we solve the spatial integral by means of Equation (43) and we employ a factorization of the *G* term as a product of rational approximations of the Smith shadowing term, $G(\omega_i, \omega_o) \approx$ $G_1(\theta_i) G_1(\theta_o)$ with $G(\theta)$ (see Walter et al. [108]), and we use the exact Fresnel term for conductors.

Note that shading with Equation (44) for lighting configurations that require anything other than a **simple evaluation** of (single) ω_i , and hence ω_h , directions would have to rely on brute force numerical integration. This quickly becomes prohibitively expensive (Figures 36 and 37). In Section 4 we present a mathematical representation, and detail an efficient and scalable algorithm, for solving integrals of Equation (44) against (non-delta) lighting distributions.

Our method inherits the following assumptions from Yan et al. [112, 113] and Jakob et al. [50]: locally flat macro-geometry and no view/light variation within the footprint. Our results are consistent with prior work, as can be observed even under highly curved surfaces, e.g., the snail (Figure 33).

3.2. Evaluating G×D

We adapt the pruning algorithm of Jakob et al. [50] to treat explicit normal maps and Gaussian roughness, when evaluating Equation (44) for ground-truth baseline renderings. Specifically, we build a min-max MIP-hierarchy in texture space. Spatial footprint pruning is trivial, as we project a conservative bounding box for pixel (or indirect path vertex) footprints onto the UV space. Normals are encoded as 2D (*s*,*t*)-coordinates on a projected disc parameterization (see Yan et al. [112]) and, at each MIP-level, a min and max (*s*,*t*) are computed and used to cull ω_x directions that fall outside of \mathcal{P} and Ω_o . The hierarchy is constructed once at start-up at a cost negligible to the total render time.

Of note, this $G \times D$ approach requires less memory and is consistently faster than the earliest approach of Yan et al. [112], however Yan et al.'s latter approach [113] converges between $5 - 25 \times$ faster (albeit requiring typically 100× more memory with higher per-evaluation cost) than $G \times D$. For this reason, we often treat $G \times D$ as our "brute force" solution for ground-truth image generation (even though it is roughly 500× faster than naïve importance sampled Monte Carlo), and we treat the latest Yan et al. technique [113] as our "high-performance" benchmark. All results generated with Yan et al.'s [113] techniques rely on implementations provided to us by the authors.

Figure 37 demonstrates the performance gap introduced when even the state-of-the-art (Yan et al. [113]) has to rely on numerical integration to resolve variation in incident radiance. This gap is compounded by global illumination (e.g., Figures 33 and 46).

In Section 4, we will incorporate incident lighting variation as a component of a new filtered appearance model and detail an approach for computing integrals that filter over the spatial, outgoing view *and* incident lighting dimensions. Our method relies on a simple multi-scale spherical histogram (Section 4.1) and basis-space integration scheme (Section 4.2), it is easy to implement in existing rendering engines (Section 6), and it scales favorably in performance and memory (outperforming the state-of-the-art by $10-30\times$).

In Section 5 we will apply our model to filtering direct illumination and multi-bounce global illumination transport, both in scenes with complex environmental (and point) lighting.

4. Filtering Appearance in Space and Direction

To synthesize spatially- and temporally-antialiased image sequences with sparkly materials, complex lighting and global illumination, we devise a compact and efficient approach to compute

integrals of Equation (44) over arbitrary illumination signals $L_i(\omega)$, namely

$$L_{o}(\mathcal{P}, \boldsymbol{\omega}_{o}) = \int_{\Omega_{i}} L_{i}(x, \boldsymbol{\omega}_{i}) \widehat{f}_{r}(\mathcal{P}, \boldsymbol{\omega}_{i}, \boldsymbol{\omega}_{o}) (n \cdot \boldsymbol{\omega}_{i}) d\boldsymbol{\omega}_{i}$$

$$= \int_{\Omega_{i}} \int_{\mathcal{P}} \frac{L_{i}(x, \boldsymbol{\omega}_{i}) D(x, \boldsymbol{\omega}_{h}) F(\boldsymbol{\omega}_{i} \cdot \boldsymbol{\omega}_{h}) G(\boldsymbol{\omega}_{i}, \boldsymbol{\omega}_{o})}{4(n \cdot \boldsymbol{\omega}_{o})} dx d\boldsymbol{\omega}_{i},$$
(45)

where Ω_i are hemispherical directions about the macro-scale normal *n* and we drop the specular coefficient ρ_s for brevity. In this context, $D(x,\omega_h)$ is the point-wise NDF $G_s(\omega_h;\omega_x)$. State-of-the-art techniques would have to rely on Monte Carlo integration to resolve Equation (45) and, even with efficient (spatially-filtered) BRDF importance sampling (Yan et al. [112, 113], Jakob et al. [50]), light importance sampling and MIS (Veach and Guibas [103]), this is prohibitively costly (Figure 37).

We simplify Equation (45) by factoring and approximating the Fresnel, geometric terms and microfacet normalization terms from the integral with a transfer function $\overline{FG}(\omega_o)$ (see Appendix A for details), leading to our *lighting-aware* filtered appearance model,

$$L_{0}(\mathcal{P},\boldsymbol{\omega}_{o}) \approx \overline{FG}(\boldsymbol{\omega}_{o}) \int_{\Omega_{i}} \int_{\mathcal{P}} L_{i}(x,\boldsymbol{\omega}_{i}) D(x,\boldsymbol{\omega}_{h}) dx d\boldsymbol{\omega}_{i}.$$

$$(46)$$



Fig. 37. Middle: Our method generates a converged filtered result in 4.89s (1spp) in a scene with a high-resolution normal map and environmental lighting. **Top halves:** equal time for our baseline (2spp, left) and Yan et al. [113] (1spp, right). Bottom halves: Equal quality rendering requires 16,384spp and 13m:8s with our baseline and 512spp and 2m:25s with Yan et al. [113].



Fig. 38. We use texture-space accumulated NDF histograms to efficiently query the multi-scale NDF histogram for an *arbitrary filtering footprint* \mathcal{P} using four lookups into this SAT-like data structure (histogram radial sizes not to scale, above). Figure 43 illustrates the visual impact of different histogram resolutions.

An immediate issue arises when addressing solutions to Equation (46): the discrepancy in the natural spherical parameterizations of L_i and D. We choose to perform integration in the space of half-vectors ω_h and reparameterize Equation (46) appropriately by introducing an incident illumination function \hat{L}_i that is dependent on both outgoing and half-vector directions, reducing the problem to solving equations of the form

$$\int_{\Omega_{i}} \int_{\mathcal{P}} L_{i}(x,\omega_{i}) D(x,\omega_{h}) dx d\omega_{i}$$

$$= \int_{\Omega_{h}} L_{i}(\bar{x},r(\omega_{o},\omega_{h})) (4(\omega_{h}\cdot\omega_{o})) \int_{\mathcal{P}} D(x,\omega_{h}) dx d\omega_{h}$$

$$\approx \int_{\Omega_{h}} \widehat{L_{i}}(\bar{x},\omega_{o},\omega_{h}) (4(\omega_{h}\cdot\omega_{o})) \int_{\mathcal{P}} G_{s}(\omega_{h};\omega_{x}) dx d\omega_{h},$$
(47)

where the $4(\omega_h \cdot \omega_o)$ factor accounts for the change of parameterization between differential incident and half-vectors (Torrance and Sparrow [100], Walter [105]), Ω_h are the hemispherical directions about ω_o , $r(\omega_o, \omega_h) = \omega_i$ reflects ω_o w.r.t. ω_h , and we define our *filtered multi-scale* NDF, $\hat{D}(\mathcal{P}, \omega_h)$, which depends on the filtering footprint \mathcal{P} . Note that above, as with previous work, we assume the incident illumination \hat{L}_i does not vary significantly within the spatial footprint and use a "central" shading point \bar{x} to distinguish between its spatial variation at scales larger than a single filtering footprint. Sections 5 and 6 will detail our representation of the incident illumination \hat{L}_i , which allows us to handle physically-based microfacet models in the presence of environmental lighting, addressing certain limitations that we discuss regarding previous work (e.g., Han et al. [41]; see Section 6.3). For the purposes of our exposition here, however, it is safe to assume that we have arrived at an expression \hat{L}_i that can be treated as a simple (hemi-)spherical function of ω_h .

Now, to efficiently solve integrals of the form in Equation (47) (and, so, Equation (46)), we combine two ideas that lead to orders of magnitude faster performance than the state-of-the-art with a modest memory requirement: first, a representation for computing multi-scale spherical NDFs over



Fig. 39. We adapt the shading to the minimum between the max frequency bandwidth of the multi-scale NDF in a footprint (top row) and the max bandwidth of the lighting environment. We visualize environment maps bandlimited to this *effective shading bandwidth* (bottom).

arbitrary filtering footprints in constant time, using *spherical histograms*; and, second, a basis-space representation for computing efficient spherical integrals that can *adapt* to the frequency-content of *both* the incident lighting \hat{L}_i and the multi-scale NDF *D* inside a footprint.

4.1. Multi-scale NDFs Using Histogram Accumulation

For explicit high-frequency normal variation represented in, i.e., a normal map, the multi-scale NDF $\widehat{D}(\mathcal{P}, \omega_h)$ is a normalized spherical distribution of all the normals that lie inside the projected filtering footprint \mathcal{P} , convolved with the isotropic roughness kernel G_s . One can naïvely arrive at a discretized (e.g., a spherical texture map) approximation of $\widehat{D}(\mathcal{P}, \omega_h)$ by convolving and summing each normal map texel that lies inside the projection of \mathcal{P} onto (u,v)-space. This naïve approach is not favorable for two key reasons:

- (1) the cost of computing \widehat{D} would scale linearly in the size of \mathcal{P} , and
- (2) the cost of spherical Gaussian convolution would scale with the underlying discretization resolution for \widehat{D} .

Motivated by these scalability issues, we propose a novel representation and an efficient datastructure, based on *spherical histogram accumulation*, to compute a discrete representation of \hat{D} in **constant time** for footprints \mathcal{P} of arbitrary size (item #1, above). We detail our histogram formulation below, before discussing how basis-space representation can help us avoid the scalability cost of spherical Gaussian convolution (item #2, above; see Section 4.2).

We observe that, instead of individually representing every possible discrete micronormal direction, we can collect histogram statistics of the normals that fall within a footprint. This has two benefits: first, we show below how histogram statistics for an *arbitrary footprint* can be computed using histogram statistics collected for each *individual* micronormal in the normal map, and this

all in constant time independent of the footprint size; second, we can control the NDF accuracy by adjusting the histogram bin resolution.

4.1.1. Histogram initialization. Specifically, we define a new 2D histogram texture with the same resolution as the normal map. Each texel stores an *accumulated spherical histogram of normals*, encoded as a 2D bin of (θ, φ) -space normal buckets. Bins have equal elevation and azimuthal angle extents. Given the SAT-like structure, a value at position (u',v') corresponds to the histogram of normals in the rectangular area (of the normal map) with corners (0,0), (0,v'), (u',0) and (u',v')

At initialization, beginning from the top-left histogram texel, we accumulate normals into bins from left-to-right and top-to-bottom (see Figure 38): each histogram texel accumulates additional normals from the normal map to cover the same area up to the current (u,v)-coordinate. Normals only get added (not removed) to histogram bins as we traverse across texels, and so a histogram at texel (u',v') is a strict superset of the histograms at texel, e.g., (u'-1,v'-1). In this case, the histogram at (u',v') is equal to the sum of histogram entries at (u',v'-1) and (u'-1,v') (correcting for overlaps) plus the corresponding frequency increase for the normal at (u',v') in the normal map. This structure allows for a fast sweeping accumulation, which adds a negligible overhead at startup (see Section 6.1).

4.1.2. Summed Area Table Histogram Queries. At run-time we compute the NDF histogram of an *arbitrary* axis-aligned (u,v) footprint by treating the histogram as a summed area table, using only four constant cost queries (Figure 38). As with Yan et al. [112, 113], we utilize axis-aligned (u,v) footprints and a Gaussian image reconstruction kernel to approximate Gaussian footprint weighting.

Given the ability to compute multi-scale NDF histograms (efficiently, in constant time w.r.t. footprint size), we require an approach to integrate the *product* of these spherical histograms with arbitrary incident illumination distributions (i.e., Equations 46 and 47).

4.2. Adaptive Basis-space Integration

A naïve integration solution is to apply a spherical quadrature at central axes of each spherical histogram bin, effectively a bin as a weighted delta function. Alternatively, a quadrature with prefiltered environmental summed area tables can be used, treating each bin as a subtended spherical rectangle. These integration approaches scale linearly with the histogram resolution, whereas we propose an *adaptive* method with several advantages:

- we adapt to the *frequency content* of the entire integrand in Equation (47), treating the bandlimits of *both* the NDF *and* the incident illumination and shading at the minimum of these bandlimits,
- our shading scales *indepedent* of the histogram resolution, and
- we do not need any specialized spherical SAT parameterizations, which would incur a resampling cost at shade-time.

To apply a basis-space shading method to our problem, we require a basis capable of efficiently:

- (1) determining the bandlimit of lighting, NDF and their product,
- (2) computing bandlimited integrals of this product (i.e., adaptivity),
- (3) capturing all-frequency signal variation (i.e., accuracy), and
- (4) having only modest memory requirements (i.e., compactness),

To these ends, we choose to employ spherical harmonics (SH). As with previous work from Han et al. [41], we detail several properties of SH that we leverage to meet our performance, accuracy and memory requirements; we contrast our choice and overall representations to that of prior work below and in Section 6.3. Basis selection involves trade-offs and, while other choices may have met our requirements (e.g., spherical radial basis functions), our SH method demonstrates significant performance and memory improvements over the state-of-the-art, and so we leave the application/exploration of other bases to this problem to future work.

Before detailing the interplay of our SH integration scheme with our spherical histogram representation, we provide a quick primer to SH and the specific properties we leverage.

4.2.1. Spherical Harmonic Preliminaries. We can represent a spherical function $f(\omega)$, with $\omega = (x,y,z) = (\theta,\varphi) \in S^2$, with SH projection coefficients obtained by projecting f onto the real SH basis as $\mathbf{f} = \int_{S^2} f(\omega) \mathbf{y}(\omega) d\omega$, where \mathbf{f} is a vector of these coefficients and $\mathbf{y}(\omega)$ is a vector of the individual SH basis functions:

$$y_l^m(\theta, \varphi) = \begin{cases} \sqrt{2} \ K_l^m \cos\left(m\varphi\right) \ P_l^m(\cos\theta) &, m > 0\\ \sqrt{2} \ K_l^m \sin\left(|m|\varphi\right) \ P_l^{|m|}(\cos\theta) &, m \le 0 \end{cases},$$
(48)

where an "order *N*" SH representation comprises bands $0 \le l \le N - 1 m$, indexes the (2l + 1) basis functions in each band *l*, K_l^m is a normalization term, P_l^m are the associated Legendre polynomials, and each band-*l* basis function is a degree *l* polynomials in (x,y,z). We often use a single index i = l(l+1) + m for basis functions and coefficients.

The signal *f* has bandlimit *M* if $f_l^m = 0$, $\forall l \ge M - 1$. We treat the concept of an *effective bandlimit M* when $|f_l^m| \le \varepsilon$, $\forall l \ge M - 1$ for sufficiently small ε (we use $\varepsilon = 10^{-6}$). We can reconstruct *f* by weighting the SH basis functions by its SH projection coefficients

$$f(\boldsymbol{\omega}) = \mathbf{f} \cdot \mathbf{y}(\boldsymbol{\omega}) = \sum_{l=0}^{N-1} \sum_{m=-l}^{m=l} f_l^m \, \mathbf{y}_l^m(\boldsymbol{\omega}) \tag{49}$$

and, unless f has bandlimit M < N, the reconstruction is a bandlimiting approximation of f (to a bandlimit of N.)

The m = 0 subset of SH functions, called *zonal harmonics* (ZH), are circularly symmetric functions of $cos\theta = z$. Sloan et al. [93] introduced a fast rotation to compute the SH coefficients g_l^m of a circularly symmetric function aligned about z (represented by ZH coefficients f_l) rotated to an *arbitrary* direction $\bar{\omega}$, by simply scaling the SH basis functions evaluated at $\bar{\omega}$ as

$$g_l^m = n_l^* f_l y_l^m(\bar{\omega}) = f_l^* y_l^m(\bar{\omega}) , \qquad (50)$$

with $n_l^* = \sqrt{4\pi/(2l+1)}$. This amounts to an application of the Funk-Hecke convolution theorem that allows us to obtain the SH coefficients of the function that results from a convolution of a circularly-symmetric function (with ZH coefficients f_l) and a spherical function (with SH coefficients h_l^m). Above, this amounts to setting $g(\omega) = f(\theta) \otimes h(\omega)$, where $h(\omega) = \delta(\omega - \bar{\omega})$ is simply a delta function in the direction of the rotation axis and *h* has SH projection coefficients $h_l^m = y_l^m(\bar{\omega})$.

The final SH property of interest is the *fast double-product integral* formulation: given two spherical functions $a(\omega)$ and $b(\omega)$ with (effective) band-limits M_a and M_b , the integral of their product is

$$\int_{\mathcal{S}^2} a(\boldsymbol{\omega}) b(\boldsymbol{\omega}) d\boldsymbol{\omega} = \sum_{i=0}^{M_{\star}^2 - 1} a_i b_i , \qquad (51)$$

where $M_{\star} = \min(M_a, M_b)$ and we arrive at the right hand side by substituting the SH expansions of *a* and *b* (Equation (49)) into the left hand side and then applying the orthonormality property of SH basis functions: $\int_{S^2} y_i(\omega) y_k(\omega) d\omega = \sigma_{i,k}$, where $\sigma_{i,k}$ is the Kroenecker delta.

Equations 49 and 51 satisfy requirements #1 and #2 for our basis (listed earlier in this subsection), and we address the final two basis requirements. Regarding all-frequency signal variation (item #3), while they are not often leveraged for all-frequency shading applications, SH basis functions are *capable* of capturing all-frequency signal variation if a sufficiently high bandlimit is chosen. We will discuss below exactly the manner in which we apply SH to our problem, as it scales independently to the normal/histogram map resolution and allows us to use extremely high bandlimits (order 300 in some cases), which are more than sufficient to capture the frequency content of NDFs (across scales) and realistic incident illumination from, e.g., environment maps (see Figure 39). While the number of SH coefficients we need to store grows quadratically in the max bandlimit order, we still maintain a memory footprint significantly smaller than the state-of-the-art (Yan et al. [113]), satisfying item #4, above (see Section 6.1).

4.2.2. Multi-scale SH NDFs and Adaptive Integration. We compute the SH projection of our filtered multi-scale NDF (Equation (47)) using the multi-scale NDF histogram we query for our filtering footprint \mathcal{P} and, after determining the effective bandlimit of this projection (and of the incident lighting; see Section 6), we set the shading bandlimit adaptively per-pixel as M_{\star} and apply Equation (51) to solve the integral (Equation (47)) of Equation (46).

Concretely, we arrive at the SH coefficients of the filtered multi-scale NDF $D_i(\mathcal{P}) = \int_{\mathcal{S}^2} \widehat{D}(\mathcal{P}, \omega) y_i(\omega) d\omega$ by summing the SH projection coefficients of its individual Gaussian roughness kernels, each aligned about the $N_{\mathcal{P}}$ normals in the filtering footprint \mathcal{P} ,

$$D_i(\mathcal{P}) = \frac{1}{N_{\mathcal{P}}} \sum_{\omega_x \in \mathcal{P}_{S^2}} \int G_s(\omega; \omega_x) y_i(\omega) \, d\omega.$$
(52)

We query our NDF histogram for \mathcal{P} (Figure 38) and iterate over the $N_{\mathcal{P}}$ micronormals in the histogram (outer sum in Equation (52)). SH projection coefficients for each of these individual Gaussians are precomputed once at initialization, as they align with the histogram bin central axes.

In fact, the histogram nature of our intermediate NDF representation allows us to further reduce the number of elements from $N_{\mathcal{P}}$ to M (with $M \ll N_{\mathcal{P}}$ as \mathcal{P} increases in size), where M is simply the resolution of the histogram.

Recalling that spatial integration of an individual Gaussian roughness kernel projects that Gaussian onto the sphere about the shading frame, so we take advantage of the fact that these spherical Gaussians are circularly symmetric and that their SH coefficients can be computed by applying the fast ZH rotation formulation (Equation (50)) to the ZH coefficients of a canonically-oriented spherical Gaussian (which we compute and store once, numerically). Computing the SH coefficients of $\hat{D}(\mathcal{P}, \omega)$ amount to a weighted sum of *M* ZH-rotated (precomputed) spherical Gaussian SH coefficients.

One could imagine avoiding the histogram abstraction and directly storing SH coefficients for incrementally accumulated NDFs, in a similar SAT-like structure as our histogram, but this approach has several limitations that we discuss in more detail in Section 6: of note, the storage costs of such an SAT would scale (quadratically) with the maximum SH order (e.g., **351 GB** for order-300 and a 1024×1024 normal map). This alternative also bears some similarities to the multi-scale NDF representation in Han et al.'s 2007 work [**41**], which we also discuss in Section 6.3.

5. Applications

After computing the SH coefficients of the filtered NDF \hat{D} we can solve Equation (46) with an efficient SH double product integral (Equation (51)). We do so in the context of two applications, detailing their specifics, below: filtered direct illumination (e.g., Yan et al. [112], Yan et al. [113], Jakob et al. [50]) and filtered global illumination (Belcour et al. [15]). In all cases, our rendering algorithms use an order-of-magnitude less time and less memory to match converged ground-truth compared to the state-of-the-art.

5.1. Appearance Filtered Direct illumination

We compute filtered direct illumination $L_d(\mathcal{P}, \omega_o)$ in a scene with complex environment and area lighting with Equations 46 and 47 and an appropriate substitution of $\widehat{L}_i(x, \omega_o, \omega_h) \equiv \widehat{L}_e(x, \omega_o, \omega_h) \widehat{V}(x, \omega_o, \omega_h)$ that accounts for the direct shadowing visibility term $V(x, \omega_i)$ and the incident radiance due to emission from light sources $L_e(x, \omega_i)$. Note that L_e combines lighting from area (e.g., polygonal), environmental and point/directional light sources.

Here, we note that we can *immediately* obtain a converged *unshadowed* direct illumination result $L_{\rm u}(\mathcal{P},\omega_o)$ with a single shading "sample" (i.e., a single evaluation of our basis-space double-product integral), if we ignore the visibility term and use $\widehat{L}_{\rm i}(x,\omega_o,\omega_h) \equiv \widehat{L}_{\rm e}(x,\omega_o,\omega_h)$. Here, we precompute the SH projection of $\widehat{L}_{\rm e}(x,\omega_o,\omega_h)(4\omega_h \cdot \omega_o)$ for many ω_o at initialization, and query these coefficients during

fast double-product integration for the unshadowed term L_u . We discuss technical details in Section 6.

A common numerical solution in scenarios such as this is to devise a Monte Carlo estimator that uses the unshadowed direct illumination as a *control variate*, and then relies on numerical sampling to resolve the shadowing:

$$L_{\rm d}(\mathcal{P},\omega_o) = L_{\rm u}(\mathcal{P},\omega_o) - \sum_{s=0}^{N_s} \frac{L_{\rm e}(x,\omega_s) \left(1 - V(x,\omega_s)\right) \widehat{f_r}(\mathcal{P},\omega_s,\omega_o)}{N_s \operatorname{pdf}(x,\omega_s)}$$
(53)

where we rely on our baseline G×D technique to resolve the Monte Carlo integral, with a spherical pdf that applies multiple importance sampling of the light/environment and our filtered BRDF sampling. Spatial sampling (i.e., to resolve the implicit spatial integral over the footprint of \hat{f}_r) is absorbed into standard sub-pixel anti-aliasing.

One side effect of this control variate estimator is that noise in the shadowed regions can become objectionable, since individual estimator samples are unbounded. We rely instead on the recent *ratio estimator* approach of Heitz et al. [44], that leverages a factored *multiplicative* decomposition of the unshadowed and shadowed direct radiance, instead of the difference-based estimator above. Unlike their approach, we do not employ any spatial denoising to our ratio estimate, to avoid adding bias; doing so would significantly improve the visual appearance of our results at low sampling rates, but our goal is to match ground-truth results.

5.2. Filtered Global Illumination

We adapt the method of Belcour et al. [15] to filter high-frequency appearance in the presence of global illumination effects, but now also in scenes with complex environmental and area light



Unshadowed $L_u(\mathcal{P}, \boldsymbol{\omega}_o)$

Shadowed $L_{d}(\mathcal{P}, \omega_{o})$

Fig. 40. We can generate converged unshadowed direct illumination with a single shading sample (left; 0m:42s) and then apply the technique of Heitz et al. [44] to compute shadowing numerically (with 4spp in 1m:08s).

sources. Their approach propagates a lightweight covariance-based lightfield representation along paths (i.e., in a standard uni- or bi-directional path tracing framework). The covariance at any path vertex can be used to define a local filtering footprint which, in turn can be applied to filter an underlying high-frequency appearance model.

Even in scenes with strong point and directional sources, their indirect filtered appearance method provided several orders-of-magnitude improvements over a naïve path traced solution (the only alternative, at the time). The added complexity of filtering indirect reflections of high-frequency appearance *in the presence of* large angularly-varying light sources stresses their method as, just with filtered appearance in direct illumination, their underlying appearance filtering model does not take variation due to incident illumination into account.

By applying our solution to Equation (46) in the next-event estimator of secondary path vertices in a standard uni-directional path tracer, augmented with Belcour et al.'s secondary footprint computation, we are effectively leveraging our method's ability to compute a *converged* value that integrates over all light directions (instead of just one). This way, we are able to outperform their approach in scenes with complex lighting by about a factor of $10 - 50 \times$ (see Figures 33 and 46).

6. Implementation and Discussion

We discuss implementation details, as well as discussing design decisions in the context of previous work, below.

At a high-level, our approach can build atop any rendering engine capable of generating ray differentials (see Igehy [47]) (for direct illumination) and, for global illumination, secondary path filtering differentials (we employ Belcour et al.'s [15] approach, here). As with previous work (Yan et al. [112, 113]), we assume axis-aligned footprints in texture-space, computed with conservative (u,v) bounding boxes.

Below, we provide technical implementation details of our data-structures and algorithm, and discuss relationships (and differences) to certain aspects of previous work. Our rendering framework involves the construction of our histogram SAT (Section 6.1), the precomputation of rotated Gaussian SH lobe coefficients (Section 6.2) and the precomputation of weighted SH lighting coefficients (Section 6.4).

6.1. Histogram Resolution

We store a discrete spherical histogram in our SAT-like multi-scale NDF lookup texture (Section 4.1). The texture itself has the same resolution as the underlying normal map, and each texel stores a discretized histogram with (θ, φ) -parameterized bins. We experimented with different histogram resolutions, and settled on a resolution of 9×32 for every result in the paper: this value was chosen such that renderings visually match the reference ground-truth, all while maintaining a

modest memory footprint (e.g., 576MB for 1024×1024 normal maps and 1.1GB for 2048×2048). Figures 43 and 47 illustrate visual artifacts that arise from reducing the histogram bin resolution.

6.2. Gaussian Roughness Lobe Discretization

We precompute SH coefficients for the rotated ZH Gaussian roughness lobes $(\int G_s(\omega; \omega_x) y_i(\omega) d\omega$ in Equation (52)), for a fixed base roughness σ_s (and each high-frequency material in the scene), at start-up. We store these rotated lobes at a 4× *higher resolution* (i.e., 65 × 128) than the histogram due to the difference in their spherical parameterizations: the NDF histograms are expressed in the *local-shading frame* (with *n* up), whereas shading with the Gaussian lobes is conducted in the *half-vector frame* (with ω_o up). This means that we need to rotate the *n*-oriented Gaussian lobes to the ω_o -frame for shading. At run-time, we rotate each (non-zero) NDF bin's central direction into the ω_o -frame and bilinearly interpolate between the four nearest pre-rotated Gaussian lobe coefficient vectors. Alternatively, we could perform a *perfect* rotation



Fig. 41. We adapt the technique of Belcour et al. [15] to compute secondary path vertex footprints for filtered appearance with global illumination.



Fig. 42. When integrating with respect to ω_h our basis-space approach automatically discards NDF normals outside the shading frame (grayscale) and uses an appropriately warped incident illumination distribution.

of a canonically-oriented Gaussian ZH lobe to the rotated bin direction (Equation (50)), but the precomputed solution allows us to avoid evaluating SH basis functions at run-time (whose number grows quadratically w.r.t. order). We illustrate the effects of different lobe direction sampling rates (relative to the NDF histogram resolution) on a simple smooth microfacet sphere, rendered in the half-vector space, in Figure 44. The effect of different rates can be observed in Figure 48.



Fig. 43. Lower resolutions for our accumulated NDF histogram result in angular blurring of details (left), albeit a much faster rendering $(2.5 \times, \text{here})$.

	Normal	Max SH	Total	Ren	Speed-up	
Scene	Mon	Ordor	Momory	Our Method	Yan et al. [113] or	Easter
	wiap	Oldel	Memory		Belcour et al. [15]	Factor
Cutlery	2048^2	60	2.7GB	50.7s	7.1m	8×
Snails	2048^2	20	2.3GB	3.53m	121.4m	$34\times$
Torus	512 ²	15	172MB	4.89s	2.35m	$28.8 \times$
Kettle $(3 \times)$	2048^2	60	2.7GB	1.22m	-	-
Kettle	1024^2	60	1GB	2.43m	-	-

 Table 2. Memory and performance comparisons for our examples.

The precomputed lobe coefficients still require very little storage: 29MB for order-30 SH, 114MB for order-60 and 714MB for order-150.

6.3. Naïve NDF SAT and Relationship to Han et al. [41]

We combine a SAT-like NDF histogram representation with rotated Gaussian lobes in order to compute the SH coefficients of a multi-scale NDF, for a certain filtering footprint. An alternative approach would be to *directly* store SH coefficients of the accumulated NDF, in an SAT-like data structure; then, when querying for a specific footprint's NDF's SH coefficients, we can similarly apply an SAT-like 4-query sampling (as in Figure 38). At a high-level, this is similar to the data structure proposed by Han et al. [41], with the main differences being that they leverage MIP-hardware to compute a hierarchy of footprints, and then use a single texture sample to query a specific footprint's NDF's coefficients.

We implemented both of these strategies and, in both cases, there are significant performance and scalability constraints that preclude their use for high, ground-truth-quality renderings with high-resolution appearance maps. Firstly, memory-wise, storing SH coefficients per texel quickly becomes prohibitive as the normal map resolution increases, and the maximum SH order \mathcal{F}_{max} increases: a 2048 × 2048 normal map and $\mathcal{F}_{max} = 100$ would require **more than 160GB** of storage for single-precision floats. And if a MIP-hierarchy is employed, instead of an SAT lookup approach, then 213GB would be needed.

Apart from storage, another downside is that the NDF coefficients would be expressed in the *n* local shading frame (the same as the NDF histogram parameterization) and, in order to shade with realistic microfacet models in the half-vector ω_h frame, these coefficients need to be rotated (for each pixel sample) at run-time, which is costly in SH (see Nowrouzezahrai et al. [77]).

We instead decouple memory costs due to increasing SH order from the asymptotic memory costs, as increases in the SH order only affect the size of our pre-rotated Gaussian lobe coefficients; and, as discussed earlier, we *could* completely avoid this precomputation by performing the fast ZH rotation on-the-fly. This decoupling also allows us to avoid costly SH rotations when changing between the local shading frame and the half-vector rendering frame.

6.4. Light Source Coefficients

When computing Equation (46) as a double product integral, we require the SH coefficients of the filtered multi-scale NDF (i.e., the rotated Gaussian NDF lobes) and the incident illumination (both expressed in half-vector space). Rewriting Equation (47) to explicitly highlight the double product integral decomposition, we have

$$\int_{\Omega_{h}} \left[\widehat{L}_{i}(\bar{x}, \omega_{o}, \omega_{h}) \left(4(\omega_{h} \cdot \omega_{o}) \right) \right] \left[\widehat{D}(\mathcal{P}, \omega_{h}) \right] d\omega_{h}$$

$$= \int \left[\sum_{j} \underbrace{\int \widehat{L}_{i}(\bar{x}, \omega_{o}, \omega) \left(4(\omega \cdot \omega_{o}) \right) y_{j}(\omega) d\omega}_{L_{i}^{SH}(\omega_{o})} \right] \left[\sum_{k} \underbrace{D_{k}(\mathcal{P})}_{Eq. 52} \right] d\omega_{h},$$

where the lighting SH coefficients L_j^{SH} , when expressed in the half-vector space, depend explicitly on the viewing direction ω_o . These coefficients also include the change-of-parameterization Jacobian.

Similarly to our discretization of the Gaussian roughness lobe coefficients, we pre-tabulate SH coefficients for the lighting, discretized at the same resolution as the Gaussian lobes (and also bilinearly interpolated at run-time; see Figure 49). This table requires three-times (or as many spectral components used) the storage as the Gaussian lobe coefficient table, due to separate RGB coefficients for lighting (so, e.g., just under 342MB for order-60).



0.5 imes histogram resolution

 $4 \times$ histogram resolution

Fig. 44. When storing rotated Gaussian roughness SH coefficients below the spherical histogram resolution (left), visible re-sampling artifacts can appear. We use a conservative discretization of $4 \times$ the histogram's resolution.

7. Results

We report render times from a dual Intel Xeon E5-2683 with 32 cores and 128 GB of RAM, and normal maps have a resolution of 2048×2048 , unless stated otherwise. In all cases, render times do not include renderer initialization, a process that typically takes roughly 4 seconds for our scenes. In addition to this initialization cost, our method requires no more than 9 seconds to construct all of its internal data structures. As discussed in Section 6, the memory consumption of our approach depends on storing the spherical NDF histograms, the rotated Gaussian roughness lobe coefficients, and the rotated lighting coefficients. The latter of these two can be shared across materials, require negligible memory compared to the spherical NDF histogram texture, and require a short precomputation time dependent on projection quality, roughly 7 minutes for all 65 × 128 order 60 projections, each computed with half a million numerical integration samples. Figures 47-50 illustrate the effect of our various simplifications and parameter settings. Table 2 summarizes our memory usage, and we discuss each scene below.

The **snails** scene uses the flakes normal map (see Yan et al. [**112**, **113**]), an environment map and a single point light, simulating global illumination with a single indirect bounce. Our method obtains a converged result (see supplemental video) in 2m:15s, and Belcour et al. 2017's approach [**15**] generates 11spp in equal time.

The **cutlery** scene uses a scratched metal normal map (see Yan et al. [**112**, **113**]) with environmental and point lighting, and our method converges in 0m:51s, whereas the method of Yan et al. [**113**] can only generate 9 spp in equal time (far from enough to convergence).

The **torus** scene uses a 512×512 scales normal map that is tiled repeatedly in the scene, with lighting from a lower-frequency environment map. This allows us to use significantly lower effective shading bandwidths (see Section 4.2) and we obtain a converged rendering in less than 5 seconds, whereas the state-of-the-art (Yan et al. [**113**]) rendered a single sample in 6 seconds (and our baseline G×D fired 2spp in equal time). Fully converged renderings for these two alternative baselines took 2m:25s and 13m:8s, respectively.

For the **kettle**, all lighting in this scene is due to an environmental source and and we use a metal panel 1024×1024 normal map. We render this scene to convergence in 2m:27s.

8. Conclusion and Future Work

We present a scalable appearance model for filtered reflections from high-frequency microfacet models under complex illumination. We demonstrate significant speed-ups, up to $10 \times$ for direct illumination and $50 \times$ for global illumination, compared to the state-of-the-art in these two scenarios (Yan et al. [113], Belcour et al. [15]). Our method includes limitations due to our assumptions, which we discuss shortly.

Axis-aligned footprints are inherent to an SAT-like data structure, and as such, susceptible of being extended to arbitrary convex footprints (see Piponi [83]).

Light source coefficients are precomputed once, per environment map. Area lights require on the fly SH coefficient calculation (see Wang et al. [109] and Belcour et al. [14]). An interesting direction for future work is to obtain analytical expressions (e.g., for spherical sources) that are suitable for half-vector space shading.



Yan et al. [113]



Ours

Fig. 45. Cutlery scene using the same environment map and point light, shadowed direct lighting only. Our method approached the ground-truth in 50.7s while Yan et al. [**113**] required 7m:6s to reach the same quality at 2116 samples per pixel.



(c) Snails scene, direct only

(d) Snails scene, indirect only

Fig. 46. Additional results for both direct and global illumination. Both (**a**) and (**b**) showcase additional models and environment maps with very high fidelity lighting while remaining temporally stable and fast to compute, 2m:27s and 1m:13s, respectively. Figures (**c**) and (**d**) are a breakdown of a frame from the Snails scene, showcasing the performance of our method across both direct and first-bounce indirect illumination in a global illumination context. In both cases, our results handily beat Belcour et al. [**15**] at equal time (11 spp).

For **F** and **G**, an alternative to our decoupling term \overline{FG} is to compute a triple product integral using *tripling coefficients*. This would increase the complexity of our method with negligible benefit, depending on the scenario.

An interesting avenue of future work is to explore interactive approximations of our method. Initial experiments show that a naïve port of our direct integrator to a shader-based GPU renderer yields performances on the order of 1Hz at a resolution of 1600×900 on an NVIDIA GTX 1080 (with 8GB of vRAM) for the cutlery scene from Figure 33. More aggressive data compression, a better trade-off between data accesses and compute (e.g., rotating the canonical ZH Gaussian roughness lobes by directly computing fast ZH rotations), and perhaps an alternative spherical basis (e.g., spherical RBFs) could increase performance into the realm of real-time applications.



Fig. 47. We rely on a spherical histogram that discretizes the upper hemisphere into bins (see Section 4.1). The resolution of this histogram directly impacts the minimally reproducible detail size: lower resolutions progressively lose detail in the normal map, effectively smoothing it out. Our selected setting (9×32) is adequate for all scenes we have tested, when compared to the ground truth.



Fig. 48. Impact of the resolution of the rotated ZH Gaussian roughness lobes (see Section 6.2). As is apparent, the resolution does not have a significant effect on image quality for the resolutions we tested. We use a conservative setting of 65×128 .



Fig. 49. Impact of the resolution of the environment light's SH coefficients table (see Section 6.4). As with Figure 48, the effect of a lower resolution is minor. We again conservatively use 65×128 .



Fig. 50. Impact of the chosen band limit for the NDF or light spherical harmonics. Band limiting either is equivalent. Artifacts start to appear when either band limit is chosen below the true band limit of the respective signal (i.e., the maximum of the band limit of the NDF and the environment map). For this scene, Order 60 was sufficient.

Acknowledgements

We would like to thank Wenzel Jakob, Ling-Qi Yan and Laurent Belcour for source code of their implementations. Computing resources provided by Calcul Québec and the National Systems of



Decoupled FG

Reference

Fig. 51. Effect of the simplification made in Equation (46) and the proposed alternative in Equation (54). Both images were generated using our reference $G \times D$ implementation with sufficient sampling to match ground-truth.

Compute Canada. Funding for this research was made possible by Consejo Nacional de Ciencia y Tecnología (CONACYT) of Mexico and the Natural Sciences and Engineering Council of Canada.



Equation (54)

Product-of-integrals

Ground-truth

A. Fresnel and Geometric Term Approximation

As is common in existing interactive and offline rendering models, we apply a factorization of the Fresnel and Geometry terms of our filtered shading model in Equation (46): $\overline{FG}(\omega_o)$. One possible approximation would be to precompute and apply an "ideal" two-factor *integral-of-products* equals product-of-integral as $\overline{FG}(\omega_o) = 1/(4(n \cdot \omega_o)) \int_{\Omega_i} F(\omega_i \cdot \omega_h) G(\omega_i, \omega_o) d\omega_i$, but we opt for a simpler alternative.

We instead evaluate the product of the Fresnel and Geometry terms in the view direction and the mirror reflection of the view, as

$$\overline{FG}(\boldsymbol{\omega}_o) = \frac{F(\boldsymbol{\omega}_r \cdot n) G(\boldsymbol{\omega}_r, \boldsymbol{\omega}_o)}{4(n \cdot \boldsymbol{\omega}_o)},$$
(54)


Fig. 52. Equal quality comparison for global illumination in the snails scene. Belcour et al. [**15**] took 121m:24s to render at 2048 samples per pixel while our method finished in 3m:32s.

where $\omega_r = 2(n \cdot \omega_o)n - \omega_o$ is the mirrored view vector. The figure below illustrates the nature of this approximation, compared to the more costly integrated factorization, above.

Third Article.

A Frequency Analysis and Dual Hierarchy for Efficient Rendering of Subsurface Scattering

by

David Milaenen¹, Laurent Belcour², Jean-Philippe Guertin³, Toshiya Hachisuka⁴, and Derek Nowrouzezahrai⁵

- (1) Weta Digital, Université de Montréal
- ⁽²⁾ Unity Labs, Université de Montréal
- (3) Université de Montréal
- ⁽⁴⁾ University of Tokyo
- ⁽⁵⁾ McGill University, Université de Montréal

This article was presented at *Graphics Interface* on May 31, 2019 in Kingston, Ontario. Formatting has been adapted for this thesis and minor typographical issues were corrected.

I am the third author of this article. My main contributions are:

- Elaboration and integration of the dual-tree within the algorithm to provide a substantial speedup versus traditional "single-tree" approaches.
- Implementation of some features related to the dual-tree.
- Final rewrite and validation of the article for its submission at GI.

David Milaenen implemented most of the algorithm and contributed to its elaboration. Laurent Belcour devised the frequency analysis elements of the algorithm. Toshiya Hachisuka participated in the dual-tree portions of the algorithm's conception. Derek Nowrouzezahrai helped bring the elements together into a cohesive algorithm and supervised the work.

This article was a collection of elements working in unison to provide an overall significant speedup. I have specifically participated on the dual-tree elements of the algorithm both from a theoretical and technical level. In addition, David Milaenen, who spearheaded the article, unfortunately concluded his degree and ended his work before the article was accepted in a venue. I was the one to pick it back up, perform a cleanup and rewrite pass, and bring it to publication.

RÉSUMÉ. Les BSSRDFs sont communément utilisées pour modéliser le transport de la lumière sous la surface dans des matériaux tels le marbre ou la peau. Le rendu de BSSRDFs requiert une intégration spatiale supplémentaire, ce qui peut être significativement plus coûteux qu'un rendu de surface seulement avec des BRDFs. Nous introduisons une méthode novatrice de rendu hiérarchique qui peut mitiger ce coût additionnel d'intégration spatiale. Notre méthode a deux composantes clés: une analyse fréquentielle du transport de la lumière sous la surface, et une hiérarchie duale sur les échantillons d'ombrage et d'illumination. Notre analyse fréquentielle prédit la variation spatiale et angulaire de la radiance sortante due au BSSRDF. Nous utilisons cette analyse pour guider une intégration adaptable spatiale avec des échantillons d'image et d'illumination épars. Nous proposons l'usage d'une structure d'arbre dual qui nous permet de traverser simultanément un arbre d'échantillons d'ombrage (c'est-à-dire, de pixels) et un arbre d'échantillons d'illumination en espace objet. Notre approche d'arbre dual sont toutes deux compatibles avec la plupart des modèles de BSSRDF, et nous démontrons que notre méthode améliore les temps de rendu comparativement à la méthode à la fine pointe de Jensen et Buhler [54].

Mots clés : modélisation de réflectivité, tracé de rayon, rendu

ABSTRACT. BSSRDFs are commonly used to model subsurface light transport in highly scattering media such as skin and marble. Rendering with BSSRDFs requires an additional spatial integration, which can be significantly more expensive than surface-only rendering with BRDFs. We introduce a novel hierarchical rendering method that can mitigate this additional spatial integration cost. Our method has two key components: a novel frequency analysis of subsurface light transport, and a dual hierarchy over shading and illumination samples. Our frequency analysis predicts the spatial and angular variation of outgoing radiance due to a BSSRDF. We use this analysis to drive adaptive spatial BSSRDF integration with sparse image and illumination samples. We propose the use of a dual-tree structure that allows us to simultaneously traverse a tree of shade points (i.e., pixels) and a tree of object-space illumination samples. Our dual-tree approach generalizes existing single-tree accelerations. Both our frequency analysis and the dual-tree structure are compatible with most existing BSSRDF models, and we show that our method improves rendering times compared to the state of the art method of Jensen and Buhler [**54**].

Keywords: reflectance modeling, ray tracing, rendering

I. Illumination sampling II. Predicted sampling rate



III. Clustered pixels IV. BSSRDF contribution

Fig. 53. We introduce a hierarchical method to accelerate the rendering of multiple scattering with BSSRDFs (IV). We overview our approach in the PICNIK scene, above: our frequency analysis of BSSRDFs allows us to predict the screen-space sampling rates (II) which are used to devise bounds on the variation of outgoing radiance. These bounds allow us to efficiently integrate the BSSRDF using a dual hierarchy over clustered illumination samples (I) and shading points (i.e., pixels; III).

1. Introduction

Including subsurface scattering effects in virtual scenes can significantly increase the realism of rendered images. Since many real-world materials exhibit subsurface scattering effects, modeling and simulating them remains an important problem in realistic image synthesis.

Accurate light transport in highly absorbing media can be modeled mathematically with the Bidirectional Scattering Surface Reflectance Distribution Function (BSSRDF). Many BSSRDF models exist with varying degrees of accuracy: classical dipole models (Jensen et al. [55], d'Eon [24]) and quantized diffusion (d'Eon and Irving [26]) do not account for the angular variation of incident radiance; however, more recent models do (Frisvad et al. [35], Habel et al. [39], d'Eon [25]). Unlike BRDFs, BSSRDFs describe light transport between two *different* locations on an object. As such, an additional spatial integration (over the surface) is required in order to render objects with BSSRDFs. Jensen and Buhler [54] introduced an adaptive hierarchical integration method to amortize the cost of this spatial integration using clusters of spatial illumination samples. While this approach has been successfully used in many applications, it does not take the smoothness of the resulting outgoing radiance (i.e., in screen-space) into account.

We propose a novel integration method that clusters **both** pixels and illumination points as illustrated in Figure 53. We conduct a frequency analysis of subsurface scattering that is agnostic to the underlying BSSRDF model. Specifically, we study the frequency content of the spatial and angular variation of radiance after its BSSRDF interaction. This leads us to a theoretically sound criterion for sparse sampling and adaptive integration. Using this criterion, we leverage a dual hierarchical data structure to accelerate the final evaluation of the multiple scattering term. Our hierarchical evaluation is motivated by the existing tree-based approach of Jensen and Buhler [54]; our dual-tree structure, however, amortizes computation cost across both pixels *and* illumination



Fig. 54. We sample incident illumination over the object (a) according to its subsurface scattering properties and construct two spatial acceleration structures: one over these samples (c) and one over pixels (d). To render, we simultaneously traverse the trees (e), using our outgoing radiance bandwidth estimate s_p (b) to stop the tree traversal and shade super-pixels of area A.

points. We are able to generate higher-quality results in less rendering time compared to the single tree method of Jensen and Buhler [54]. Concretely, we propose:

- a frequency analysis of shading with BSSRDFs,
- a numerical approach for estimating the BSSRDF spectra, which we use to determine the variation of outgoing radiance over the surface of a translucent object, capable of supporting any underlying dipole model, and
- the application of a dual-tree structure to the problem of BSSRDF estimation in joint imageand object-space, directly leveraging our frequency analysis to adaptively traverse the structure and accelerate the final rendering.



Fig. 55. Assuming that the incoming light-field has infinite bandwidth, we estimate the bandwidth of the outgoing light-field $[B_s, B_\theta]$ as the bandwidth of the BSSRDF along the outgoing spatial positions and directions (a). The interaction with the material limits the spectrum of the local light-field by the BSSRDF spatial and angular bandwidth (b). To estimate the bandwidth at the camera position, we first shear spatially the spectrum to account for curvature (c). Then, we scale spatially to account for foreshortening (d) and finally shear angularly the spectrum to account for transport (e).

2. Previous Work

We focus on work that most closely align with our approach: specifically, we review integration schemes for BSSRDF models, and frequency analyses of light transport.

BSSRDF Integration Techniques. In all cases, the bottleneck of dipole-like techniques remains the numerical evaluation of the spatial-angular integration in Equation (55). Jensen and Buhler [54] compute an approximate evaluation of this contribution from sparse irradiance samples distributed over a translucent object's surface. Here, the outgoing radiance at any shade point is computed by traversing a tree over the irradiance samples and terminating traversal according to a quality criterion. This two-pass approach introduces a controllable bias and remains compatible (often without modification) with many of the newer dipole models we discussed in Section 1. Notably, Frisvad et al. [35] need only substitute the (diffuse) irradiance samples with a vector of differential irradiance samples, and d'Eon and Irving [24] use a supplemental 1D radial directional radiance bin.

We are motivated by the lack of techniques that fully leverage image-space coherence to reduce the computation time of rendering translucent materials. Approaches based on LightCuts (Walter et al. [107]) fail to either efficiently treat BSSRDFs or amortize computation cost across similar pixels. Arbree et al. [5] propose a scalable approach to rendering large translucent scenes, based on multidimensional LightCuts (Walter et al. [106]), aggregating the computation of irradiance samples by simultaneously clustering lights and irradiance samples. Their clustering is designed to approximate the resulting contribution at a given shade point. While this method also uses two trees, it treats each pixel independently and does not take the resulting image smoothness into account (see Figure 4 and Section 4.1 of Arbree et al. [5]). We do not consider the evaluation cost of (ir)radiance samples, but we do cluster evaluation over pixels. In contrast, multidimensional-LightCut methods, such as IlluminationCut (Bus et al. [17]), could (in theory) be extended to BSSRDF shading but, in doing so, would require a prohibitive number of BSSRDF evaluations to determine their error threshold; indeed, we implemented such an extension of IlluminationCut to validate this claim (see Figure 56). On the other hand, our technique can also be used to provide a frequency-based cut threshold specifically designed for BSSRDFs, all while avoiding any explicit evaluation of the BSSRDF model.

The idea of applying a doubly-adaptive traversal originates from the particle simulation literature (Greengard and Rokhlin [36]), and the implementation of d'Eon and Irving's quantized diffusion model (d'Eon and Irving [26]) in Pixar's RenderMan implicitly leverages a similar principle (i.e., with REYES' adaptive micropolygon evaluation). One of our contributions is a well-founded oracle to terminate shading tree traversal based on our BSSRDF frequency analysis. Similarly, Jarabo et al. [52] leverage trees over virtual point lights and shading points without explicitly using dual trees,

but their algorithm is specifically restricted to VPLs and diffuse materials, making the transition to translucent surfaces difficult.

Frequency Analyses of Light Transport. Durand et al. [**32**] present the first comprehensive Fourier analysis of light transport in scenes with opaque surfaces, and a proof-of-concept adaptive image space sampling approach to reconstruct noise-free images at super-pixel sampling rates. Dubouchet et al. [**29**] use frequency analysis to construct a sampling cache which improves efficiency when rendering animations using distant direct lighting. Bagher et al. [**9**] derive atomic operators for bandwidth estimation in order to study environmental reflection with acquired BRDFs. Belcour et al. extend these frameworks to incorporate the study of defocus and motion blur (Belcour et al. [**13**]), scattering in arbitrary participating media (Belcour et al. [**11**]), and for global illumination (Belcour et al. [**15**]), but do not directly tackle dense media or BSSRDFs. We bridge this gap with a frequency analysis of scattering in dense media, similarly leveraging matrix-vector formulations of frequency-space bandwidth operators.



Fig. 56. We compare IlluminationCut (Bus et al. [17]) to the method of Jensen and Bulher [54] on the BUNNY scene with various ε error bound settings. The cost of computing the upper-bound metric (Eq. 13 of Walter et al. [106, Eq. 13]), which requires multiple BSSRDF evaluations, precludes the direct applicability of IlluminationCut to adaptive BSSRDF shading.

2.1. Overview

Figure 54 overviews our approach: after sparsely evaluating incident radiance on the surface of each translucent object (Figure 54a), we compute a per-pixel bandwidth estimate of the multiply-scattered outgoing radiance (Figure 54b). We build two spatial acceleration structures, one over illumination samples (Figure 54c) and another over pixels (Figure 54d). In order to compute the object's final shading, we simultaneously traverse both trees, hierarchically accumulating the contribution of groups of illumination samples to groups of pixels (Figure 54e). We use the frequency bandwidth of the outgoing radiance predicted by our theory (Section 3) to terminate traversal along each tree, significantly reducing the number of BSSRDF evaluations necessary to compute the final image without introducing visible artifacts.

We present our BSSRDF frequency analysis theory, as well as its numerical realization for computing image-space radiance bandwidths in Section 3. We introduce our variant of the dual tree construction and how the bandwidth predictions are used during hierarchical traversal in Section 4. Finally, we discuss our implementation details in Section 5 and compare our method to the state of the art in Section 6.

3. Fourier Analysis

We will derive conservative, numerical estimates of the frequency bandwidth of the outgoing radiance in image space, taking into account the effects of curvature, foreshortening, transport and multiple scattering on the incident light-field's frequency content. We will show that the BSSRDF acts as a band-limiting filter on the incident radiance distribution, and we will derive an expression of the resulting spatio-angular bandwidth of the outgoing radiance spectrum (Section 3). We will use these bandwidth estimates, combined with the formulation of Bagher et al. [9], to predict the variation of outgoing radiance in image space (Section 3.2), which will in turn drive our hierarchical dual tree traversal and integration (Section 4).

3.1. Fourier Transform of a BSSRDF

Given a BSSRDF model $S(\mathbf{x}_i, \boldsymbol{\omega}_i, \mathbf{x}_o, \boldsymbol{\omega}_o)$, the outgoing radiance at the object surface L_o in direction $\boldsymbol{\omega}_o$ and at position \mathbf{x}_o is expressed as:

$$L_o(\mathbf{x}_o, \boldsymbol{\omega}_o) = \iint_{\mathcal{A} \times \mathcal{H}} S(\mathbf{x}_i, \boldsymbol{\omega}_i, \mathbf{x}_o, \boldsymbol{\omega}_o) L_i(\mathbf{x}_i, \boldsymbol{\omega}_i) d\boldsymbol{\omega}_i^{\perp} d\mathbf{x}_i,$$
(55)

where \mathcal{A} is the object's surface area, \mathcal{H} is the set of (hemispherical) incident directions, L_i is the incident radiance, and $d\omega_i^{\perp} = \cos \theta_i d\omega_i$ is the projected solid angle.

If we apply a Fourier transform to Equation (55), converting products in the primal domain to convolutions in the frequency domain and integration in the primal domain to DC evaluation in the

frequency domain, we obtain:

$$\mathcal{F}[L_o](\Omega_{\mathbf{x}_o}, \Omega_{\boldsymbol{\omega}_o}) = \left[\mathcal{F}[\widehat{S}] \circ \mathcal{F}[L_i]\right](0, 0, \Omega_{\mathbf{x}_o}, \Omega_{\boldsymbol{\omega}_o}),$$
(56)

where $\mathcal{F}[f]$ is the Fourier transform of f, \circ the convolution operator, and Ω_x the frequency variation of x. Concretely, the outgoing radiance's spatial-angular frequency spectrum $\mathcal{F}[L_o](\Omega_{\mathbf{x}_o}, \Omega_{\omega_o})$ results from evaluating the convolution of the Fourier transform of the cosine-weighted BSSRDF $\mathcal{F}[\widehat{S}] = \mathcal{F}[S(\mathbf{x}_i, \omega_i, \mathbf{x}_o, \omega_o) \cos(\theta_i)]$ with the Fourier transform of the incident light $\mathcal{F}[L_i]$ at the incoming spatial and directional DC frequencies $(\Omega_{\mathbf{x}_i}, \Omega_{\omega_i}) = (0,0)$.

Assuming that $\mathcal{F}[L_i]$ contains all-frequency content, the resulting outgoing bandwidth (along $\Omega_{\mathbf{x}_o}$ and Ω_{ω_o}) after convolution against the spectrum of the cosine-weighted BSSRDF $\mathcal{F}[\widehat{S}]$ will match the bandlimit of $\mathcal{F}[\widehat{S}]$ (see Figure 55a). We will discuss how to compute the spatial and angular bandwidths $\{B_o, B_\theta\}$ of the cosine-weighted BSSRDF given its local orientation.

Spatial Bandwidth. We compute the cosine-weighted BSSRDF's spatial bandwidth numerically by sampling and projecting $S(\mathbf{x}_i, \boldsymbol{\omega}_i, \mathbf{x}_o, \boldsymbol{\omega}_o) \cos(\theta_i)$ into the frequency domain, across its different dimensions. Depending on the underlying BSSRDF model, the cosine-weighted BSSRDF may depend on the viewing direction, the incident lighting direction, and the distance and angle between \mathbf{x}_o and \mathbf{x}_i .

For instance, the dipole model has a separable form:

$$\mathcal{F}[\widehat{S}] = \mathcal{F}[R_d(||\mathbf{x}_i - \mathbf{x}_o||) F_i(\theta_i) \cos(\theta_i) F_o(\theta_o)] ,$$

where R_d is the diffuse reflectance, and F_i and F_o are the incident and outgoing Fresnel terms [55, Equation 5]. Here, we take advantage of the separability of the model (w.r.t. θ_i and θ_o) to express its Fourier transform as

$$\mathcal{F}[\widehat{S}] = \underbrace{\mathcal{F}[R_d(||\mathbf{x}_i - \mathbf{x}_o||)] \mathcal{F}[F_i(\theta_i) \cos \theta_i]}_{\mathcal{F}[\widehat{S}_i](\Omega_{\mathbf{x}_i}, \Omega_{\mathbf{x}_o}, \Omega_{\omega_i})} \mathcal{F}[F_o(\theta_o)] .$$

Since we are only concerned with the DC $[\Omega_{\mathbf{x}_i}, \Omega_{\omega_i}] = [0,0]$ hyperplane, the spatial bandwidth is computed with the 1D diffuse reflectance spectrum $\mathcal{F}[R_d](\Omega_{x_o})$. We discuss the outgoing term $\mathcal{F}[F_o](\Omega_{\omega_o})$ below.

In contrast, the directional dipole (Frisvad et al. [35]) additionally takes ω_i and the direction between \mathbf{x}_i and \mathbf{x}_o into account:

$$\mathcal{F}[\widehat{S}_i] = \mathcal{F}\left[\frac{e^{-\sigma_{tr}||\mathbf{x}_i-\mathbf{x}_o||}}{4\pi^2||\mathbf{x}_i-\mathbf{x}_o||^3}M(\mathbf{x}_i-\mathbf{x}_o,\boldsymbol{\omega}_{12}) F_i(\boldsymbol{\theta}_i)\cos\boldsymbol{\theta}_i\right],$$

where $M(\mathbf{x}_i - \mathbf{x}_o, \omega_{12})$ models the spatial-directional scattering distribution and ω_{12} is the refraction of ω_i at \mathbf{x}_i [35, Equation 17]. We extract the outgoing spatial bandwidth by taking the maximum 1D bandwidth for various angles between $\mathbf{x}_i - \mathbf{x}_o$, the normal at \mathbf{x}_i and the refracted ray ω_{12} . In all instances, we compute a conservative — that is to say, such that our derived frequency domain bounds are strictly larger than the true underlying bounds — estimate of the outgoing spatial and directional frequency *bandwidths*, B_s and B_{θ} , as the values required to retain 95% of the energy of the discrete power Fourier spectrum.

Angular Variation. The angular variation of the BSSRDF is modulated by the outgoing Fresnel term above, and we use a windowed Fourier transform to compute the bandwidth of $\mathcal{F}[F_o](\Omega_{\omega_o})$, again as the 95th energy percentile spectrum value. We tabulate these bandwidths as a function of θ_o , and use them to modulate B_θ ; this is particularly important at grazing angles, where the effects of the spectrum of the outgoing Fresnel term can significantly impact the angular bandwidth of the outgoing radiance.

3.2. Outgoing Radiance Bandwidth Computation





Fig. 57. First row: The sampling rate s_p computed from the screen-space bandwidth estimation. Second row: Pixel areas from which the sampling rate predicts an adequate approximation of the outgoing radiance variation.

Given the spatial-angular bandwidth of the outgoing radiance at a shade point, estimated as the BSSRDF bandwidth, we need to compute the associated pixel frequency bandwidth. To do so, we are motivated by Bagher et al.'s [9] bandwidth tracking approach, applying bandwidth evolution operators defined by Durand et al. [32] to the bandwidth vector $[B_s, B_\theta]^T$. Figure 55 (c – e) illustrates the transport operators in the following order:

- (1) we transform from local shade point coordinates to global coordinates by projecting the outgoing spectrum onto the shade point's tangent plane, which amounts to a shear in the spatial frequency according to the local curvature *k*,
- (2) we take the foreshortening towards the viewpoint due to $\cos \theta_o$ into account, stretching the spectrum spatially, and
- (3) we evaluate the spectrum at the sensor, after transport through free-space with a distance *d*, by applying an angular shear to the spectrum.

These operations can be compactly expressed as matrix operators, if we act directly on frequency bandwidths instead of the full spectra (Bagher et al. [9]), as:

$$\mathbf{T}_d = \begin{bmatrix} 1 & 0 \\ d & 1 \end{bmatrix}, \ \mathbf{P}_x = \begin{bmatrix} 1/\cos\theta_x & 0 \\ 0 & 1 \end{bmatrix}, \ \text{and} \ \mathbf{C}_k = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}.$$

We apply these operators, in order, to the outgoing radiance bandwidth (i.e., the BSSRDF bandwidth $[B_s, B_\theta]$), to predict the final screen-space bandwidth vector for a pixel as:

$$\begin{bmatrix} B_p & B_a \end{bmatrix}^{\mathrm{T}} = \mathbf{T}_d \, \mathbf{P}_x \, \mathbf{C}_k \, \begin{bmatrix} B_s & B_\theta \end{bmatrix}^{\mathrm{T}}.$$
(57)

Isolating the screen-space angular bandwidth B_a above,

$$B_a = B_\theta + d \left(B_s + k B_\theta \right) / \cos \theta , \qquad (58)$$

and applying the Nyquist criterion, we arrive at the pixel sampling rate s_p (in units of pixel⁻¹) as twice the angular screen-space bandwidth,

$$s_p = 2 B_a \max\left(f_x/W, f_y/H\right),\tag{59}$$

for a $W \times H$ image resolution and a horizontal and vertical field of view of f_x and f_y . Figure 57 visualizes the screen-space sampling rate for the scenes we render.

4. Hierarchical Approach

We now explain how to utilize our bandwidth estimation in order to accelerate rendering with BSSRDFs. First, we review the single hierarchy approach of Jensen and Buhler [54], then explain how we can use a dual hierarchy to adaptively cluster both illumination samples and pixels simultaneously.

4.1. Hierarchical Surface Integration

Jensen and Buhler [54] pointed out that we can cluster illumination samples over the surface in order to reduce the cost of BSSRDF evaluations. The underlying observation is that we can aggregate contributions from illumination samples that are distant from a given shading point. We can thus evaluate the BSSRDF only once for a cluster of such illumination samples, resulting in fewer BSSRDF evaluations. This approach has two passes. In the first pass, pre-integrated illumination samples are inserted into a tree data structure where each inner node *i* represents the aggregated information of its children. For example, each node stores the average illumination, the total surface area A_i , and the irradiance-weighted average location \mathbf{p}_i of its children. In the second pass, we traverse this tree until the current node accurately represents all the contributions of its children to a given shading point. If the shading point is in the bounding volume of the current node, we keep traversing the tree and consider contributions from the child nodes. Otherwise, we traverse to the child nodes only if the estimate of the solid angle subtended by the illumination samples $\Delta \omega = A_i / ||\mathbf{x}_o - \mathbf{p}_i||^2$, is larger than the user-defined quality threshold ε (Algorithm 1). While this approach significantly reduces the cost of integration over the surface, it is repeated for each shading point without considering the smoothness of resulting pixels values in screen-space.



Fig. 58. We compare our approach (red) to Jensen and Bulher [**54**] (blue) for different settings of ε . We highlight the $\varepsilon \in [0.01, 0.2]$ values and consistently reach equal-quality (measured in RMSE; *y*-axis) in less render time (in seconds; *x*-axis). The PICNIK scene challenges the assumptions of our work, and we only obtain equal-quality benefit at lower rendering times (albeit enough for visual convergence).

Algorithm 1: Single-hierarchy tree traversalInput: \mathbf{x}_o is the shading point/pixel, with I_L and I_R as children of the active node.if I is leaf or ($\Delta \omega < \varepsilon$ and $\mathbf{x}_o \notin BBox(I)$) then $| c \leftarrow$ contribution of I to \mathbf{x}_o add c to \mathbf{x}_o else $| Single(\mathbf{x}_o, I_L) Single(\mathbf{x}_o, I_R)$ end

4.2. Dual Hierarchy for Pixel-Surface Integration

We leverage a dual hierarchy to avoid traversing the illumination tree at *every* pixel. Similar to the spatial hierarchy of illumination samples in the previous approach, we also cluster pixels in the screen-space and traverse two trees simultaneously. Each node in our pixel-tree stores the average world-space position \mathbf{p}_o corresponding to the pixel group, its bounding box, the average normal direction, the average view direction, and the list of pixels covered by the node. This dual-tree

approach allows us to evaluate the contribution from a cluster of illumination samples to a cluster of pixels. Algorithm 3 is a pseudocode of our dual-tree approach.

The key difference from the single tree approach is that, at each traversal step, we have a choice of refining the pixel and/or illumination point clusters. For refining clusters of illumination samples, we use a criterion similar to the single tree approach. We always traverse down the tree if bounding volumes of pixels and illumination samples intersect. Otherwise, we decide if we want to keep traversing the tree based on the extended solid angle measure, $\Delta \omega = A_i / ||\mathbf{p}_o - \mathbf{p}_i||^2$, which uses the average position \mathbf{p}_o of clustered pixels.

Criterion to Refine Pixel Clusters. To refine pixel clusters, we use our frequency analysis to predict the potential variation in pixels. Given a pixel sampling rate $s_p[i]$ for the i^{th} pixel in a pixel tree node, an estimate of a screen-space filter extent, centered about the node, is

$$P = \rho / \max_{i} \left(s_{p}[i] \right), \tag{60}$$

where ρ is a user-defined parameter that intuitively corresponds to the fraction of captured outgoing radiance required to avoid discontinuity artifacts. The ρ setting influences pixel cluster refinement during traversal.

We refine the cluster only if our criterion predicts a high variation of outgoing radiance in the parent node's pixels (the SHADE routine in Algorithm 3). During shading (SHADE procedure) we do not adaptively refine the illumination cluster and conservatively assume that $\Delta \omega < \varepsilon$ is satisfied for all the children nodes. We could alternatively continue refining along the illumination tree for sub-nodes of the pixel tree. However, not refining results in higher performance without any noticeable visual artifacts.

a	A	lgor	ithm	2:	Sha	ıde
----------	---	------	------	----	-----	-----

Input: S and I are the root nodes of the shading point and illumination trees, with $S_{\{L|R\}}$ and $I_{\{L|R\}}$ their respective left and right children. **if** Length(S) < ρ / Bandwidth(S) **then** | Shade(S_L,I) Shade(S_R,I) **else** | $c \leftarrow$ contribution of I to S add c to all pixels **x** in S **end**

5. Implementation

We implemented our approach in the G3D Innovation Engine [72] and our results were measured on a 3.9 GHz Intel Core i3-7100 with 12 GB of RAM. Both our illumination and pixels hierarchies are kd-trees, split along the largest bounding volume dimension. Our single- and dual-tree implementations use the same underlying kd-tree structure.

Algorithm 3: Dual-hierarchy traversal

Input: <i>S</i> and <i>I</i> are the root nodes of the shading point and illumination trees, with $S_{\{L R\}}$ and $I_{\{L R\}}$ their respective left and right children.
if $\Delta \omega < \varepsilon$ and $BBox(S) \cap BBox(I) = \emptyset$ then Shade(S,I)
else if <i>S</i> is leaf and <i>I</i> is leaf then Shade(<i>S</i> , <i>I</i>)
else if S is leaf then Dual(S,I _L) Dual(S,I _R)
else if <i>I</i> is leaf then Dual(<i>S_L</i> , <i>I</i>) Dual(<i>S_R</i> , <i>I</i>)
else Dual (S_L, I_L) Dual (S_R, I_L) Dual (S_L, I_R) Dual (S_R, I_R) and

We uniformly sample points on translucent objects with Bowers et al.'s [16] blue noise approach, and image-space curvature values are interpolated from object-space values precomputed with the robust curvature estimator of Kalogerakis et al. [57]. In Section 3.1 we compute BSSRDF bandwidths as the 95th percentile of the discrete spectrum, since we find this setting balances numerical stability and accuracy. We use $\rho = 0.75$ (Equation (60)) in all our scenes and plots, as we found this value avoids discontinuity artifacts while providing good performance. We discuss the performance vs. accuracy trade-offs of ρ and ε in Section 6.

6. Results and Discussions

We have tested our approach on objects with a range of scattering parameters, as well as adapting our frequency analysis to support *several* BSSRDF models: the standard dipole (Jensen et al. [55]), the "better dipole" (d'Eon [24]), and the directional dipole (Frisvad et al. [35]). We use three scenes of increasing radiometric complexity: BUNNY, TOAD, and PICNIK (Figures 60, 59, and 53). TOAD uses the directional dipole, and the remaining scenes use the better dipole.

We compare root mean square error (RMSE) of our technique to the single hierarchy of Jensen and Buhler [54], for *total* render time, on the BUNNY and TOAD scenes (Figure 58). We sampled ε to generate the plots, and our approach consistently reaches equal quality in less time.

Comparisons in the BUNNY scene (Figure 60) illustrate our scalability with pixel coverage: the performance discrepancy between the full-view (Figure 58a) and zoom-in (Figure 58b) renderings is due to the total number of pixels present in the pixel hierarchy. As expected, the benefit of our approach increases with the number of translucent pixels: one can expect our approach to scale sub-linearly here, which is particularly favorable given recent trends towards higher resolution renderings and higher pixel supersampling rates.

We provide computational timing breakdowns when rendering an image with our technique in Table 3: specifically, we measured the Fourier precomputation, construction of the illumination tree

as well as the shading tree, and final rendering times (all on the BUNNY scene and for different values of ε). All timings are reported on a single core. Note that we use the same precomputed illumination tree across trials. Unsurprisingly, since the Fourier precomputation and the shading tree construction are independent of ε , we obtain similar timings across trials.

ε	Illumination	Fourier	Shading	Final
	Tree	Precomputation	Tree	Shading
0.2		0.10s	5.59s	6.77s
0.1	1.74s	0.11s	5.47s	11.73s
0.05		0.09s	5.12s	21.21s

Table 3. Computation times for various parts of the algorithm.

Our screen-space adaptive sampling rate for distance, local curvature, foreshortening and BSSRDF properties from first principles. Moreover, it properly explains (and it subsumes) most of the previously used heuristics in the literature, e.g., depth and normal min/max methods (Nichols and Wyman [76]). Our sampling rate formulation (Equation (58)) is simple and only requires the precomputation of two values (B_{θ} , B_s) *per material*. We do not require an additional pass to aggregate min/max statistics over the G-buffer.

We introduce a new error metric for aggregating pixel rendering cost and reducing shading cost in scenes with BSSRDFs. In doing so, we opted to follow the solid angle metric methodology of Jensen and Buhler [54] in order to avoid the cost of evaluating upper-bound metrics that rely on BSSRDF evaluation, such as in LightCuts methods (Walter et al. [106], Bus et al. [17]). We observed that such an upper-bound metric cannot scale to more complex BSSRDF shading models (i.e., the complexity of the material evaluation). We do, however, note that our frequency metric could be used as a well-found replacement of the upper-bound metrics for the specific case of BSSRDFs. Indeed, Belcour and Soler [12] have shown that frequency criterion can be used to provide an approximate relative error measure.

Limitations. The PICNIK scene (Figure 58d) is a "failure" case: specifically, our current implementation creates a separate dual tree per object in order to prevent illumination from bleeding between neighboring objects, and since the PICNIK scene includes several (smaller) translucent objects, we only obtain a benefit for a sub-region of the quality/performance range. Moreover, the solid angles $\Delta \omega$ spanned by pixel-tree nodes are more sensitive to errors for small objects and small BSSRDF scales. Since our technique approximates $\Delta \omega$ for a group of pixels, it is sensitive to these scenarios and we plan to address this issue in the future by devising more appropriate $\Delta \omega$ estimates. Overall, the fact that the additional tree construction time is amortized over fewer pixels, and the nature of our non-conservative $\Delta \omega$ estimate in the presence of smaller objects (in image-space), contribute to the suboptimal performance profile in this scene. This also explains the reduced error reduction rate for small ε .



Dual-tree (Ours)

Single-tree [54]

Fig. 59. The TOAD scene has a bumpy geometry with detailed textures. We compare the difference images of the multiple scattering term against the ground-truth for an equal rendering time (196s). The difference images are scaled by 50 for visualization. Our approach achieves more accurate estimation than the single-tree in the same rendering time.

In some difficult scenarios, high frequencies may be missed due to pixel discretization: for instance, a worst-case scenario would involve a camera facing an object with staggered depth discontinuities, which may miss small depth changes due to pixel aliasing. Here, we would group pixels that should not have been grouped.



Dual-tree (Ours)

Single-tree [54]

Fig. 60. The BUNNY scene. We compare the difference images of the multiple scattering term against the ground-truth for an equal rendering time (60s). The difference images are scaled by 200 for visualization. In this example, our approach removes artifacts under the tail and reduces Moiré patterns present in the single-tree approach.

7. Conclusion

We presented a new frequency analysis of BSSRDFs in order to predict the variation of outgoing radiance for multiple subsurface scattered light. We build and traverse a dual hierarchy over illumination samples and pixels using a well-founded refinement strategy that leverages our frequency bandwidth estimates. This yields an adaptive rendering strategy that almost consistently outperforms the state-of-the-art. Moreover, our frequency analysis and bandwidth estimates apply to a variety of existing BSSRDF models with negligible precomputation, our rendering technique scales positively with shading resolution, all without introducing any additional approximation error.

Our approach leads to several interesting open questions:

- (1) An interesting avenue would be to combine our work and the one of Arbree et al. [5]. They cluster both light source positions and illumination points at the surface of object while we cluster both illumination points and shading points. Based on the same multiple cluster idea, it should be possible to build a trial-tree that accounts for those three components during rendering;
- (2) Our frequency analysis does not account for surface global illumination transport: we ignore visibility, and the use of spatial illumination samples ignores incident radiance variation. Modeling this behavior more accurately could lead to less conservative bandwidth estimates and traversal criteria;
- (3) There are no reasons why our theory and implementation could not support other existing diffusion models (e.g., *quantized diffusion* from d'Eon and Irving [26]), and so implementing these models under our framework is interesting even if only for the sake of completeness;
- (4) Investigating how increases in ε should affect our choice of ρ , and vice versa, leads to the interesting question of whether an "optimal" parameter setting for both these values could be computed automatically;
- (5) The effects on performance and accuracy of replacing our position-based solid angle approximation with the actual projected solid angle of the underlying surface elements would also be worth investigating;
- (6) Lastly, there is much potential in analyzing our algorithm's temporal properties, notably in terms of information reuse across neighboring frames as well as in ensuring that no temporal artifacts occur due to image-space filtering.

Acknowledgements

We also graciously acknowledge funding from the Fonds de recherche sur la nature et les technologies, through their Établissement de nouveaux chercheurs universitaires program (2014-NC-173734), as well as a Master's scholarship.

Fourth Article.

Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Field Probes

by

Zander Majercik¹, Jean-Philippe Guertin², Morgan McGuire³, and Derek Nowrouzezahrai⁴

- (1) NVIDIA Research
- (2) Université de Montréal
- ⁽³⁾ McGill University, NVIDIA Research
- ⁽⁴⁾ McGill University

This article has been published in the *Journal of Computer Graphics Techniques* 2019, volume 8, issue 2, pages 1–30. Formatting has been adapted for this thesis and minor typographical issues were corrected.

I am the second author of this article. My main contributions are:

- Implementation of many components of the algorithm, in particular related to scene raytracing, probe updates and temporal accumulation and filtering.
- Elaboration of roughly half of the presented scenes, ensuring that they accurately represent the algorithm's performance and adaptability to challenging lighting situations.
- Formulation of many extensions to the algorithm which were either integrated in later prototypes or in the official NVIDIA SDK release as RTXGI [78].

Zander Majercik introduced certain key features of the algorithm, implemented them and elaborated a library to interface between NVIDIA's raytracing APIs and G3D [70]. Morgan McGuire contributed to numerous features of the algorithm as well as its implementation. Derek

Nowrouzezahrai played a supervisory role and assisted with the validation of the algorithm's accuracy and purpose.

While the algorithm lends itself to a large number of interesting extensions, the most practical and important ones which I raised were:

- Implementing a hirerachical scheme over the probe grid rather than using a purely regular grid. This allows for much larger scenes to be efficiently rendered with more modest hardware and would be one of the key improvements to the algorithm as published.
- Varying probe update patterns according to visibility and importance heuristics. The viability of specific heuristics would've been part of possible future work.

These extensions can prove useful for future implementations but were not of a substantial enough impact to warrant a separate publication as of this thesis. Instead, the release of the NVIDIA RTXGI SDK has been the primary focus for future improvements.

RÉSUMÉ. Nous montrons comment calculer l'illumination globale efficacement dans des scènes comportant des objets et de l'illumination dynamiques. Nous étendons les sondes d'irradiance classiques avec un encodage compact du champ d'irradiance complet de la scène. Premièrement, nous calculons le champ d'irradiance *dynamique* à l'aide d'un agencement mémoire GPU efficace, de lancer de rayon géométrique, et de taux d'échantillonnage appropriés sans devoir sous-échantillonner ou filter des textures sphériques prohibitivement larges. Deuxièmement, nous concevons une requête d'irradiance filtrée robuste en utilisant un nouvel interpolant basé sur les moments et prenant en compte la visibilité. Nous validons expérimentalement les compromis au niveau de la performance et de la fidélité et montrons que notre méthode d'*illumination globale diffuse dynamique* (IGDD) illumine des scènes de complexité géométrique et radiométrique variables de façon robuste (Figure 61). Pour être exhaustif, nous démontrons des résultats avec un lanceur de rayons spéculaire de pointe pour échantillonner le champ de lumière dynamique complet et nous incluons du code GLSL de référence.

Mots clés : modélisation de réflectivité, tracé de rayons, rendu

ABSTRACT. We show how to compute global illumination efficiently in scenes with dynamic objects and lighting. We extend classic irradiance probes to a compact encoding of the full irradiance field in a scene. First, we compute the *dynamic* irradiance field using an efficient GPU memory layout, geometric ray tracing, and appropriate sampling rates without down-sampling or filtering prohibitively-large spherical textures. Second, we devise a robust filtered irradiance query, using a novel visibility-aware moment-based interpolant. We experimentally validate performance and accuracy trade-offs and show that our method of *dynamic diffuse global illumination* (DDGI) robustly lights scenes of varying geometric and radiometric complexity (Figure 61). For completeness, we demonstrate results with a state of the art glossy ray tracing term for sampling the full dynamic light field and include reference GLSL code.

Keywords: reflectance modeling, ray tracing, rendering



Fig. 61. Combined with state of the art glossy ray tracing and deferred direct shading, our method (left) generates full global illumination in dynamic scenes that are visually comparable to offline path traced results (right) but several orders of magnitude faster: 6 ms/frame, versus 1 min/frame in this scene (on GeForce RTX 2080 Ti at 1920×1080). Insets isolate the direct lighting contribution and visualize the probe locations.

1. Introduction

Probe-based Global Illumination. Synthesizing images with accurate global illumination (GI) effects contributes significantly to the believability of computer generated imagery. Accurately solving physics-based GI formulations is a longstanding area of research, and doing so with offline numerical solvers remains a time consuming cost. In an interactive rendering context, a significant amount of work on generating *convincing* real-time GI effects has lead to many different solutions, each with specific tradeoffs between accuracy, flexibility and performance.

Recent work on *light field probes* strikes one such tradeoff. That representation encodes the static local light field of scene using a specialized encoding of precomputed probes placed statically in a scene (see Mara et al. [71]). The probe representation has many benefits, including the ability to perform efficient and accurate world-space (filtered) ray tracing for glossy and near-specular indirect transport, as well as supporting irradiance probe-like queries that are robust to light-leaking artifacts for smoother indirect diffuse illumination.

Given the query and sampling operations exposed by the light field probe representation, many shading algorithms can be implemented using this representation as a basis, often resulting in high fidelity images generated at high performance rates. The main limitations of standard light field probes lie in their precomputed nature and the manner in which they sample lighting in the scene. Precomputing the probe data can be costly, and therefore only treat fixed lighting and geometric conditions are handled. Moreover, the irradiance spatial interpolation and prefiltered

glossy sampling schemes can lead to aliasing and light-leaking in the diffuse and specular indirect illumination.

Real-time GI.. Unlike offline rendering, global illumination solutions for real-time applications such as video games currently rely fundamentally on lighting data that can be rapidly read from spatial-angular data structures and is usually precomputed or limited to slow updates from static geometry for dynamic lighting. Examples include lightmap representations, irradiance and radiance probes, and voxelized representations of the scene or lighting information. Each of these representations strikes a particular tradeoff between compactness, runtime flexibility, accuracy, and cost. In geometrically and/or radiometrically complex scenes, these all have well-documented undesirable artifacts that manifest as a result of undersampling and reconstruction. The most significant of these artifacts are light and shadow leaking in areas of complex visibility. Many recent GDC and SIGGRAPH talks isolate and discuss these issues. We highlight two representative ones in Figure 62.

Typically, heuristic workarounds are applied. These vary with the art and technical constraints of a particular production. In cases where only static geometry and/or lighting are treated, a



Light leaks due to undersampling in classic irradiance probes, see Hooker et al. [**46**]

Light and shadow leaks along lightmap seams (top), see Hooker et al. [46] and in voxels (bottom), see Iwanicki et al. [48]

Fig. 62. Previous interactive GI methods suffer from artifacts that often necessitate heuristic solutions, typically based on art-direction or technical constraints. Visual artifacts in these methods can manifest themselves in various forms: (in reading order) light leaking, lightmap seams, visibility/occlusion undersampling, and inter-voxel visibility mismatches.

largely manual post-processing intervention is often performed. Of course, such an approach scales poorly with scene complexity and still admits significant offline precomputation. This problem is further exaggerated in the context of dynamic environments, where the scene geometry and lighting can change at runtime, precluding manual intervention. As such, there is a great practical need for automatic caching solutions that are robust to dynamic scenes and do not sacrifice the high-performance nature of pre-cached global illumination solutions.

The core problem underlying prior techniques is not inherent in the representations, which are often efficient and well-designed for capturing either radiance (light energy along a ray used for the glossy portion of shading) or irradiance (cosine-weighted integral of radiance necessary for the diffuse portion of shading). Rather, the problem is that the techniques lack visibility information and thus cannot encode the full light field or *irradiance field* (irradiance taking occlusion into account).

This paper describes a method for extending classic irradiance probes to a representation of the full irradiance field, shows how to efficiently update that representation at runtime, and then evaluates the performance and quality of that method. The academic term for the quantity computed is the dynamic indirect irradiance field; we call the new probe technique *dynamic diffuse global illumination* (DDGI) in keeping with game industry jargon.

The specific contributions of this work are:

- Extension of irradiance probes with accurate and dynamic occlusion information by an incremental scheme that leverages a compact, GPU-tailored data layout and compute schedule for converged "infinite" bounce diffuse global illumination;
- An algorithm for ray tracing irradiance probes independently of the primary visibility resolution and frame rate, avoiding the cost of denoising or prefiltering prohibitively high resolution spherical textures;
- A spatial interpolation, occlusion, and filtering scheme more robust to irradiance queries in scenes with temporally-varying geometry and lighting;
- Evaluation of a system for producing results nearly identical to (offline) path tracing in many cases, combining dynamically-updated occlusion-aware irradiance with GPU ray-traced glossy and specular reflections, reducing aliasing artifacts in these indirect contributions;
- Open source reference shaders for implementing DDGI, taken directly from and compatible with the open source G3D Innovation Engine [70].

2. Related Work

Works on interactive global illumination span several decades. We review the areas most relevant to our work.

Image-based Lighting. Image-based lighting methods form the basis of most interactive precaching lighting solutions in modern video games (Martin and Einarsson [64], Ritschel et al. [89], McAuley [66], Hooker [46]). Here, a common workflow involves placing light probes (of various types) densely inside the volume of a scene, each of which encodes some form of a spherical (ir)radiance map. Prefiltered versions of these maps can also be stored to accelerate diffuse and glossy runtime shading queries.

One interesting variant of traditional light probes allows digital artists to manually place box or sphere proxies in a scene, and these proxies are used to warp probe queries at runtime in a manner that better approximates spatially-localized reflection variations (see Lagarde and Zanuttini [60]). Similarly, manually-placed convex proxy geometry sets are also used to bound blending weights when querying and interpolating between many light probes at runtime, in order to reduce light leaking artifacts — one of the predominant artifacts of such probe-based methods.

These probe- and image-based lighting techniques are ubiquitous in modern offline and real-time rendering, and we refer interested readers to a comprehensive survey (see Reinhard et al. **[86**]).

While production-quality image-based lighting systems generate convincing illumination effects, practitioners agree that eliminating manual probe and proxy placement remains an important open problem in production (see Hooker et al. [46]). Currently, without manual adjustment, it is impossible to automatically avoid probe placements that lead to light and dark (i.e., shadow) leaks or displaced reflection artifacts. To avoid these issues, some engines rely instead on screen-space ray tracing (see Valient [102]) for pixel-accurate reflections. These methods, however, fail when a reflected object is not visible from the camera's point of view, leading to inconsistent lighting and view-dependent (and so temporally unstable) reflection effects.

Light Field probes (see Mara et al. [71]) automatically resolve these issues in scenes with static geometry and lighting by encoding additional information about the scene geometry into spherical probes. A solution for dynamic lighting is presented in Silvennoinen et al. [91], but this solution only supports coarse dynamic occluders and requires complex probe placement based on static geometry. We inherit the advantages of the representation in Mara et al. [71], which we extend fundamentally to treat *dynamic* geometry and lighting variations at runtime (Section 5). No manual placement is necessary and a naïve uniform grid probe placement results in artifact-free renderings. Reflections appear (consistently) where they should due, in part, to an accurate world-space ray tracing algorithm (Section 5.) without the need for manually placed proxy geometry. As such, light field probes can be leveraged in both the context of traditional (prefiltered) radiance lookups, as well as shader-enabled world-space ray tracing.

Interactive Ray Tracing and Shading. Many recent interactive rendering approaches treat the problem of resolving point-to-point visibility queries, shaping modern solutions applied in practice today. Ritschel et al.'s **[88]** imperfect shadow maps encode a sparse, low-resolution representation of point-to-point visibility in a scene, which they use to compute accurate secondary diffuse and glossy reflections using virtual point lights (generated, e.g., with a ray-tracer). Our work is motivated by

another such solution: voxel cone tracing (see Crassin et al. [23]). At a high-level, one can interpret our ray tracing technique (Section 4.2) as tracing rays against spherical voxelized representation of the scene (i.e., as opposed to the octree representation constructed for traditional voxel cone tracing). Two important differences that contribute to many of the practical advantages of our representation are: first, that we *explicitly* encode geometric scene information (i.e. radial depth and depth squared) instead of relying on the *implicit* octree structure to resolve local and global visibility details; and, second, that neither our spatial parameterization nor our filtering relies on scene geometry. This allows us to completely sidestep the light (and dark) leaking artifacts present in traditional voxel cone tracing. Finally, we are able to resolve *centimeter-scale* geometry at about the same cost (in space and time) as a voxel cone tracer that operates at *meter-scale*.

Representation. We use Cigolle et al.'s [18] octahedral mapping from the sphere to the unit square to store and query our spherical distributions. This parameterization has slightly less distortion and provides simpler border management than, for example, cube maps. Since we target *true world-space* ray tracing in a pixel shader, and not just screen-space ray tracing, our technique can be seen as a generalization of many previous real-time environment map Monte Carlo integration methods (Stachowiak and Uludag [96], Wyman [111], Toth et al. [101], Jendersie et al. [53]).

We are also motivated by the preliminary investigations of Evangelakos [34] and Donow [28] that validate the accuracy of single-probe ray tracing and the feasibility of multi-probe traversal. Specifically, a single probe can perfectly sample the geometry of a region with *star-shape topology*, and so ray tracing with a single probe in these regions will incur no visibility error (outside of errors due to probe directional discretization).

3. Dynamic Diffuse Global Illumination Probes: Overview

As in Mara et al. [71], we encode geometric and radiometric scene data into spherical distributions at discrete probe locations. We combine efficient GPU ray tracing to enable accurate shader-based world-space ray tracing (using either a probe-based marching approach, or native GPU ray tracing APIs), with filtered irradiance queries to compute diffuse, glossy and specular indirect illumination at real-time rates.

Specifically, we encode the spherical diffuse irradiance (in GL_R11G11B10F format at 8×8 octahedral resolution), spherical distance and squared distances to the nearest geometry (both in GL_RG16F format at 16×16 octahedral resolution). We pack each of these square probe textures into a single 2D texture atlas with duplicated gutter regions to allow bilinear interpolation without any boundary artifacts (see Figure 63).

Instead of precomputing the probe data once at scene initialization, we dynamically update the probes to capture variations due to dynamic geometry and lighting. This allows us to enable truly dynamic high-fidelity global illumination. Our method retains the high-performance of Mara et al. **[71]** and Figure 64 illustrates our ability to compute fully converged multi-bounce global illumination.

At each frame we are able to efficiently blend updated ray-traced illumination into our probe atlas in addition to interpolating probe depth information to adapt to changes in scene geometry. In a forward or deferred rendering pass, probes can effectively be treated as indirect lighting buffers analogous to standard precomputed environment maps.

We detail our method for updating dynamic diffuse global illumination probe distributions in Section 4 before discussing how to use probes at runtime to efficiently compute dynamic global illuminations in Section 5.



Fig. 63. Spherical irradiance and depth textures. We encode spherical data in an octahedral parameterization, packing all the probes in an atlas. One-pixel texture gutter/border to ensure correct bilinear interpolation, and additional padding aligns probes on 4×4 write boundaries.



Fig. 64. Left to right: direct illumination only, direct illumination with *one bounce* of indirect diffuse illumination (computed with spherical irradiance updated by our dynamic filtered ray casting approach), and the fully converged *multi-bounce* global illumination that iteratively incorporates bounced lighting computed across probes.

4. Updating Dynamic Diffuse Global Illumination Probes

We place our probes as in Mara et al. [71] before incrementally updating the probe content (Section 4.1). At every frame, we follow a multi-step process to update probe information in order to incorporate the effects of dynamic geometry and lighting:

- (1) generate and trace *n* primary rays from each of the *m* active probes in a scene, storing geometry for (up to) $n \times m$ surface hits in a G-buffer like structure of *surfels* with explicit position and normals (Section 4.2);
- (2) **shade** the surfel buffer with direct and indirect illumination (Section 4.3), with the same routine used to shade final image pixels, i.e., those directly visible from the camera (Section 5); and
- (3) **update** the texels in the octahedral representations of the *m* active probes by blending in the updated shading, distance, and square-distance results for each of the *n* intersected surfels (Section 4.4).

We discuss several methods to select *active probes* to update in Section 4.2, however we employ a conservative selection approach and set *all* the probes in a scene to active. As such, our rendering performance metrics are a conservative upper bound on the expected performance of our algorithm.

Shading the probe-intersected surfels (see Section 4.3) relies on lighting and probe data from the previous frame, which serves two purposes: first, this allows us to amortize the cost of computing multiple indirect bounces over several frames; second, when combined with our blending approach (see Section 4.4), this enables a smooth transition between sharp geometric and radiometric discontinuities (over time). A negative side effect, of course, is that *indirect illumination* can sometimes appear to "flow" in and out of areas with dramatic visibility changes, due to the latency in the *indirect illumination* update. Given the relative smoothness of indirect illumination, compared to direct illumination (which we update precisely at every frame), we follow the guidelines and observations of prior work that indicate that these specific artifacts remain an acceptable perceptual tradeoff for viewers (Crassin et al. [22]).

4.1. Probe Grid Placement

We place probes in the volume of the scene at the vertices of an axis-uniform 3D grid. We use a power-of-two resolution per axis, simplifying probe indexing to simple bitwise operations. We can scale per-axis grid cell spacing independently for scenes that require different spatial discretization per axis.

We note that visibility-aware probe selection and sampling affords us a certain latitude when placing probes: probes that fall inside walls or other geometry will be ignored by visibility query metrics (McGuire et al. [71]). Also, other than simplifying probe indexing, no aspect of the probe generation or shading *requires* a uniform grid placement; indeed, probes can be placed according to schemes used in other probe-based algorithms, including tetrahedral grids.

Every point in space is associated with a *cage* of vertices corresponding to the eight vertices of the grid cell that contains the point. We recommend using a grid resolution and scale that results in at least one full cage of vertices in each room-like space. This is needed to ensure a sufficient sampling of local illumination variation inside each separated/distinct space in a scene. For human-scale scenes, we found a spacing of one to two meters sufficient, however we illustrate results with a variety of grid spacings.

4.2. Generating and Tracing Probe Update Rays

We update texels in probes to account for dynamic geometry and lighting variation, blending in their results over time in order to smoothly account for the effects of these dynamic changes on the final rendered result.

At each of the *m* active probes, we uniformly sample *n* spherical directions according to a stochastically-rotated Fibonacci spiral pattern, similar to Mara et al. [71]. We then spawn *n* rays with these directions and a (shared) origin of the probe center. We lay out the rays across the *m* probes in a thread-coherent fashion, casting all of them in one batch.

Technical Notes. While Vulkan Ray Tracing and DirectX Ray Tracing APIs permit ray dispatching from primary shaders, we benchmarked our ray batching and observed that it minimizes register pressure and facilitates debugging through inspection of intermediate shading results.

We experimented with several probe and ray sub-sampling schemes: e.g., updating only a subset of the probes in a scene, such as those within a certain radius of the camera; or varying the ray count based on distance to the camera. While these adaptive schemes led to some expected performance improvements, they introduced many additional scene-dependent user parameters and additional bookkeeping.

We instead opted for simplicity in our final results: our reference implementation updates every probe at every frame (i.e., sets every scene probe as *active* during probe updates) and dispatches the same number of rays per probe. More complex usage scenarios, such as large open world environments, could benefit from a probe streaming scheme as well as probe LODs operating at several grid scales (see Section 7.1 for more discussion).

4.3. Secondary Surfel Shading

We employ a unified shading model for both probe updates and final rendering. Specifically, we compute global illumination in two contexts at runtime: first, when updating the shading on the $m \times n$ probe-sampled surfels, and, finally, when shading pixels from the camera for the final output image. Both of these contexts use the same shading routine, composed of a direct illumination pass that uses state-of-the-art interactive practices and an indirect lighting pass that leverages our probe data. We discuss the details of the shading routines in Section 5, focusing on the subtle differences in its application during probe surfel updates, below.

To abstract over the differences in how shading queries are made during probe update and final rendering, our shading routines expect a shading position, normal, and viewing direction as input (Section 5). For probe-traced surfel shading updates, we pass the intersected surfel locations and normals, as well as the direction from the surfel to the probe center, as input to the shading routine.

4.4. Probe Surfel Updates

After surfel shading, each of the $m \times n$ surfel points will have an updated shading value, and the sampled surfel distances (and squared distance) are also updated relative to their associated probe centers.

We update the probe texels (associated to each of surfel) by alpha blending in the new shading results at a rate of $1 - \alpha$ as follows,

$$E_{t}(\tau) = \sum_{r \in \Omega} \max\left\{0, \omega_{\tau} \cdot \omega_{r}\right\} \cdot L_{r}$$

$$\text{Irradiance}_{(t)}[\omega_{\tau}] = \operatorname{lerp}\left\{\operatorname{Irradiance}_{(t-1)}[\omega_{\tau}], E_{t}(\tau), \alpha\right\}$$
(61)

where α is a *hysteresis* parameter that controls the rate at which updated shading overrides shading results from previous frames, $E_t(\tau)$ is the irradiance at frame t and texel τ for probe Ω , ω_{τ} and ω_r are the texel and ray directions, respectively, L_r is the incoming radiance of ray r, and Irradiance_t[ω] stores the final probe irradiance at time t for a direction ω . We set α between 0.85 – 0.98 for all our results.

We directly compute the filtered irradiance using a moment-based filtered shadow query, allowing us to avoid brute-force prefiltering of a (higher resolution) incident radiance map. This smooth incident irradiance field will be used to compute diffuse indirect illumination (Section 5.2), and we can optionally maintain a higher-frequency shading map for glossy and specular indirect shading.

Technical Notes. We purposefully lay out our data and update computation in order to promote coherence in execution: probe texels operate in (near) lockstep to their neighbors, often operating

on the same ray, blending in its result. This yields not only coherent memory fetches on the GPU, but also coherent compute. We update irradiance and depth texels against a cosine lobe distribution, a necessary step for correct irradiance representation (see Akenine-Möller et al. [1]). In the case of the depth and depth-squared buffers (as with Mara et al. [71]), we employ an additional *depth sharpening*, warping them according to a cosine-*power* lobe distribution. We do not update texels weighted below a threshold (we used 0.001) in the cosine-power lobe distribution.

Note that, while the updated spherical irradiance distributions will be used to shade *view-independent* diffuse reflectance effects, they are updated to correctly account for any glossy/mirror *view-dependent* shading due to dynamic geometry and lighting in the environment. For a indepth discussion of irradiance and computing irradiance using light probes, we refer readers to Akenine-Möller et al. [1] (pg.268, 490).

5. Shading with Dynamic Diffuse Global Illumination Probes

We compute multi-bounce global illumination effects with diffuse, glossy and specular transport. We will separately discuss the shading procedure for each of these transport components. We motivate and outline the novel contributions of our technique below. Instead of explicitly detailing every algorithmic detail and/or parameter setting of our implementation, we provide a full source reference implementation as a supplemental reference for the exact technical details.

5.1. Direct Illumination

We compute direct illumination from point and directional light sources using a deferred renderer with variance shadow mapping (see Thaler [98], Donnelly and Lauritzen [27]).

We can also handle direct illumination from extended area light sources using our *indirect* illumination pipeline: all one-bounce indirect lighting contributions (Section 5.2 compute one bounce of lighting *seeded* by the direct illumination in a scene. Multiple bounces of indirect illumination are instead seeded by the *previous bounce* of indirect lighting in the scene (Section 5.3). With this in mind, we can compute *direct illumination* from *area lighting* by seeding our indirect illumination shading routine with the area lighting emission profile in the scene (i.e., instead of the direct illumination profile).

The last row of Figure 66 gives a sense of this area lighting setup: apart from the "direct light" on the window geometry, the remainder of the shading in the bathroom scene is computed as an "indirect" contribution from the window light.

With this approach, we can avoid approximating direct illumination from area lights, instead relying on the robustness of our probe-based shading technique to compute smooth area shadows and reflections. We detail this method, below.

5.2. Diffuse Indirect Illumination

We compute spherical incident irradiance distributions at each probe, and we extend the original visibility-aware probe weighting scheme of [71] to query diffuse irradiance from probes at shading points in the scene. We modulate incident irradiance by spatially-varying diffuse surface albedo in order to compute one bounce of indirect outgoing diffuse radiance (see Akenine-Möller et al. [1]).

To compensate for the fact that the incident diffuse irradiance at a probe location does not account for local occlusion around a shading point, we (optionally) modulate outgoing diffuse reflection by a screen-space ambient occlusion variant (see Shanmugam and Arikan [90]). Note, however, that we do not include this local occlusion when computing surfel shading updates (Section 4.3): the impact of omitting this term on *secondary* lighting (i.e., computed as the diffuse, glossy or specular reflection of the surfel shading) is significantly less than on the lighting of directly-visible surfaces.



Fig. 66. Shading of a surfel X. We sample each probe in the 8 probe cage using the surface normal n in world space. We backface weight each probe P using dir, the direction from X to P. The mean distance stored for P is represented by r. To avoid sampling visibility near the visibility function boundary (i.e. the surface), we offset from the world space position at X based on the surface normal and the camera view vector.

Our indirect diffuse interpolation and sampling technique differs from that of Mara et al. [71], incorporating ideas from the ray-tracing and shadow mapping literature that are designed to increase robustness to dynamic geometry and lighting. Specifically, after computing the indices of the eight-probe cage that contains the shading point, we compute interpolation weights for each irradiance probe from its position and direction (relative to the shading point), as follows (see Figure 66):

• we backface-cull probes that lie below the shading point's tangent plane, using a soft threshold that falls off smoothly as the dot product of the shading normal with the direction towards a probe approaches zero,

- we apply a perceptually-based weighing to account for the human visual system's sensitivity to (relatively) low-intensity lighting in otherwise dark regions (i.e., light leaks): we reduce the contribution of very low irradiance values (i.e., less than 5% of the representable intensity range) according to a monotonically-decreasing curve profile,
- we apply mean- and variance-biased Chebyshev interpolants, as detailed in the variance shadow mapping method, to our visibility queries (see Figure 68) in order to appropriately filter radiance queries,
- we offset the shading point according to bias proportional to the shading normal and the direction to the probes: this improves the robustness of the visibility-based interpolation weights by moving away from potential shadowed-unshadowed discontinuities, and
- we then perform a standard trilinear interpolation based on the distance between the shading point and the probe centers, using the aforementioned weighting and biasing factors.

Each of these weighting terms are appropriately bound using conservative epsilon tests in order to avoid numerical issues when normalizing the weights, e.g., when per-probe weights approach zero. Figure 68 illustrates the visual impact that each of these weighting stages has on the final rendering, highlighting how each factor contributes to eliminating artifacts, starting from a traditional irradiance-probe rendering and progressing through each of our factors above.

Note that shading with standard irradiance probes results in significant light-leaking artifacts, as expected and similar to those highlighted in related work (see Figure 62), whereas our final renderings agree much more closely with path-traced ground-truth. Any color banding artifacts in our results are due to conversion from HDR to LDR into the PDF-embedded PNG format; these artifacts are not present on display.

Render Pass	Time (ms)	
Ray generation	0.1	
Ray cast	0.8	
Ray shade	0.4	
Probe update	0.7 (0.3 color + 0.4 depth)	
Sample irradiance probes for primary ray shading	0.5	
Deferred direct shading	0.1	
Total	2.6	
Add brute-force ray-traced glossy	+2.4	

Fig. 67. Timings for the indirect light components of a single frame render using $32 \times 8 \times 32$ probes with 64 rays/probe. Probes were at 8×8 resolution using RGB10A2 format for color and RG16F format for depth. Timings were taken using glTimerQuery. We allocate time for the combined ray cast according to the proportion of rays for diffuse and glossy. We did not profile the unoptimized modular passes for parts of the system outside our contributions (shadow maps, AO, G-buffer generation) though we include unoptimized glossy ray cast and unoptimized glossy indirect shade in the final row for context.

Technical Note. Previous work use $128 \times 128 \times 6$ high-precision cube maps to store depth information, however our additional weighting criterion allow us to scale down to 16×16 medium-precision depth values without incurring any numerical issues.



Fig. 68. Irradiance interpolation and sampling in a closed room scene (**a**), where light enters only from a single door opening (**b**). Images (**b**) through (**i**) place the camera at the back corner of the room, illustrating how light leaking artifacts from traditional irradiance probes (**c**) are progressively compensated for using the terms in our novel interpolant (**d**) – (**f**). We visualize a $2 \times$ error image (**h**) between our final result (**f**) and the path traced reference (**g**).

5.3. Multiple Bounces of Global Illumination

We compute multiple bounces of indirect illumination recursively, across frames, by seeding the radiance buffers with the previous bounce of light, similar to McGuire et al. [71]. This leads to a time-lag artifact for indirect bounces that is most evident in static scenes viewed by a static camera, which is not the use case in which we are primarily interested: when the view, lighting, and/or scene geometry is dynamic, the lag in indirect bounces is not noticeable.

Our approach could easily be adapted to collect the per-bounce results (up to a maximum bounce bias) before display, if fixed view and geometry usage scenarios are a priority. Given the

performance of our approach (see Section 6), we would still reach interactive shading rates despite not being able to amortize the cost of multiple bounces across frames.



(a) Classic irradiance probes

(b) Low-res raycast visibility

(c) New variance visibility

Fig. 69. Comparison of probes with no visibility test, visibility test by a low-res ray cast, and our new variance visibility.



Fig. 70. Indirect shadows with increasing probe grid resolution. As the resolution of the probe grid increases, the indirect shadow approaches the pathtraced reference without the overdarkened look of SSAO.
6. Results

We benchmark our approach on scenes with a mix of geometric and radiometric complexity. We explore the impact that probe count, resolution, and pixel format play in final rendering quality. We also compare the quality of our final rendering to path-traced references (computed offline).

We show results with direct illumination, glossy ray-traced reflection, shadows, tone-mapping, bloom, and other standard rendering terms to show the diffuse GI in the context of a full renderer. The code for this pipeline is available as the open source G3D Innovation Engine (https://casual-effects.com/g3d), where we injected the diffuse and glossy GI terms using the G3D::DefaultRenderer's path for reading from two screen-resolution textures. The glossy ray tracing is simply brute force mirror-ray tracing per pixel followed by a bilateral blur pass based on glossiness and distance, which is a standard practice (Valient [102]). For a state of the art ray traced glossy approach, see Schmid et al. [51].

Since the rays will be convolved with a clamped cosine, glossy reflections on second-hit surfaces will be indistinguishable in most cases. We recommend adding the energy from the glossy portion of the BRDF to the matte portion in those second-hit surfaces, which we did in all of our result figures. This handles today's practical cases well and gives a slight speedup to shading. However, in the theoretical case of extremely dense and high resolution probes, this step would affect correctness, so we left a disabled code path in the reference implementation that performs the full BRDF shade including glossy for the surfaces seen by probes.

For consistency when reporting performance statistics, we conservatively update every probe every frame, with the understanding that many probes that may not impact the final rendering quality will still be updated (and, so, will incur a performance cost). This is especially true in large scenes (e.g., Figures 66, 69 and 77); here, many probes either fall outside the camera frustum and/or do not contribute to any directly (or indirectly) visible scene geometry. We leave optimized probe selection and adaptive probe updates to future work.

		Number of probes		
		$16 \times 8 \times 16$	$32 \times 8 \times 32$	$32 \times 16 \times 32$
s/probe	32	1.63 GRays/s	1.66 GRays/s	1.6 GRays/s
	64	1.63 GRays/s	1.62 GRays/s	1.59 GRays/s
	128	1.65 GRays/s	1.6 GRays/s	1.5 GRays/s
Ray	256	1.65 GRays/s	1.5 GRays/s	1.48 GRays/s

Fig. 71. Ray throughput in Gigarays per second. Timings were taken on our Greek Temple Scene (876127 primitives) using an NVIDIA RTX 2080 Ti. For reference, we used $32 \times 8 \times 32$ probes with 64 rays/probe for our largest scenes.



(b) Path-traced Reference

(c) Direct Only

Fig. 72. Quality comparison across a selection of probe resolutions and densities. Probe resolution is given as $X \times Y \times Z$, with *Y* increasing towards the camera. Note that even at low resolution, an image rendered with a sufficient number of probes looks identical to the path-traced reference.

6.1. Probe Count and Resolution

Figures 72 and 73 illustrate the impact that probe count and resolution have on the final rendering.





(b) Second view, indirect and direct. 8x8x8 probe grid at 64x64 resolution.

Fig. 73. Quality comparison across a selection of probe resolutions and densities for a more complex scene.

Figure 70 illustrates the impact of probe count in the specific case of indirect shadows. Probes are initialized in a simple uniform 3D grid, scaled to the bounding volume of the scene. There is no need for manual probe placement due to the visibility-aware sampling of probe data, one of the main advantages of the probe representation.

We can conclude that probe count plays a larger role than probe resolution: rendering with low (angular) resolution probes still leads to results that converge favorably compared to path traced

reference; however, low probe count/density can lead to subtle light leaking artifacts. These are resolved with modestly chosen probe density settings.



(a) Path-Traced Reference

(b) New Irradiance Field

Fig. 74. Comparing path tracing (left) to our dynamic diffuse global illumination probes (right), with full diffuse global illumination, in a box with a dynamically translating dragon. See our video supplement for a real-time animation.

6.2. Ray Tracing Performance

Figure 71 shows throughput for ray casts with varying probe densities and rays/probe. Except in extreme cases, throughput is above 1.5 GRays/second. Figure 67 shows timings for our algorithmic contributions within our rendering pipeline, and gives some context to the times in Figure 71. The total time of our contributions is 7ms. However, note that our implementation updates all probes every frame, and thus incurs a high probe update cost. Adaptive probe selection (see Section 7.1) would reduce this time considerably.

6.3. Probe Texel Format

Figure 75 illustrates the impact of probe texel formats on the quality of our final rendering. Using 8-bit integer pixel formats can lead to artifacts that vanish at 16-bit floating point representations. Experimentally, we find that 11-bit floating point representations strike a good balance between precision and storage: at this bit depth, we maintain the visual fidelity of the 16-bit floating point representation while reducing storage by a factor of 45%.

6.4. Quality Comparisons

Figure 74 compares our method to path traced reference on a dynamic scene using a grid of $4 \times 2 \times 4$ probes at 8×8 color resolution and 16×16 depth resolution. Our results are almost indistinguishable from the path-traced reference, rendered at several orders of magnitude faster. Note the variations of subtle diffuse indirect illumination caused by the reflection of the red dragon onto the white walls of the box as the dragon passes under the light.

Figure 68 also illustrates the impact that the individual components of our robust diffuse indirect weighting scheme have on rendering.

7. Conclusion and Discussion

We present an approach for updating and interpolating the irradiance field, as represented in dynamic diffuse global illumination probes, in the presence of dynamic scene geometry and lighting, robustly treating temporal occlusion and lighting variation. Our method does not suffer from light or shadow leaking artifacts, suppressing aliasing due to undersampling. We compute accurate diffuse, glossy and specular global illumination effects in arbitrarily dynamic scenes at high performance.

This is due, in part, to an efficient data packed probe layout that enables ray and shading computation to be dispatched in a coherent manner across probes. Our occlusion-aware spatial irradiance interpolation scheme is more robust to variations in smooth diffuse illumination, compared to the original scheme presented by Mara et al. [71].

We demonstrate how traditional forward and deferred rendering architectures that leverage precomputed lighting can be combined with modern GPU-enabled ray tracing in order to combine the advantages of both these enabling technologies.

Indeed, we build atop the idea that an efficient ray tracer should be used not as a *substitute* for rasterization, but rather as a means to *complement* it when incoherent visibility queries are needed. When combined with design strategies commonly used in the interactive rendering community, such as temporal amortization and probe-based precomputed lighting, our hybrid rendering solution generates results that amount to more than the sum of its technological parts.



Fig. 75. Color precision at 128 rays/probe/frame. GL_RGB5A1 requires a low hysteresis $\alpha = 0.8$ in order to not fall below the blending threshold with many rays and suffers from flicker and oversaturation. The other formats, using $\alpha = 0.95$, are nearly indistinguishable from one another although GL_R11G11B10F has a greenish tint because it cannot represent exact grays. GL_RGB10A2 balances quality and size. Note that GL_RGB8 gives less precision but requires the same 32 bits/texel storage on modern GPUs due to word alignment. GL_RGB10A2 and GL_RGB8 are too dark because they lack the dynamic range of floating point.



Fig. 76. Time-lapse images showing fully dynamic GI with moving geometry. In this example, not only are a large number of spheres animating and casting complex GI, but they are also moving *through* the probes, which would lead to objectionable shadow leaks without correct occlusion. See our supplemental video for a real-time animation.



(b) Evening

Fig. 77. Time-lapse showing different times of day simulated with dynamic lighting. See our supplemental video for a real-time animation.



Fig. 78. Our pillars test scene initialized with an enclosing 8x4x8 probe grid. The grid is perfectly aligned with the box in the top left image. All other images have a rotation and translation of the grid. Some images are chosen to intentionally break the algorithm by leavin part of the scene uncovered by probes (bottom left). Others are chosen at random. As long as there are probes covering the area being shaded, our algorithm is robust to rotation and translation of the probe grid. See our supplemental video for a demonstration of multiple random rotations, including some failure cases for positions outside the rotated and translated probe grid.

Our indirect diffuse shading relies on a fundamental assumption about the spatial and angular relationship of radiance in a scene: here, we assume that the incident light at a shade point is similar to the incident light at the probes that surround it, *if the probes and the point are mutually visible*. The error induced by this assumption increases as probe density decreases (Figure 72).

7.1. Future Work

There are two immediate areas of future work that merit further investigation: the inclusion of more fall-back/alternative rendering paths, and adaptive selection for probe updates.

Alternative Glossy and Specular Render Paths. All of our glossy and specular transport is computed by sampling rays using our dynamically-updated probes. This brute force sampling solution errs on the side of accuracy, at the cost of performance. Compared to public demonstrations of the recent interactive ray tracing advances, such as the PICA PICA (Andersson and Barré-Brisebois [3]) and Battlefield V engines, our glossy and specular shading solution is rather simple. These works give a sense of the potential performance gains that are possible by heuristically shortening rays, falling back to environment mapped reflections, combining true ray-traced reflections with screenspace reflection approximations, using lower-resolution geometry LODs for distant intersections, and simplifying reflection shaders after the first bounce direct illumination.

These approximations are powerful — incorporating them in a manner that is both robust to different scene geometries and materials, and that allows direct control over error bounds, are interesting directions of future work.

Adaptive Probe Selection. In large and complex scenes, even conservatively culling probes can result in significant performance improvements. The scene depth information we currently sample and store at each probe is immediately useful (and, likely, sufficient) to inform a more efficient probe scheduling routine. For example, one can readily cull probes from the *active probe list* using a *furthest surface* heuristic (i.e., from every ray traced at every probe).

Acknowledgements

Thanks to Dylan Lacewell, Keith Morley, and James Bigler for invaluable advice on working with OptiX. Thanks to Michael Mara for the initial light field probe code. Our implementation is built on the G3D Innovation Engine and benefits from the work of Michael Mara, Corey Taylor, and the many other open source contributors to that platform.



Fig. 65. Direct illumination (left) versus full (diffuse, glossy & specular) global illumination (right) computed using dynamically-updated irradiance probes and ray-traced reflections.

Chapter 2

Conclusion and Future Work

This thesis presented four high-impact articles focused on improving the accuracy and performance of evaluating realistic camera models. We have shown that, by taking these models into account at various steps in the pipeline, we can create more realistic and more appealing results in less time than it would to consider the camera model separately.

In Part 1, we leveraged traditional distribution effects in a novel way to construct an extremely efficient non-linear motion blur algorithm which provides never-before-seen fidelity all while still being real-time. We believe that this algorithm still shows excellent promise and could be used successfully to augment motion blur rendering, especially in situations such as virtual reality, where simulating distribution effects has goals beyond visual fidelity and allows the human eye and brain to better interpret virtual worlds.

In Part 2, we extended typical appearance filtering approaches to consider area lights such as environment maps, greatly broadening the utility of such algorithms. We manage to perform substantially better than the state of the art while remaining scalable and flexible, creating solid ground work for future forays into the topic.

In Part 3, we devised a smart hierarchy on both surface and camera samples to quickly generate complex subsurface scattering effects using otherwise traditional and accurate algorithms. This approach, which effectively takes advantage of extra dimensions by more directly considering pixel locations holistically, allows us to very precisely scale down the amount of work we have to perform according to the requirements of the surface, lighting and geometry.

In Part 4, we created a novel algorithm for real-time global illumination which manages to compactly store irradiance information in a view-independent, stable and scalable way. This framework has since seen further refinements and was adopted as an official Nvidia RTX software development kit under the name RTXGI [78], making it accessible to many game developers and streamlining implementation and performance.

The presented articles each contain numerous avenues for future work, but we would like to highlight some specific ways forward that may be of particular interest or which have already begun.

2.1. Real-time Appearance Filtering

Our first topic of interest was already alluded to in Part 2: we suspect that the algorithm we have devised has potential for real-time applications. Currently, various parts of the implementation are not very amenable to GPU rendering, especially as far as memory usage is concerned, but we believe that these are resolvable limitations.

We have spent a significant amount of time working on implementing this algorithm in real-time already, but unfortunately have been unable to produce satisfactory results that would have been worth including in the final publication. Instead, we decided that these findings would deserve their own article at a later date.

There are many interesting aspects to this problem that we would like to cover, most notably:

- The current implementation leverages a dense SAT histogram which takes up significant memory space and requires multiplying and adding large matrices. We believe that, for a real-time implementation, a less accurate but much faster representation, maybe using a hybrid approach depending on the pixel footprint, would be viable.
- The rotated zonal harmonic Gaussian roughness lobes are inefficient to process on a GPU since they are much larger than the accelerated formats such as 4-component vectors and matrices. We think that a similarly approximate method as for the prior point would be necessary, combined with reconfiguring the mathematical structure to favor the hardware.
- Some computations, both at runtime and as part of the algorithm's precomputations, can be trivially accelerated with a GPU, which should provide immediate gains for little work.

The state of the art for online appearance filtering is currently far behind that of offline rendering, and therefore we believe that there is still a lot of potential for even fairly large approximations of our algorithm to outperform, both visually and in wall clock terms, current approaches.

2.2. Surface-distributed Ray Tracing

Another interesting idea is attempting to use what we have learned with Part 2, but applying it directly to distribution effects rather than area light sources. In this scenario, we reuse and refine certain parts of our algorithm, such as our SAT histogram, such that we can handle pixel footprints which vary in time and/or lens space.

This project is currently under way as part of a collaboration with Ubisoft La Forge. The algorithm generates a histogram of normals over a surface and stores it in a SAT for efficient lookup. When tracing any ray, we also compute the full cone of projection coming from the lens to the surface that the ray has hit, and we then sweep this cone alongside the direction of motion, if any.

This allows us to then project one or more shapes on the intersected surface, giving us a very accurate representation of the footprint. This footprint can then be used to compute the BRDF for all time and lens locations at once, rather than requiring hundreds or even thousands of rays to be computed to approximate this through Monte Carlo.

In addition, since the footprints generated in such a fashion can be complex shapes, we generate multiple rotated versions of the original normal map and compute a SAT for each, allowing us to choose one of many orientations to best approximate the shape of the footprint. If necessary, subdivision of the footprint is done.

Finally, given the histogram recovered for this footprint, we can compute the NDF to a high degree of precision and thus generate a faithful response. Our algorithm presently makes the assumption that local flatness holds in spite of very large footprints, which is something we are looking to confirm or deny. We also decouple geometric discontinuities from surface filtering: to take into account overlapping geometry, we perform a pre-processing step per frame which sends highly simplified rays which only determine which surface was hit, allowing us to construct a "weighting" buffer which can be used to combine every surface we have filtered. We can also reuse some of those rays for appearance filtering, reducing costs.

We believe that this approach should be fast enough to be used in interactive rendering, with potential for approximations bringing it into true real-time territory.

2.3. Appearance Filtering in Time, Lens and Light

As a final keystone project, we want to devise a new framework which combines and optimizes Part 2 with Section 2.2. Thanks to the numerous (and intentional) similitudes between the two algorithms, we believe that we can very efficiently merge the computations and reuse many intermediate results to minimize overhead.

The main challenges of this framework would be twofold. First, we need to devise a scalable means of continuing the swept conic projection we use in Section 2.2 to at least one surface bounce, which would allow us to extend our time and lens volume to direct light sampling. This presents numerous challenges for both glossy and matte surfaces. Second, we need to unify our computations such that they can all be shared by all steps of the algorithm, thus giving us an actual advantage versus simply applying both original algorithms separately.

We think that this project has potential to create a new paradigm for appearance filtering and distribution effects by more tightly coupling each step of the rendering process, thus giving us access to more information and letting us exploit to to optimize visual fidelity and performance.

2.4. Final Thoughts

We believe that a more holistic approach to realistic camera models is key to broadening their use and ensuring that they remain efficient to process in rendering systems. With just a select few considerations, we can combine different parts of light transport and evaluate them simultaneously or reuse information across different steps. The future work we have outlined above make it an extremely exciting and interesting field which should see substantial growth in the next few years as rendering engine developers and researchers seek to further improve performance and efficiency while producing ever more realistic images.

References

- Tomas Akenine-Möller, Eric Haines, Naty Hoffman, Angelo Pesce, Michał Iwanicki, and Sébastien Hillaire. *Real-Time Rendering 4th Edition*, chapter 10.6, pages 425–433. A K Peters/CRC Press, Boca Raton, FL, USA, 2018.
- [2] Tomas Akenine-Möller, Jacob Munkberg, and Jon Hasselgren. Stochastic rasterization using time-continuous triangles. In *Graphics Hardware*, pages 7–16. Eurographics, 2007.
- [3] Johan Andersson and Colin Barré-Brisebois. Shiny pixels and beyond: Real-time raytracing at seed. GDC 2018.
 EA SEED, 2018.
- [4] Alvy Ray Smith (Director) & John Lasseter (Animator). The Adventures of André and Wally B. Lucasfilm, 1984.
- [5] Adam Arbree, Bruce Walter, and Kavita Bala. Single-pass scalable subsurface rendering with lightcuts. In Computer Graphics Forum, volume 27, pages 507–516. Wiley Online Library, 2008.
- [6] Asen Atanasov and Vladimir Koylazov. A practical stochastic algorithm for rendering mirror-like flakes. In ACM SIGGRAPH 2016 Talks, SIGGRAPH '16, pages 67:1–67:2, New York, NY, USA, 2016. ACM.
- [7] Mahdi Bagher, Cyril Soler, and Nicolas Holzschuch. Accurate fitting of measured reflectances using a Shifted Gamma micro-facet distribution. *Computer Graphics Forum*, 31(4), June 2012.
- [8] Mahdi M. Bagher, John Snyder, and Derek Nowrouzezahrai. A non-parametric factor microfacet model for isotropic brdfs. *ACM Transactions on Graphics*, 36(6), August 2016.
- [9] Mahdi M. Bagher, Cyril Soler, Kartic Subr, Laurent Belcour, and Nicolas Holzschuch. Interactive rendering of acquired materials on dynamic geometry using bandwidth prediction. In ACM 13D, pages 127–134, 2012.
- [10] P. Beckmann and A. Spizzichino. The scattering of electromagnetic waves from rough surfaces. *New York: Pergamon*, 1963.
- [11] Laurent Belcour, Kavita Bala, and Cyril Soler. A local frequency analysis of light scattering and absorption. *ACM Trans. on Graph.*, 33(5):163:1–163:17, September 2014.
- [12] Laurent Belcour and Cyril Soler. Frequency based kernel estimation for progressive photon mapping. In SIG-GRAPH Asia 2011 Posters, page 47. ACM, 2011.
- [13] Laurent Belcour, Cyril Soler, Kartic Subr, Nicolas Holzschuch, and Fredo Durand. 5d covariance tracing for efficient defocus and motion blur. ACM Trans. Graph., 32(3):31:1–31:18, July 2013.
- [14] Laurent Belcour, Guofu Xie, Christophe Hery, Mark Meyer, Wojciech Jarosz, and Derek Nowrouzezahrai. Integrating clipped spherical harmonics expansions. ACM Trans. Graph., 37(2):19:1–19:12, March 2018.
- [15] Laurent Belcour, Ling-Qi Yan, Ravi Ramamoorthi, and Derek Nowrouzezahrai. Antialiasing complex global illumination effects in path-space. *ACM Trans. Graph.*, 36(1), January 2017.
- [16] John Bowers, Rui Wang, Li-Yi Wei, and David Maletz. Parallel poisson disk sampling with spectrum analysis on surfaces. ACM Trans. on Graph., 29(6):166:1–166:10, December 2010.

- [17] Norbert Bus, Nabil H. Mustafa, and Venceslas Biri. IlluminationCut. Computer Graphics Forum (Proceedings of Eurographics 2015), 34(2):561 – 573, 2015.
- [18] Zina H. Cigolle, Sam Donow, Daniel Evangelakos, Michael Mara, Morgan McGuire, and Quirin Meyer. A survey of efficient representations for independent unit vectors. *Journal of Computer Graphics Techniques (JCGT)*, 3(2):1–30, April 2014.
- [19] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. ACM Trans. Graph., 1(1):7–24, January 1982.
- [20] Robert L. Cook. Stochastic sampling in computer graphics. ACM Trans. Graph., 5(1):51–72, 1986.
- [21] Robert L. Cook and Tony DeRose. Wavelet noise. Transactions on Graphics, 24(3):803–811, July 2005.
- [22] Cyril Crassin, David Luebke, Michael Mara, Morgan McGuire, Brent Oster, Peter Shirley, Peter-Pike Sloan, and Chris Wyman. CloudLight: A system for amortizing indirect lighting in real-time rendering. *Journal of Computer Graphics Techniques (JCGT)*, 4(4):1–27, October 2015.
- [23] Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum*, 30(7):1921–1930.
- [24] Eugene d'Eon. A better dipole. http://www.eugenedeon.com/project/a-better-dipole/, Nov 2012.
- [25] Eugene d'Eon. A dual-beam 3d searchlight bssrdf. In ACM SIGGRAPH 2014 Talks, pages 65:1–65:1, 2014.
- [26] Eugene D'Eon and Geoffrey Irving. A quantized-diffusion model for rendering translucent materials. ACM Trans. Graph., 30(4):56:1–56:14, July 2011.
- [27] William Donnelly and Andrew Lauritzen. Variance shadow maps. In Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, I3D '06, pages 161–165, New York, NY, USA, 2006. ACM.
- [28] Samuel Donow. Light probe selection algorithms for real-time rendering of light fields. Master's thesis, Williams College, 2016.
- [29] Renaud Adrien Dubouchet, Laurent Belcour, and Derek Nowrouzezahrai. Frequency Based Radiance Cache for Rendering Animations. In Matthias Zwicker and Pedro Sander, editors, *Eurographics Symposium on Rendering -Experimental Ideas & Implementations*. The Eurographics Association, 2017.
- [30] Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Fabrice Neyret, and Victor Ostromoukhov. Linear efficient antialiased displacement and reflectance mapping. ACM Transactions on Graphics, 32(6):1–11, November 2013.
- [31] Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, and Victor Ostromoukhov. Extracting Microfacetbased BRDF Parameters from Arbitrary Materials with Power Iterations. *Computer Graphics Forum*, page 10, 2015.
- [32] Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X. Sillion. A frequency analysis of light transport. ACM Trans. Graph., 24(3):1115–1126, July 2005.
- [33] Kevin Egan, Yu-Ting Tseng, Nicolas Holzschuch, Frédo Durand, and Ravi Ramamoorthi. Frequency analysis and sheared reconstruction for rendering motion blur. ACM Trans. Graph., 28(3), 2009.
- [34] Daniel Evangelakos. A light field representation for real time global illumination, 2015.
- [35] Jeppe Revall Frisvad, Toshiya Hachisuka, and Thomas Kim Kjeldsen. Directional dipole model for subsurface scattering. ACM Trans. Graph., 34(1):5:1–5:12, December 2014.
- [36] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. J. Comput. Phys., (2):325–348, December 1987.
- [37] Jean-Philippe Guertin, Morgan McGuire, and Derek Nowrouzezahrai. A fast and stable feature-aware motion blur filter. In *High Performance Graphics*. ACM/Eurographics, June 2014.
- [38] Jean-Philippe Guertin, Morgan McGuire, and Derek Nowrouzezahrai. A fast and stable feature-aware motion blur filter. In *High Performance Graphics*. ACM/Eurographics, June 2014.

- [39] Ralf Habel, Per H. Christensen, and Wojciech Jarosz. Photon beam diffusion: A hybrid monte carlo method for subsurface scattering. In *Eurographics Symposium on Rendering*, pages 27–37, 2013.
- [40] Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. Multidimensional adaptive sampling and reconstruction for ray tracing. ACM Trans. Graph., 27(3), 2008.
- [41] Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. Frequency domain normal map filtering. *ACM Trans. Graph.*, 26(3), July 2007.
- [42] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *SIGGRAPH Comput. Graph.*, 24(4):145–154, September 1990.
- [43] Eric Heitz, Johannes Hanika, Eugene d'Eon, and Carsten Dachsbacher. Multiple-scattering microfacet bsdfs with the smith model. ACM Trans. Graph., 35(4):58:1–58:14, July 2016.
- [44] Eric Heitz, Stephen Hill, and Morgan McGuire. Combining analytic direct illumination and stochastic shadows. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '18, pages 2:1–2:11, New York, NY, USA, 2018. ACM.
- [45] Eric Heitz, Derek Nowrouzezahrai, Pierre Poulin, and Fabrice Neyret. Filtering color mapped textures and surfaces. In ACM Siggraph Symposium on Interactive 3D Graphics and Games, New York, NY, USA, 2013. ACM.
- [46] J.T. Hooker. Volumetric global illumination at treyarch. In Advances in Real-Time Rendering 2016, SIGGRAPH 2016. Treyarch, 2016.
- [47] Homan Igehy. Tracing ray differentials. In Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, pages 179–186, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.
- [48] Michal Iwanicki. Lighting technology of the last of us. In ACM SIGGRAPH 2013 Talks, SIGGRAPH '13, pages 20:1–20:1, New York, NY, USA, 2013. ACM.
- [49] Wenzel Jakob. Mitsuba renderer, 2021. http://www.mitsuba-renderer.org.
- [50] Wenzel Jakob, Miloš Hašan, Ling-Qi Yan, Jason Lawrence, Ravi Ramamoorthi, and Steve Marschner. Discrete stochastic microfacet models. ACM Trans. Graph., 33(4):115:1–115:10, July 2014.
- [51] Johannes Deligiannis Jan Schmid, Yasin Uludag. It just works: Ray-traced reflections in "battlefield v". Presented at GPU Technology Conference, 2019, 2019.
- [52] Adrian Jarabo, Raul Buisan, and Diego Gutierrez. Bidirectional clustering for scalable vpl-based global illumination. In CEIG, 2015.
- [53] Johannes Jendersie, David Kuri, and Thorsten Grosch. Real-Time Global Illumination Using Precomputed Illuminance Composition with Chrominance Compression. *Journal of Computer Graphics Techniques (JCGT)*, 5(4):8–35, December 2016.
- [54] Henrik Wann Jensen and Juan Buhler. A rapid hierarchical rendering technique for translucent materials. *ACM Trans. Graph.*, 21(3):576–581, July 2002.
- [55] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *ACM SIGGRAPH*, pages 511–518, 2001.
- [56] James T. Kajiya. The rendering equation. SIGGRAPH Comput. Graph., 20(4):143–150, August 1986.
- [57] Evangelos Kalogerakis, Patricio Simari, Derek Nowrouzezahrai, and Karan Singh. Robust statistical estimation of curvature on discretized surfaces. In *Eurographics Symposium on Geometry Processing*, pages 13–22.
- [58] Nickolay Kasyan and Nicolas Schulz. Secrets of cryengine 3 graphics technology. In *SIGGRAPH Talks*. ACM, 2011.

- [59] Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. Procedural noise using sparse Gabor convolution. *j*-*TOG*, 28(3), August 2009.
- [60] Sébastien Lagarde and Antoine Zanuttini. Local image-based lighting with parallax-corrected cubemap. SIG-GRAPH 2012. DONTNOD Entertainment, 2012.
- [61] Jaakko Lehtinen, Timo Aila, Jiawen Chen, Samuli Laine, and Frédo Durand. Temporal light field reconstruction for rendering distribution effects. *ACM Trans. Graph.*, 30(4):55, 2011.
- [62] Eric Lengyel. Motion blur and the velocity-depth-gradient buffer. In Eric Lengyel, editor, *Game Engine Gems*. Jones & Bartlett Publishers, March 2010.
- [63] Zander Majercik, Jean-Philippe Guertin, Derek Nowrouzezahrai, and Morgan McGuire. Dynamic diffuse global illumination with ray-traced irradiance fields. *Journal of Computer Graphics Techniques (JCGT)*, 8(2):1–30, June 2019.
- [64] Sam Martin and Per Einarsson. A real time radiosity architecture for video games. In *Advances in Real-Time Rendering 2010*, SIGGRAPH 2010. Geomerics and EA DICE, 2010.
- [65] Nelson L. Max and Douglas M. Lerner. A two-and-a-half-d motion-blur algorithm. In Proc. of SIGGRAPH, pages 85–93, NY, 1985. ACM.
- [66] Stephen McAuley. Calibrating lighting and materials in far cry 3. In *Practical Physically Based Shading in Film and Game Production*, SIGGRAPH 2012. Ubisoft Montreal, 2012.
- [67] Morgan McGuire. Computer graphics archive, July 2017. https://casual-effects.com/data.
- [68] Morgan McGuire, Eric Enderton, Peter Shirley, and David P. Luebke. Real-time stochastic rasterization on conventional GPU architectures. In *High Performance Graphics*, 2010.
- [69] Morgan McGuire, Padraic Hennessy, Michael Bukowski, and Brian Osman. A reconstruction filter for plausible motion blur. In *I3D*, pages 135–142, 2012.
- [70] Morgan McGuire, Michael Mara, and Zander Majercik. The G3D innovation engine, 01 2017. https:// casual-effects.com/g3d.
- [71] Morgan McGuire, Michael Mara, Derek Nowrouzezahrai, and David Luebke. Real-time global illumination using precomputed light field probes. In ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, page 11, February 2017.
- [72] Morgan McGuire, Michael Mara, and Others. G3D Innovation Engine, 2014. http://g3d.sourceforge.net/.
- [73] Max McMullen. Direct3D New Rendering Features, September 2014.
- [74] Fernando Navarro, Francisco J. Serón, and Diego Gutierrez. Motion blur rendering: State of the art. Computer Graphics Forum, 30(1):3–26, 2011.
- [75] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental Analysis of BRDF Models. In *Eurographics Workshop on Rendering*, 2005.
- [76] Greg Nichols and Chris Wyman. Multiresolution splatting for indirect illumination. In ACM I3D, pages 83–90, 2009.
- [77] Derek Nowrouzezahrai, Patricio Simari, and Eugene Fiume. Sparse zonal harmonic factorization for efficient sh rotation. *ACM Transactions on Graphics*, 2012.
- [78] Nvidia. Rtx global illumination, 2020.
- [79] Marc Olano and Dan Baker. Lean mapping. In Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, I3D '10, pages 181–188, New York, NY, USA, 2010. ACM.
- [80] Ryan S. Overbeck, Craig Donner, and Ravi Ramamoorthi. Adaptive wavelet rendering. *ACM Trans. Graph.*, 28(5), 2009.
- [81] Edwin Catmull & Fred Parke. A Computer Animated Hand, 1972.
- [82] Ken Perlin. Improving noise. Transactions on Graphics, 21(3):681–682, July 2002.

- [83] Dan Piponi. Polygon kernels for image processing, March 2013. US Patent 8,400,461 B1.
- [84] Boris Raymond, Gaël Guennebaud, and Pascal Barla. Multi-scale rendering of scratched materials using a structured sv-brdf model. ACM Trans. Graph., 35(4):57:1–57:11, July 2016.
- [85] W. T. Reeves. Particle systems a technique for modeling a class of fuzzy objects. *ACM Trans. Graph.*, 2(2):91–108, April 1983.
- [86] Erik Reinhard, Paul Debevec, Greg Ward, Karol Myszkowski, Helge Seetzen, Drew Hess, Gary McTaggart, and Habib Zargarpour. High dynamic range imaging: Theory and practice. SIGGRAPH 2006, 2006.
- [87] Matt Ritchie, Greg Modern, and Kenny Mitchell. Split second motion blur. In SIGGRAPH Talks, NY, 2010. ACM.
- [88] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect shadow maps for efficient computation of indirect illumination. In ACM SIGGRAPH Asia 2008 Papers, SIGGRAPH Asia '08, pages 129:1–129:8, New York, NY, USA, 2008. ACM.
- [89] Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, I3D '09, pages 75–82, New York, NY, USA, 2009. ACM.
- [90] Perumaal Shanmugam and Okan Arikan. Hardware accelerated ambient occlusion techniques on gpus. In Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games, I3D '07, pages 73–80, New York, NY, USA, 2007. ACM.
- [91] Ari Silvennoinen and Jaakko Lehtinen. Real-time global illumination by precomputed local reconstruction from sparse radiance probes. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia), 36(6):230:1–230:13, 11 2017.
- [92] Peter-Pike Sloan, Naga K. Govindaraju, Derek Nowrouzezahrai, and John Snyder. Image-based proxy accumulation for real-time soft global illumination. In *Proceedings of Pacific Graphics*, pages 97–105, USA, 2007. IEEE.
- [93] Peter-Pike Sloan, Ben Luna, and John Snyder. Local, deformable precomputed radiance transfer. In SIGGRAPH, NY, USA, 2005. ACM.
- [94] Tiago Sousa. Cryengine 3 rendering techniques. In Microsoft Game Technology Conference. August 2011.
- [95] Tiago Sousa. Graphics gems from cryengine 3. In ACM SIGGRAPH Course Notes, 2013.
- [96] Tomasz Stachowiak and Yasin Uludag. Stochastic screen-space reflections. In Advances in Real-Time Rendering 2015, SIGGRAPH 2015. EA DICE, 2015.
- [97] Natalya Tatarchuk, Chris Brennan, and John R. Isidoro. Motion blur using geometry and shading distortion. In Wolfgang Engel, editor, *ShaderX2: Shader Prog. Tips & Tricks with DirectX 9.0.* 2003.
- [98] Jonathan Thaler. Deferred rendering. 02 2011.
- [99] Michael Toksvig. Mipmapping normal maps. Journal of Graphics Tools 10, (3):65–71, 2005.
- [100] K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces*. J. Opt. Soc. Am., 57(9):1105–1114, Sep 1967.
- [101] Robert Toth, Jon Hasselgren, and Tomas Akenine-Möller. Perception of highlight disparity at a distance in consumer head-mounted displays. In *Proceedings of the 7th Conference on High-Performance Graphics*, HPG '15, pages 61–66, New York, NY, USA, 2015. ACM.
- [102] Michal Valient. Killzone shadow fall demo postmortem. Sony Devstation 2013. Guerilla Games, 2013.
- [103] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '95, pages 419–428, New York, NY, USA, 1995. ACM.

- [104] Zdravko Velinov, Sebastian Werner, and Matthias B. Hullin. Real-Time Rendering of Wave-Optical Effects on Scratched Surfaces. *Computer Graphics Forum*, 2018.
- [105] Bruce Walter. Notes on the ward brdf. Technical report, Cornell Program of Computer Graphics, April 2005.
- [106] Bruce Walter, Adam Arbree, Kavita Bala, and Donald P Greenberg. Multidimensional lightcuts. ACM Transactions on Graphics (TOG), 25(3):1081–1088, 2006.
- [107] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: A scalable approach to illumination. ACM Trans. Graph., 24(3):1098–1107, July 2005.
- [108] Bruce Walter, Stephen Marschner, Hongsong Li, and Keneth Torrance. Microfacet models for refraction through rough surfaces. *Eurographics Symposium on Rendering*, 2007.
- [109] Jingwen Wang and Ravi Ramamoorthi. Analytic spherical harmonic coefficients for polygonal area lights. ACM Trans. Graph., 37(4):54:1–54:11, July 2018.
- [110] Sebastian Werner, Zdravko Velinov, Wenzel Jakob, and Matthias B. Hullin. Scratch iridescence: Wave-optical rendering of diffractive surface structure. ACM Trans. Graph., 36(6):207:1–207:14, November 2017.
- [111] Chris Wyman. An approximate image-space approach for interactive refraction. ACM Trans. Graph., 24(3):1050– 1053, July 2005.
- [112] Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. Rendering glints on high-resolution normal-mapped specular surfaces. ACM Trans. Graph., 33(4):116:1–116:9, July 2014.
- [113] Ling-Qi Yan, Miloš Hašan, Steve Marschner, and Ravi Ramamoorthi. Position-normal distributions for efficient rendering of specular microstructure. ACM Trans. Graph., 35(4):56:1–56:9, July 2016.
- [114] Ling-Qi Yan, Miloš Hašan, Bruce Walter, Steve Marschner, and Ravi Ramamoorthi. Rendering specular microgeometry with wave optics. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018), 37(4), 2018.
- [115] Renaldas Zioma and Simon Green. Mastering DirectX 11 with Unity, March 2012. Presentation at GDC 2012.
- [116] Tobias Zirr and Anton S. Kaplanyan. Real-time rendering of procedural multiscale materials. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '16, pages 139–148, New York, NY, USA, 2016. ACM.