Cache Points for Production-Scale Occlusion-Aware Many-Lights Sampling and Volumetric Scattering

Yining Karl Li
Walt Disney Animation Studios
Burbank, USA
karl.li@disneyanimation.com

Charlotte Zhu
Walt Disney Animation Studios
Burbank, USA
charlotte.zhu@disneyanimation.com

Gregory Nichols*
Latitude AI
Pittsburgh, USA
greg@nichols.pro

Peter Kutz*
Adobe
San Francisco, USA
peter.kutz@gmail.com

Wei-Feng Wayne Huang*
NVIDIA
Los Angeles, USA
wahuang@nvidia.com

David Adler
Walt Disney Animation Studios
Burbank, USA
david.adler@disneyanimation.com

Brent Burley
Walt Disney Animation Studios
Burbank, USA
brent.burley@disneyanimation.com

Daniel Teece
Walt Disney Animation Studios
Burbank, USA
daniel.teece@disneyanimation.com

Figure 1: A production scene from *Us Again* containing 4881396 light sources (analytical lights, emissive triangles, and emissive volumes), rendered using 32 samples per pixel with uniform light selection (a), locally optimal light selection (b), and our cache points system (c). Uniform light selection produces a faster result but converges poorly, while building a locally optimal light distribution per path vertex produces a more converged result but is much slower. Our *cache points* system (c) produces a noise level similar to (b) while maintaining performance closer to (a). To clearly show noise differences, this figure does not include the post-renderer compositing that is present in the final production frame. © 2024 Disney

ABSTRACT

A hallmark capability that defines a renderer as a *production* renderer is the ability to scale to handle scenes with extreme complexity, including complex illumination cast by a vast number of light sources. In this paper, we present Cache Points, the system used by Disney's Hyperion Renderer to perform efficient unbiased importance sampling of direct illumination in scenes containing up to millions of light sources. Our cache points system includes a number of novel features. We build a spatial data structure over points that light sampling will occur from instead of over the lights themselves. We do online learning of occlusion and factor this into our importance sampling distribution. We also accelerate sampling in difficult volume scattering cases.

*This author contributed to the presented work while employed at Walt Disney Animation Studios.

DigiPro '24, July 27, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0690-5/24/07
https://doi.org/10.1145/3665320.3670993

Over the past decade, our cache points system has seen extensive production usage on every CG feature film and animated short produced by Walt Disney Animation Studios, enabling artists to design lighting environments without concern for complexity. In this paper, we will survey how the cache points system is built, works, impacts production lighting and artist workflows, and factors into the future of production rendering at Disney Animation.

CCS CONCEPTS

• **Computing methodologies** → **Rendering**; **Ray tracing**.

KEYWORDS

path tracing, global illumination, light selection, importance sampling, volume rendering

ACM Reference Format:
Yining Karl Li, Charlotte Zhu, Gregory Nichols, Peter Kutz, Wei-Feng Wayne Huang, David Adler, Brent Burley, and Daniel Teece. 2024. Cache Points for Production-Scale Occlusion-Aware Many-Lights Sampling and Volumetric Scattering. In *The Digital Production Symposium (DigiPro '24), July 27, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3665320.3670993

1 INTRODUCTION

A major challenge in production rendering is light sampling in scenes containing anywhere from a small number to hundreds of thousands or even millions of light sources. Furthermore, the types of lighting scenarios that arise in any particular production are often unpredictable. A key principle in the design of Hyperion is an emphasis on simplicity over flexibility [Burley et al. 2017]: we try not to burden users with non-artistic controls as much as possible. In accordance with this principle, we prefer systems that are as sufficiently and automatically robust to as many production scenarios as possible; in this paper, we present an in-depth description of our system for guiding direct light sampling in our production scenes, from the simplest to the most complex lighting scenarios. Our system, *cache points*, builds locally optimal estimates for light sampling weights and incorporates an online learning metric for local light visibility estimates. Our system is able to (1) combine local estimates for analytical lights, emissive geometry, and emissive volumes into a single combined system, and (2) provide unbiased direct light sampling guiding for both surface points and points inside of participating media. Additionally, we have also extended our system for use in importance sampling volumetric in-scattering in participating media. While we have previously alluded to our cache points system for solving the many-lights sampling problem [Burley et al. 2018; Fong et al. 2017; Nichols and Eisenacher 2015] and have described cache points for volumetric scattering [Huang et al. 2021] at a high level, in this work we present a detailed, in-depth look into these methods and describe our technique as a whole (Sections 3 and 4), provide real-world production case studies (Section 5), and discuss potential future improvements (Section 6).

Figure 2: In addition to normal scenes with up to hundreds of lights (top), *Big Hero 6* featured scenes with anywhere from thousands of lights (middle), to hundreds of thousands of lights (bottom). Our cache points system was designed to efficiently and robustly perform many-lights sampling in all of these scenarios. © 2024 Disney

2 MOTIVATION AND RELATED WORK

Motivation. Production renderers often have to handle scenes with extraordinary levels of lighting complexity. Disney's Hyperion Renderer was first written for the production of *Big Hero 6*, which featured nighttime city scenes that contained as many as half a million small, bright, directional light sources (Figure 2). Later productions further increased the complexity level by including emissive mesh geometry and emissive volumes, resulting in scenes with millions of discrete light sources if each triangle in each emissive mesh and each emissive volume is considered separately (Figure 1). While sampling direct lighting from individual analytical lights using next event estimation and combining direct lighting samples with BSDF samples via multiple importance sampling [Veach 1998] is now well understood, automatically choosing which light to sample out of potentially millions of lights remains an active area of research.

Manual Light Grouping. One possible solution to managing lighting complexity is tasking artists with manually pruning low contribution lights from scenes, or requiring artists to run pilot renders,

gather statistics on light usage, and then build pruning into the production pipeline [Vavilala 2019]. However, in the spirit of seeking simple and efficient user workflows, we prefer to avoid any solution that requires manual intervention or additional pipeline complexity before artists can focus on creative work. Instead, we seek automatic in-renderer solutions to handling many-lights scenarios.

Hierarchical Light Trees. Hierarchical tree data structures for light selection are one of the most commonly used solutions to the many-lights problem today [Fascione et al. 2018; Georgiev et al. 2018; Gospodnetić 2017; Keller et al. 2017; Kulla et al. 2018; Pharr et al. 2023]. The particular variant of a hierarchical light tree used in Kulla et al. [2018] is further described in detail by Conty et al. [2018]. While we do use a hierarchical light tree approach for clustering triangles within individual emissive meshes, we found that a hierarchy-based approach had difficulties with certain use cases for overall global many-lights sampling. Specifically, we noticed that narrow, highly directional IES profile based lights were challenging for light trees to handle. Incorporating complex occlusion information into light trees also proved to be intractable. Overall, we see hierarchical tree data structures as a different but largely complimentary approach to our cache points; we discuss this further in Section 5.2.2.

Photon Mapping. The core of our cache points data structure consists of a large number of points scattered throughout the scene in world space, placed in a KD-tree for fast nearest-neighbor lookups. At a high level, this data structure strongly resembles a photon map [Jensen 1996, 2001]. However, our cache points differ from photon mapping in how the data structure is built and what it stores and in how it is used. Unlike photon mapping, which places photons at points throughout the scene using forwards light tracing from light sources, our cache points are placed throughout the scene using a combination of sampling points on surfaces and backwards path tracing from the camera. Our approach of storing cumulative distribution functions (CDFs) at each point has similarities with Jensen [1995]; however, instead of storing a rough approximation of irradiance for guiding importance sampling of indirect illumination, we store a high-quality estimate of direct illumination for use in many-lights sampling. Hyperion contains an adaptive photon mapping system [Burley et al. 2018]; we discuss how photon mapping and cache points could potentially be integrated with one another in Section 6.

VPLs. The many-lights sampling problem historically has also been extensively studied as part of *virtual point light* (VPL) methods [Dachsbacher et al. 2014]. VPL methods focus on solving the indirect illumination problem by discretizing illumination into large numbers of virtual point lights at path vertices; the contributions from these point lights must then be summed. Performing this summing operation without having to loop over every point light is the focus of Lightcuts [Walter et al. 2005] and other related variations [Davidovič et al. 2012; Pantaleoni 2019]. Much like VPL methods, our method's core data structure is an unstructured collection of points generated in part from path vertices, but beyond this, we consider the problem we are focused on to be orthogonal to the VPL literature. Instead of focusing on gathering illumination from individual light sources, we consider the problem of selecting an individual light out of a large set of possible candidates for next event estimation.

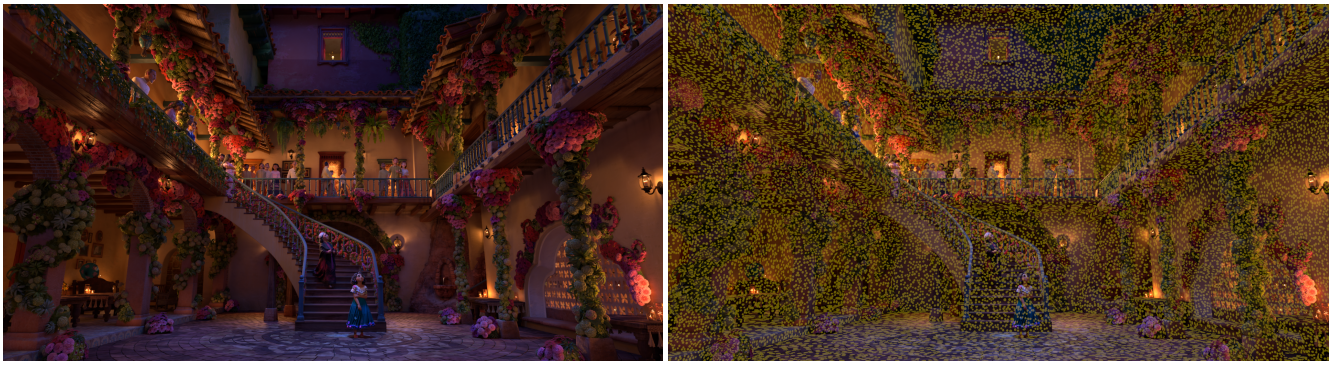
Learning Techniques. Our approach bears a strong high-level resemblance to later work by Vevoda et al. [2018], which expands upon Vevoda et al. [2016]. They similarly use a learning technique to derive light selection probabilities incorporating visibility information. However, the details of their approach differs greatly from ours, including their use of a light hierarchy as the underlying data structure versus our use of unstructured points. Our approach likely also shares high-level similarities with Nichols [2016], although not enough detailed information about their approach is available for us to make a more thorough comparison.

ReSTIR. More recently, reservoir-based spatiotemporal importance resampling, or ReSTIR [Bitterli et al. 2020], has seen much active research. The ReSTIR family of techniques [Lin et al. 2022; Wyman et al. 2023] builds upon resampled importance sampling [Talbot et al. 2005] to resample lights from candidate sample pools that are rapidly built by sharing samples spatially and temporally based on weighted reservoir sampling [Chao 1982], allowing for rapid efficient sampling of extremely complex direct lighting scenarios in interactive GPU ray tracing contexts. ReSTIR is reliant on an initial candidate generation strategy for reservoir sampling; Bokansky et al. [2021] and Tokutshi et al. [2021] focus on this problem. Another recent work comparable to ReSTIR is Dittebrandt et al. [2023]. We see our method as potentially complementary to ReSTIR and similar approaches; if our method was to be adapted into a progressive format, it could serve as an initial candidate generation strategy, although additional consideration would need to be given for dynamic lights.

Path Guiding. Path guiding techniques such as OLPMM [Vorba et al. 2014], Practical Path Guiding [Müller 2019; Müller et al. 2017], Zero-Variance Random Walk Guiding [Herholz et al. 2019], and others [Guo et al. 2018; Herholz et al. 2016; Rath et al. 2020; Ruppert et al. 2020; Vorba et al. 2020] all rely on learning about illumination throughout the scene in an online process and building some variant of a spatial acceleration structure to guide sampling of indirect illumination. Path guiding techniques are distinct from many-lights sampling techniques in that path guiding focuses on *indirect* illumination, whereas many-lights sampling techniques focus on direct illumination; generally path guiding and many-lights sampling techniques are orthogonal but complementary to each other. In Hyperion, we use a combination of our cache points system for direct illumination and optionally Practical Path Guiding for indirect illumination.

3 CACHE POINTS FOR MANY-LIGHTS SAMPLING

The cache points system begins with a simple observation: if computational time and memory were not concerns, then the theoretical best possible light selection strategy (in terms of minimizing noise) would be to estimate the contribution from every single light, factoring in occlusion, for each given path vertex and from all of these estimates, build a single probability distribution from which to draw a light to sample. We refer to this theoretical best possible strategy as *locally optimal light selection*. In practice, this approach



(a) Production scene **(b) Cache point locations overlaid in yellow**

Figure 3: A production scene from *Encanto* (a) containing 38720 light sources (analytical lights, emissive triangles, and emissive volumes) sampled using cache points (b). Initial cache point placement suggested 388888 candidate cache point locations, which were pruned down to 48364 final cache point locations. For illustrative purposes, in this figure we only show cache point locations on surfaces; in actuality we also populate cache points volumetrically in space to support light sampling in participating media. © 2024 Disney

is infeasible for all but the simplest scenes due to the computational time required; early tests on *Big Hero 6* using this approach resulted in renders with impressively low noise levels, but long render times dominated by light selection. Furthermore, factoring in occlusion when building a locally optimal light selection strategy per path vertex means that the the computational time required per path vertex can be highly variable; with large numbers of lights and difficult occlusion, the computational time can be orders of magnitude greater than for a simpler strategy (demonstrated in Figure 1), while in other cases the computational time can be a smaller multiple over a simpler strategy (demonstrated in Figure 4).

Instead of building a perfect locally optimal light sampling distribution at every path vertex, our system focuses on caching and reusing local light selection information in a sparse point-based data structure. Conceptually, our approach can be thought of as performing lossy compression on a perfect locally optimal light sampling distribution. Since we use this distribution for sampling and not direct evaluation of the direct lighting term, the result of using an approximation of the locally optimal light sampling distribution is unbiased but trades off increased noise in return for less computational workload and memory usage. Our objective is therefore to decrease computational workload and memory usage as much as possible, while getting as close to the quality of a perfect sampling distribution as possible.

3.1 Building and Initializing the Cache Points Data Structure

Building and initializing the cache point system occurs using the following steps:

(1) Generate an initial set of candidate spatial locations for placing cache points
(2) Merge spatially similar candidate locations
(3) Build a light distribution for each cache point
(4) Merge neighboring cache points with similar light distributions
(5) Blur light distributions between cache points

In this section, we describe each of these steps in detail.

3.1.1 Initial Candidate Point Generation. Our system begins by generating an initial set of 100,000 candidate points randomly distributed within the individual bounding boxes of all objects within the scene. Candidate points are distributed between bounding boxes proportional to their volume. Note that candidate points are not placed in the bounding boxes of objects that we will not perform lighting sampling from, such as area lights or objects with zero reflectance. We then trace a small number of pilot paths through the scene from the camera and select a random subset of path vertices from these pilot paths; additional candidate points are placed on these path vertices. This includes paths that enter and scatter through volumes.

To ensure a fixed upper bound on the amount of memory that the cache points system will use up, we cap the number of candidate points generated from pilot path vertices at 1,000,000. This strategy gives us a good combination of points on surfaces and points distributed through empty space to account for participating media.

If our initial seeding process generates fewer than 10 cache points, we do not bother with the rest of the cache points build process and instead simply fall back to calculating perfect locally optimal light distributions per path vertex.

Before we proceed any further, we spatially sort all of these initially-at-most 1.1 million candidate points and assign each candidate point an index. We then place the candidate points into an initial KD-tree, which allows us to perform nearest-neighbor lookups to prune the candidate point set into the final set of points we will build light distributions across. Next, we use a two-pass approach to prune the set of candidate points; the first pass occurs before we build any light distributions, and the second pass occurs after we have built light distributions for each point that survived the first pass.

3.1.2 Merging Spatially Similar Points. In the first pruning pass, we merge points that fall within a minimum radius of a pre-existing point are pruned. Our minimum radius heuristic works as follows: for each candidate point, we use a kNN search to find the 20 nearest neighboring points within a 1e-5 units radius and merge those points together into a single surviving candidate point. We perform this operation by first performing a parallel loop where for each point we find its 20 nearest neighbors and then find the point with the lowest index value; all points except the point with the lowest index are atomically marked for deletion. Next, we perform a serial loop to remove all points marked for deletion and rebuild the KD-tree with the surviving points. The choice of 20 as the number of neighbors to use in this step is somewhat arbitrary; empirically, this number seems to do a good job of balancing look-up efficiency (by avoiding undesirable clustering and reducing the overall point count) with time taken to distribute and prune the points.

3.1.3 Building Light Distributions. Next, we build a light distribution at each surviving point from the first pass. The light distribution at each cache point actually consists of two separate distributions: a *nearby* light distribution, and a *far* light distribution. We make the determination of whether a given light e is close or far away to cache point p using several metrics. First, we always consider infinite lights (dome lights, distant / solid-angle lights) to be far away. All finite lights that have a large solid angle relative to the position of a cache point are considered to be close to the cache point; we use the following fast heuristic to determine this:

$$isNear(e, p) = distance(x_e, x_p)^2 \leq (r_p * D)^2 \tag{1}$$

where x_e is the centroid of the light, x_p is the position of cache point p, r_p is the radius of cache point p, and D is an adjustment term for cache point separation distance, which we have empirically determined should be set to 4.0 for best results.

We separate nearby lights from further away lights because the irradiance from nearby lights potentially can be highly varying relative to position and surface normal direction. The nearby light distribution places all of its members into a single bin, whereas the far light distribution is actually made up of seven bins corresponding to seven imaginary sensors at the cache point location: six oriented planes facing the cardinal directions and one omnidirectional receiver at the center of the point. We loop over all lights in the scene and estimate the total contribution each light would make to each sensor ignoring occlusion, and we store a list in each bin of between 4 to 256 lights that account for up to 97 percent of the energy reaching the point [Shirley et al. 1996].

We do not precompute anything for the nearby light distribution; instead, at render-time we build a light selection PDF on-the-fly for each path vertex over the irradiance contributions from the nearby lights to the exact path vertex location. We describe this in more detail in Section 3.3. For each light in the six cardinal bins in the far light distribution, we directly store a precomputed irradiance estimate at the cache point location. For each light in the omnidirectional bin in the far light distribution, we directly store a precomputed direct fluence estimate at the cache point location. The omnidirectional bin serves a special purpose: for surfaces that have a well defined normal, estimating irradiance makes sense since irradiance is integrated over an oriented 2D surface, but for cases where a well defined normal is either difficult or impossible to define, we rely on a direct fluence estimate instead since direct fluence is integrated over a 3D sphere. Since curve-based hair tends to have extremely complex and rapidly changing surface orientations and volumetric participating media has no defined surface normal, we use direct fluence for driving light sampling for curves and volumes.

At this stage we only estimate these values for each light but defer building CDFs and PDFs until render-time cache point lookup. There are two reasons for this: first, as mentioned earlier, for nearby lights we can build a higher-quality sampling distribution on-the-fly at render-time, and second, Hyperion supports a sophisticated light linking system, which means that we cannot determine which lights to exclude from a light distribution until render-time evaluation of light linking relationships has been carried out.

3.1.4 Merging Neighboring Points with Similar Light Distributions. After building a light distribution at each cache point, we perform a second pruning step that merges nearby cache points that share similar light distributions. Like in the first pruning step, for each cache point p, we gather the nearest 20 neighbors within a 1e-5 unit radius. We then calculate an average similarity metric M_{avg} between cache point p's light distribution and its nearest neighbors' light distributions. The similarity metric is calculated as follows: for two given sets of lights A and B, we find the intersection of A and B (meaning the set of lights that are common to both sets) and then calculate the similarity metric M as:

$$M = \frac{2 * size_{weighted}(intersection(A, B))}{size(A) + size(B)} \tag{2}$$

We denote size of the intersection of set A and set B as being *weighted* because we need to take into account the possibility that a given light may exist in both sets A and B but have different assigned probabilities in each set. So, instead of just counting up the number of lights in $intersection(A, B)$, we instead calculate a *similarity percentage* S for each light, which we define as:

$$S = 1 - \frac{|P_B - P_A|}{2 * \left(\frac{P_A + P_B}{2}\right)} \tag{3}$$

where P_A and P_B denote the probabilities for a given light in sets A and B, respectively. This definition for S works well so long as the probability for any given light is non-negative, which we guarantee since Hyperion's lights only permit positive emission values; for systems where lights support negative emission values [Foundation 2024], a modified metric would be required. The *weighted* size of the intersection of sets A and B is then defined as the sum of S for every light in the intersection. This approach for calculating a similarity metric between light distributions is somewhat ad-hoc, but in practice we have found that this approach works well. M_{avg} is then defined as simply the sum of M for every nearest neighbor cache point to p divided by the number of nearest neighbors found.

Next, we calculate the average distance between cache point p and its gathered nearest neighbors; we assign this average distance as an initial guess for the radius of cache point p. We then adjust the radius of the cache point to take into account how similar the cache point is to its nearest neighbors; this is done by simply

Figure 4: A production scene from *Encanto* containing 38720 light sources (analytical lights, emissive triangles, and emissive volumes) with complex occluding geometry with opacity masks, rendered using 32 samples per pixel with uniform light selection (a), locally optimal light selection (b), and our cache points system (c). Building locally optimal distributions on-the-fly performs relatively well in this case, while our cache points system performs no worse in terms of sampling quality while still requiring less render time. © 2024 Disney

multiplying the radius by M_{avg}. Since a smaller M_{avg} value indicates that the light distribution in p is relatively *dissimilar* from its nearest neighbors, meaning the direct illumination radiance field in this area of space changes at a higher frequency, shrinking the sphere of influence for p makes sense, and vice-versa. To prevent the radius reduction from inhibiting the effectiveness of a cache point, we limit this to 25% of the original size in world space and also guarantee a minimum projected screen space size (equivalent to 3 pixels).

Finally, we loop over the nearest neighbors found earlier within p's adjusted radius; for this subset of nearest neighbors, we preserve the point with the lowest index and atomically mark for deletion all of the other points in the subset. Since the radius is adjusted for similarity between cache point light distributions, we consider points that are inside of the adjusted radius to have light distributions that are sufficiently similar that they can just be merged.

All of the above in the second pruning step is performed in a parallel loop over all cache points; a serial loop is then performed to remove all cache points marked for deletion. After we have the final set of cache points, we rebuild the KD-tree a last time for use during rendering.

3.1.5 Blurring Light Distributions Across Cache Points. After building light lists at each cache point and merging points with similar light distributions, the last step in our cache points data structure build process is to *blur*, or aggregate, far light distributions between cache points. This blurring steps allows each cache point's far light distribution to be influenced by the far light distributions in neighboring cache points, which serves to make all cache point light distributions more spatially conservative; this step allows us to safely only look up a single cache point per path vertex during path tracing. We don't blur the nearby light distributions since the change in nearby light distributions between neighboring cache points is often higher frequency than the change in far light distributions.

The blurring step for a given cache point p begins by finding the closest 16 neighboring cache points to p via kNN search. Note that in previous steps we used 20 neighbors for kNN search operations but in this step we choose 16; the rationale behind 16 is as follows. The maximum number of spheres of equal size that can be densely packed around another sphere of the same size in three dimensions is 12 [Dai et al. 2019; Hales et al. 2017], in either a face-centred cubic or hexagonal close packing configuration [Conway and Sloane 1999]. However, since our cache points have variable radii, we add an additional 4 points to the maximum perfect packing number of 12 as an empirically determined adjustment factor.

With our 16 nearest neighbors gathered, we next find the distance to the farthest gathered neighboring point d_{far}; we use d_{far} to determine the relative contributions of each of the gathered cache points to cache point p. To cache point p we assign a relative weight of 1, and to the closest gathered neighboring point we also assign a relative weight of 1, while the farthest gathered neighboring point is assigned a relative weight of 1/16. For point p_n in between the closest and farthest points, the relative weight is assigned as follows:

$$weight(p_n) = mix\left(1, \frac{1}{16}, \frac{distance(x, x_n)^2}{(d_{far})^2}\right) \tag{4}$$

We then normalize the relative weights to add up to 1; the reason the farthest point is assigned a relative weight of 1/16 is to ensure that after normalization, cache point p's 16 gathered neighbors will account for at least half of p's final blurred light distribution. For

Figure 5: A production scene from *Encanto* containing 4406 light sources (analytical lights, emissive triangles, and emissive volumes) with complex occlusion, rendered using 32 samples per pixel with uniform light selection (a), cache points with no learned visibility estimates (b), and cache points with learned visibility estimates (c). In this scene, using learned visibility estimates results in a 9.3% improvement in RMSE for little to no additional render time. © 2024 Disney

each of the 16 gathered neighbors, we then take all lights from the neighbor's light distribution, multiply them by the neighbor's normalized relative weight, and add those lights into p's light distribution. After blurring the light distributions between neighboring points, we leave the light distributions un-normalized and do not calculate CDFs yet; we postpone this step until we actually perform light sampling because we interpolate between the light distribution for each cardinal bin.

Note that in this step, in order to allow for parallel processing and in order to avoid repeated blurring of the same points, we cannot perform blurring in-place in memory; instead, we need to write the output blurred cache points to a new set of cache points and swap the memory afterwards.

3.2 Online Learning for Visibility Estimates

3.2.1 Visibility Estimates by Tracking Sample Ratios. Cache points are initially built without factoring in occlusion information; during the course of rendering, we improve sampling by learning a per-light visibility estimate at each cache point. Our approach bears some conceptual similarity to importance caching [Georgiev et al. 2012]. Each time we select a light from a cache point, we increment an internal sample attempt counter for that light within that cache point. We then perform direct lighting and in the event that the sample successfully reaches the light and receives a useful light contribution, we atomically increment an internal successful sample counter for that same light within that same cache point. Hyperion uses a batched wave-front path tracing architecture where the rendering process is divided up into a number of discrete iterations [Eisenacher et al. 2013]; between iterations, we use the ratio R between successful samples $H_{success}$ and total sample attempts H_{total} towards a given light to adjust the sampling weight of that light; lights with a lower ratio of successful sampling attempts are weighted down while lights with a higher ratio are weighted up. This process effectively corrects for cases where a bright light is initially identified as being important to a particular region of space but ends up not being important due to shadowing. The specific mechanism we use to assign every light e in a cache point p a visibility weight $W(p,e)$ based on the successful sampling attempt ratio R is as follows:

$$R(p,e) = \frac{H(p,e)_{success} + 1}{H(p,e)_{total} + 1} \tag{5}$$

$$W(p,e) = \begin{cases} \left(\frac{R(p,e)}{R_{min}}\right)^2 & R(p,e) \le R_{min} \\ 1.0 & \text{otherwise} \end{cases} \tag{6}$$

R_{min} is a minimum ratio threshold; when R falls below R_{min}, we consider the light e as being effectively completely shadowed at cache point p and therefore a candidate for down-weighting. From production experience, we empirically determined 0.04 to be a good default value for R_{min}. Importantly, since cache points are relatively sparsely distributed and therefore cannot capture high-frequency shadow detail, in order to maintain an unbiased result we must ensure that the visibility metric can never weight a light's selection probability down completely to zero [Ward 1991]. To guarantee that down-weighting follows a relatively aggressive curve but never reaches exactly zero, we choose a quadratic falloff (Equation 6).

A single scalar visibility weight per light and quadratic falloff are relatively simple, but even so, we have found that in complex shadowing cases, the use of our visibility metric combined with the relatively high density of cache points can noticeably improve noise over no visibility metric at all. For example, in a room lit by sunlight

through small windows, our visibility estimate significantly reduces noise by weighting down the otherwise high selection probability for a bright sun in most of the room, while leaving the selection probability high in small pools of sunlight.

3.2.2 Blurring Visibility Estimates Across Cache Points. After we calculate an individual visibility per light per cache point, we then blur the visibility weights between cache points in a manner analogous to the light distribution blurring process described in Section 3.1.5. Using the same rationale as in 3.1.5, we choose the 16 closest points to the current point p and assign the gathered neighbors relative weights using Equation 4. We sum up the relative weights of all of the neighbors plus 1.0 for p and invert this sum to produce a normalization term that we normalize all of the relative weights by.

For each light e in p's light distribution, we multiply e's visibility weight $W(p, e)$ by p's normalized relative weight, and then we loop through all of the neighbors and if a neighboring point's light distribution also contains that light, we take the visibility weight $W(p_n, e)$ for that light in that neighbor, multiply by that neighbor's normalized relative weight, and add that weight to p's visibility weight for e.

Blurring the visibility weights allows us to prevent sharp discontinuities in noise at the boundary between cache points from when the visibility term to a given light changes dramatically between neighboring cache points.

The effect of the online learned visibility estimate system is that renders with complex occlusion initially converge slowly, but as the learning system's quality improves during the renderer's initial iterations, the convergence rate improves as the renderer is able to make better and better light selection decisions. Since the visibility estimate is just another metric that feeds into the cache point update mechanism, utilizing visibility estimates adds little to no additional overhead to the system (Figure 5).

3.3 Light Selection from Cache Points

At each path vertex, we perform light selection by first selecting the nearest cache point to the path vertex via kNN search; because the light distribution blurring step in the build process results in every cache point already containing information from its neighboring cache points, we can get away with only selecting a single cache point per path vertex. For volumetric scattering, in order to avoid excessive or redundant cache point lookups, we also store the previously used cache point and re-use it if the current path vertex is still within that cache point's radius.

For a path vertex on a regular surface, we loop over the lights in the cache point's nearby light distribution and evaluate the irradiance contribution for the exact path vertex location, and for lights in the further away cardinal bins, we approximate the irradiance by using the stored precomputed irradiance estimate at the cache point location. The cardinal bins are combined, weighting by similarity between each bin's cardinal direction and the path vertex's shading normal. The weights for the cardinal bins are normalized to add up to one, which makes sure that the combined weighted cardinal bins still produce a normalized light distribution. We then assign a weight to each light in our combined list of lights from the near and far distributions; we directly use each light's irradiance

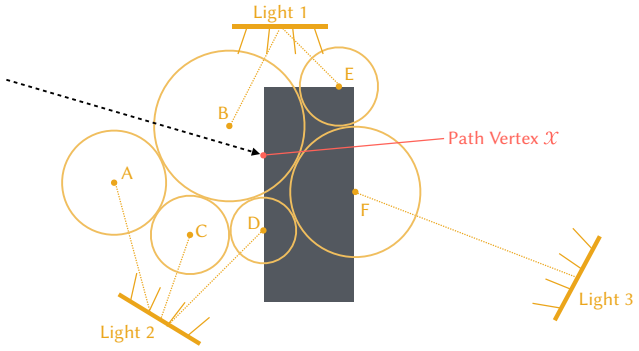


Figure 6: A simple 2D-example of a cache point system and a ray looking up a light distribution from a cache point. Each yellow dot is a cache point, with the radius of each cache point indicated by the outer yellow circle surrounding the point; each cache point is labeled with a letter. Cache points D, E, and F were placed on the surface of the grey object by path vertices during the intial cache point population phase, while cache points A, B, and C were placed in empty space by being distributed inside of a bounding box. A yellow dotted line from each cache point leads to the most important light in the cache point's light distribution. Lights 1, 2, and 3 are all equal intensity and the same size and shape. Cache points A, C, and D's most important light is the closest light, Light 2, while cache points B and E most important light is their closest light, Light 1. Although Light 1 is closer to cache point F, cache point F's visibility estimate over time has learned that Light 1 is occluded from cache point F, so cache point F's most important light is now Light 3. When a ray enters the scene and hits the grey object at path vertex x, the renderer looks up the closest cache point to x via kNN search through the cache point KD-tree and finds cache point B. From cache point B, the renderer then learns that the most important light to sample at path vertex x is Light 1.

estimate as its un-normalized sampling weight. We then remove any lights that are disabled for the current path vertex through light linking and add back in any lights that are marked as exclusive to the current path vertex via light linking. Finally, we multiply the un-normalized sampling weight for each light by the cache point's learned visibility weight for that light.

For a path vertex that belongs to volumetric scattering or that is on a curve with a hair shader [Chiang et al. 2016], we instead calculate the direct fluence for the exact path vertex location for nearby lights and use the omnidirectional bin in the far light distribution to get precomputed direct fluence estimates for far lights. The direct fluence estimates are then used as the un-normalized sampling weights, and then we apply the same light linking and visibility weight adjustments as in the regular surface case.

A CDF and PDF over the sampling weights is then generated, and the resultant probability distributions are then used to select a light for next event estimation. Since the light distribution at each cache point only accounts for at most 97 percent of the energy reaching the point, we also give each path vertex a small probability

to randomly select a light from all light sources in the scene. Because each cache point typically only contains a relatively small number of nearby important lights, scenes with enormous numbers of lights become tractable to efficiently render since the proportion of lights that need to be considered for a locally optimized light selection distribution can be capped to a small fixed maximum number. Figure 6 illustrates a simple 2D example of a small cache point distribution and the renderer using cache points to determine which light to important sample at a path vertex.

One interesting side effect of our cache points method is that despite the small additional overhead required for initializing the cache points system and the per-path-vertex overhead of looking up a cache point, sometimes renders using cache points can still be competitive in render time to renders utilizing a less optimal but less computationally complex light selection strategy; this is the case in Figure 5. This effect tends to occur in scenes with extreme complex occluding geometry; cache points can effectively steer the renderer away from needlessly casting shadow rays through complex occluding geometry, improving overall shadow ray traversal performance. Occluding geometry with opacity masks tends to make this performance difference even more dramatic, since evaluating opacity masks adds additional shading complexity on top of increased traversal complexity for shadow rays.

At this point, an advantage of our approach with respect to handling infinite lights becomes clear. In hierarchy-based approaches, infinite lights are often a special case that needs to be handled in an ad-hoc manner because infinite lights, by their nature of having no definable surface area or location in space, cannot have well-defined bounding boxes and therefore cannot be easily included in a BVH or any other kind of spatial acceleration structure. As a result, hierarchy-based approaches typically have to define some mechanism for deciding the ratio of light samples to send towards finite lights in the light hierarchy versus light samples to send towards infinite lights. However, since our approach builds a spatial data structure over the points we want to perform light sampling from instead of over the lights themselves, we can handle infinite lights just like any other finite light type in the far light distribution and correctly weight infinite lights in our final per-path-vertex light selection PDF.

4 CACHE POINTS FOR VOLUMETRIC SCATTERING

In addition to using cache points for learning optimal local light selection distributions, we also use the cache points system to learn distributions for importance sampling volumetric in-scattering. When combined with an extension we have made to null-collision theory for efficiently gathering emission from heterogeneous volumes, our cache point strategy for volumetric in-scattering gives us the ability to efficiently render previously difficult cases such as highly emissive volumes with low extinction (e.g. fire and flames) embedded in optically thin media, or thin anisotropic media with highly directional direct lighting (e.g. god rays and light shafts).

Figure 7: Production frames from some our recent films depicting scenes containing huge numbers of lights. Our artists routinely create huge number of lights for effects ranging from vast city scapes to magical sparkles and particles; cache points allows us to render all of these scenarios efficiently. © 2024 Disney

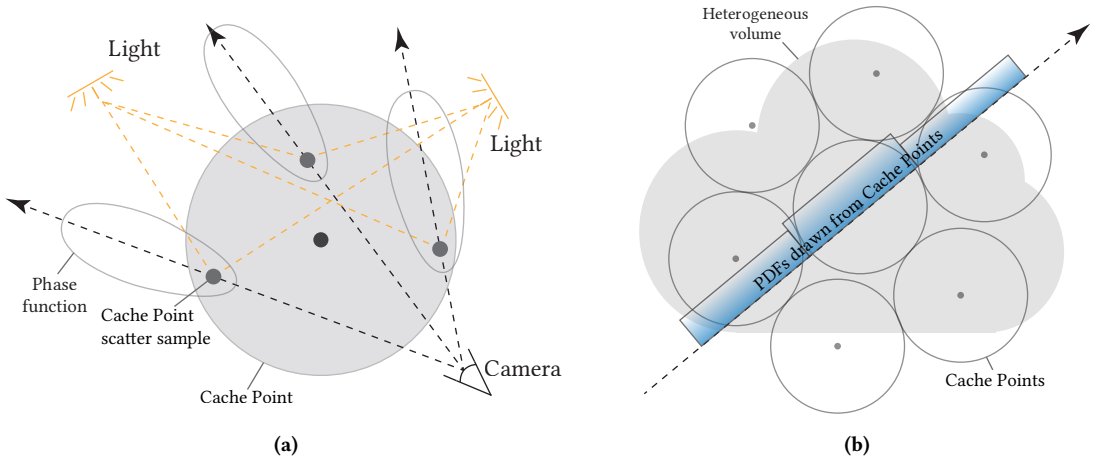


Figure 8: Scattering sample weight computation (a) and distance sampling CDF built on-the-fly from cached PDFs (b).

4.1 Null-collision Review

To best describe how we use cache points for importance sampling volumetric in-scattering, we must first briefly review the role in-scattering plays in volume rendering. We first review the null-collision integral formulation of the volume rendering equation, which evaluates the radiance at location $\mathbf{x}$ in direction ω within distance d:

$$L(\mathbf{x},\omega) = \int_0^d \bar{T}(\mathbf{x},\mathbf{y}) \Big(\mu_a(\mathbf{y})L_e(\mathbf{y},\omega) + \mu_s(\mathbf{y})L_s(\mathbf{y},\omega) + \mu_n(\mathbf{y})L(\mathbf{y},\omega)\Big)dt, \quad (7)$$

where $\mathbf{y} = \mathbf{x} - t \times \omega$. Null-collision techniques add imaginary null particles to heterogeneous volumes, producing an imaginary homogeneous volume through which free-flight distances along rays can be analytically sampled with a PDF proportional to the combined transmittance $\bar{T}$ formed by a constant combined extinction coefficient $\bar{\mu}$. $\bar{\mu}$ in turn is the sum of volume coefficients of three types of possible events: absorption μ_a, scattering μ_s and null-collision μ_n. These events can then be selectively evaluated using Monte Carlo estimators with probabilities P_s (for scattering events), P_a (for absorption events), and P_n (for null-collision events).

4.2 Volumetric In-scattering Sampling

We use our cache points system to learn an important term in Equation 7: product of scattering μ_s and in-scattered radiance L_s, which combined gives the result of volumetric in-scattering. During the cache point initialization process when we loop over every light for every cache point, in addition to estimating the total contribution the light makes at each cache point, we also calculate a *scattering sample weight* s, which approximates the integral of the product of $\mu_s(\mathbf{y})$, incoming radiance $L(\mathbf{y},\omega')$ and phase function $\rho(\mathbf{y},\omega,\omega')$ over solid angle, where $\mathbf{y}$ is a randomly sampled location within the cache point's radius of influence and ω' and ω are directions from $\mathbf{y}$ to a randomly sampled location on the light source and to the camera origin respectively (Figure 8a). This weight s is stored per light per cache point (Figure 8b).

During volumetric path tracing, we query the nearest cache points along the ray and use the weight s stored in each cache point to build a piece-wise linear 1D CDF to draw scattering samples from Figure 8b. Since building this 1D CDF can represent a large amount of overhead per ray in some scenarios, we currently only use this sampling strategy for direct lighting samples along camera rays. Limiting the use of this technique to only camera rays still allows us to noticeably improve visually prominent single-scattering effects while keeping the overall performance overhead low.

To efficiently render cases such as high-order scattering in optically thick volumes, we combine our technique with conventional transmittance-based sampling using multiple importance sampling (MIS) [Miller et al. 2019]. We start by using the 1D CDF to pick a scattering point $p_{select}(x_k)$, and then we use ratio tracking moving towards the scattering point to update the path's PDF. This process is repeated until distance sampling steps the ray through the selected scattering point; we can represent this process as:

$$p_{cachepoint}(\bar{x}) = p_{select}(x_k)\bar{T}(x_0,x_1)\bar{\mu}(x_1)\bar{T}(x_1,x_2) \bar{\mu}(x_2)\bar{T}(x_2,x_3)...\bar{u}(x_{k-1})\bar{T}(x_{k-1},x_k) \quad (8)$$

where each $\bar{T}(x_n,x_{n+1})\bar{\mu}(x_n)$ is the result of step n. To formulate the same path using null-collision tracking to get the PDF, we use the sampled distance and $\bar{\mu}$ to compute $\bar{T}$, and we already know P_n and P_s based on our choice of tracking algorithm. For all of the path vertices found before our selected scattering point, we apply the PDF P_n and repeat the distance sampling process and update the corresponding PDFs until we reach the selected scattering point and apply PDF P_s; this process gives us:

$$p_{null}(\bar{x}) = \bar{T}(x_0,x_1)\bar{\mu}(x_1)P_n(x_1)\bar{T}(x_1,x_2)\bar{\mu}(x_2) P_n(x_2)...\bar{T}(x_{k-1},x_k)\bar{\mu}(x_{k-1})P_s(x_k) \quad (9)$$

While the path PDFs represented by Equations 8 and 9 look very long, most of the terms cancel out to form a much simpler final expression for the MIS weight:

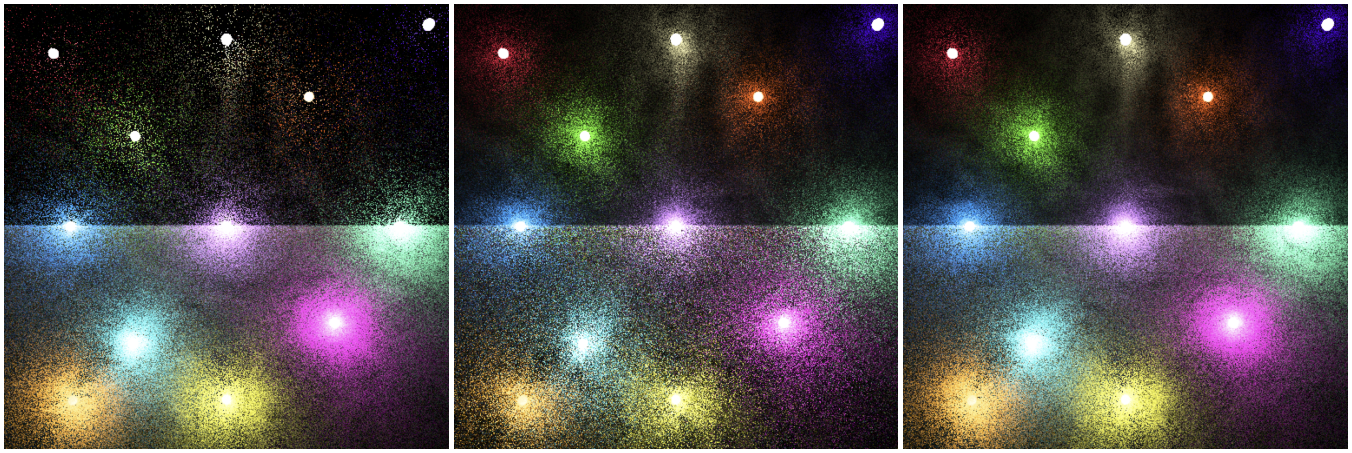


Figure 9: A scene consisting of bright lights embedded in a heterogeneous volume with low (top) and high (bottom) extinction coefficients. Null-collision tracking alone (left) does not work well with thinner volumes; our cache points based technique (middle) performs well with thin volumes but has trouble with thicker volumes. Combining both techniques through multiple importance sampling (right) efficiently samples both the thin and thick volume cases. Results shown are equal sample. © 2024 Disney

$$p_{null}(\bar{x}) : p_{probes}(\bar{x}) = P_n(x_1)...P_n(x_{k-1}) \\ \bar{\mu}(x_k)P_s(x_k) : p_{select}(x_k) \quad (10)$$

We demonstrate our technique working in conjunction with conventional null-scattering through MIS in an equal-sample comparison in Figure 9 and in an equal-time production comparison in Figure 10.

Compared to equi-angular sampling [Kulla and Fajardo 2012], our cache points based method performs better when rendering highly anisotropic volumes since our approach effectively factors in the phase function term. Additionally, our approach bypasses the need to sample a light vertex before performing distance sampling, instead, we rely on the cache points system's scattering sample weight s as a global estimate of direct illumination.

4.3 Volumetric Emission Sampling

In the case where a highly emissive heterogeneous volume is embedded in thin anisotropic media, our cache points method for sampling volumetric in-scattering needs to be combined with a method for efficiently gathering volumetric emission. The reason an additional method for gathering emission is required can be seen in how $\bar{\mu}$ typically chosen: as the majorant of $\mu_t = \mu_a + \mu_s$ and with medium events chosen with probabilities $P_t = \frac{\mu_t}{\bar{\mu}}$, $P_a = \frac{\mu_a}{\bar{\mu}}$, and $P_s = \frac{\mu_s}{\bar{\mu}}$. In cases where the volume emission function is strongly uncorrelated with the extinction function, such as in the case of highly emissive volumes with low extinction, null-tracking is likely to entirely step over potentially highly emissive regions.

We take advantage of the observation made by [Kutz et al. 2017] that $\bar{\mu}$, P_a, P_s, and P_n can be treated as arbitrary uncorrelated parameters as long as their contributions are counter-balanced by appropriate sample weights. To force ratio tracking [Novák et al. 2014] to take more steps in highly emissive regions, instead of setting $\bar{\mu}$ to the local maximum of μ_t, we choose $\bar{\mu}$ as $\bar{\mu} = max(\mu_t, \mu_a L_e)$. We always set $\bar{\mu}$ to be smaller than the average voxel sxize in order to avoid an excessive number of lookups within a single voxel. Since absorption and null-collision events don't require tracing new rays, we set $P_a = P_n = 1$, which results in the tracker gathering emission at every free-path sample, producing a higher quality emission estimate per ray (Algorithm 1).

Algorithm 1

```
function EVALUATEEMISSION(x, ω, d)
    w ← 1, L_e ← 0
    repeat
        Δt ← −ln(1−ζ)/μ̄
        x ← x − Δt × ω
        L_e ← L_e + w × μ_a(x)×L_e(x)/μ̄
        w ← w × μ_n(x)/μ̄
    until (t ← t + Δt) > d
    return L_e
end function
```

Next, in order to make our technique usable for next event estimation, we need a way to not just better evaluate emission from a heterogeneous volume, but also sample and evaluate the PDF of a sampled direction. To do the above, we begin by extending Villemin & Hery [2013]: we use an emission-energy-distribution grid, which is just a coarser version of the volume, in order to make sure that more emissive regions of the volume have a higher chance of receiving light samples. In Villemin & Hery [2013], point sampling is used, but point sampling can be sub-optimal when emission is occluded by heavy smoke or when the emissive region is large; in these cases, high sample counts are required to capture emission details in glossy reflections. Instead, we use our emission-optimized tracker to evaluate every tracking point along the ray, gathering more information in each light sample, effectively performing line integration. Finally, in order to use MIS to combine BSDF samples

Figure 10: A production scene from *Raya and the Last Dragon* containing a small bright light source embedded in a thin heterogeneous volume. Equal-time comparison of a conventional null-collision approach utilizing spectral-decomposition tracking (left) and incorporating our cache point based in-scattering sampling via MIS (right). Combining our cache points based in-scattering sampling with null-collision tracking produces a robust technique that works well both further from (top two rows) and closer to (bottom row) small bright light sources. © 2024 Disney

with our emissive volume light samples in the solid angle domain [Simon et al. 2017], we track which cells in the emission-energy-distribution grid that the light sample ray has passed through and integrate PDFs stored in each of these cells using a Jacobian transform:

$$p_\sigma(\omega) = \int_0^\infty p_x(t)t^2\mathrm{d}t$$
$$= P_0(t_1^3 - t_0^3)/3 + P_1(t_2^3 - t_1^3)/3 + P_2(t_3^3 - t_2^3)/3 \qquad (11)$$

We summarize this approach in Algorithm 2.

We combine our cache point based volumetric in-scattering sampling approach and our volumetric emission sampling approach using MIS to produce a single unified volume integrator, allowing us to efficiently sample strong light sources embedded in heterogeneous volumes with either low or high extinction coefficients (Figure 11).

Algorithm 2

```
function pdfEmission(x, ω)
    p = 0
    for voxel v along ray(x, ω) do
        [t0, t1] ← v entry/exit
        p ← p + (t1^3 − t0^3)/3 × pdf(v)
    end for
    return p
end function
```

5 PRODUCTION EXPERIENCES AND DISCUSSION

We have used our cache points system as the default light selection strategy on every film rendered using Disney's Hyperion Renderer; over the past decade, we have rendered millions of final frames and several orders of magnitude more work-in-progress frames using this approach with great success. Since the cache points system is enabled by default and all of its build processes are automatically carried out at renderer startup with no additional user intervention, user input, or user guidance required, our artists have not had to concern themselves with the number of lights they place in a scene. The cache points system has allowed us to meet art direction requirements calling for scenes with enormously complex illumination in many of our recent films without requiring any additional intervention from our TDs or the rendering team; see Figure 7. For example, the pier sequence from *Us Again* (Figure 1) contains approximately 4.8 million light sources; lighting and rendering process for this sequence was so routine and uneventful thanks to the cache point system that we did not realize how many lights were in this scene until after the sequence was completed.

All of the above is possible due to the large amount of effort that has gone into making cache points production-ready. Building a production-ready system requires considering how a system needs to fit into daily artist workflows and demonstrating a clear advantage in real-world production cases; we discuss some of these topics in Section 5.1. Another part of what makes a production-ready system production-ready is simply experience through real-world usage. In real-world usage, the challenging cases and failure cases are often just as interesting as the success cases (if not more so!); in Section 5.2 we provide several case studies of interesting challenges and failure cases we have encountered in production.

5.1 Performance Results and Determinism

When deploying novel rendering techniques in a production setting, the trade off between added overhead in both render time and memory versus overall benefit in terms of both convergence and artist workflow is a critical part of deciding whether or not

Figure 11: A production scene from *Raya and the Last Dragon* lit by torches (modelled as emissive heterogeneous volumes) embedded in thin anisotropic heterogeneous mist. Equal-time comparison of a conventional null-collision approach utilizing spectral-decomposition tracking (left), incorporating our emission sampling strategy (middle), and additionally combining with our scattering sampling strategy via MIS (right). In areas surrounding the emissive heterogeneous volumes (top two rows), combining our emission and scattering sampling strategies produces a significant improvement over emission sampling alone, while in areas lit by the emissive volume but without as much thin mist (bottom row), MIS ensures that combining our emission and scattering sampling strategies performs no worse than only emission sampling. © 2024 Disney

the technique is worth deploying in production. Furthermore, determinism and temporal coherence are always major production concerns as well. In this section, we discuss both of these topics and present render time and memory measurements from real-world production examples.

5.1.1 Results on Production Scenes. We have chosen three interesting real-world production scenes to showcase performance results. The first scene (Figure 1) is from our short film *Us Again* and was chosen because showcases a combination of complex occlusion and large number of light sources- 4881396 emissive triangles, emissive volumes, and analytical lights in total. The second and third scenes are from our feature-length film *Encanto*. The second scene (Figure 4, containing 38720 light sources) is a case where locally optimal light selection performs relatively well; we chose this scene to demonstrate that our cache points system is still superior in overall performance even in a best-case real-world scenario for locally optimal light selection. The third scene (Figure 5, containing 4406 light sources) demonstrates a case where our learned visibility estimates become important in making cache points robust even with complex occlusion. All three scenes contain some amount of volumetric scattering as well. We carried out out measurements on a system with dual 18-core Intel Xeon Gold 6254 processors; times are presented as elapsed wall-clock time, and all measurements were taken on renders using 32 samples-per-pixel (SPP).

We present measurements of total render time (Table 1), root mean square error (RMSE) (Table 2), and memory usage (Table 4). In order to quantify the performance of all sampling approaches in a single useful number, we present results using *time-to-unit-variance* (TTUV), which is defined as the variance multiplied by total render time (Table 3). This metric represents the time in minutes required to achieve a variance value of 1, which allows us to directly compare the convergence rates of each technique. For all of the metrics we present here, lower values represent better performance.

Across all three of our real-world production examples, cache points shows an absolute advantage over both uniform light selection and locally optimal light selection in time-to-unit-variance, which means that cache points will always reach a given desired noise level faster than any of the other techniques. Even in the scene from Figure 4, where locally optimal light selection performs well, cache points is nearly 1.2x faster in time-to-unit variance, while in the scene from Figure 1, cache points is an order of magnitude faster in both absolute wall-clock time and in time-to-unit variance. When compared with uniform light selection, cache points is anywhere from approximately twice as fast to an order of magnitude faster in time-to-unit variance, which comes from the cache points system's ability to maintain RMSE values close to optimally local light selection while keeping render times close to that of uniform light selection.

Using online learning for visibility estimates typically provides a modest but still useful improvement in time-to-unit-variance; however, when complex occlusion is present (Figure 5), we see that visibility estimates provide a more significant 1.2x speedup in time-to-unit-variance. Furthermore, the tests presented here are relatively low SPP renders, during which the visibility estimate system can only learn a rough approximation of the visibility term; in practice as the visibility estimate improves, time-to-unit-variance further improves as well.

Building the cache points data structure does add some overhead to the renderer's time-to-first-pixel, and storing the cache points data structure requires additional memory overhead; as seen in Table 1, time-to-first-pixel increases by around a minute, and as seen in Table 4, additional memory overhead is typically on the order of a few gigabytes. Relative to typical render times for production frames, an additional minute of startup time is typically

Table 1: Timings for 32 SPP renders using uniform light selection, locally optimal light selection, and cache points. We present timings for both only the cache point initialization process (Build) and for the entire render, inclusive of the initialization process (Total):

Time (32 SPP)	Us Again: 4881396 Lights	Encanto: 38720 Lights	Encanto: 4406 Lights
Uniform Selection	10m 16s	12m 2s	19m 10s
Optimal Selection	123m 51s	16m 32s	24m 21s
Cache Points (Total)	13m 13.1s	15m 0.5s	19m 35s
Cache Points (Build)	1m 32s	1m 4s	34s
Speed vs. Uniform	0.29x Slower	0.37x Slower	0.02x Slower
Speed vs. Optimal	9.37x Faster	1.10x Faster	1.24x Faster

Table 2: Root Mean Square Error (RMSE) measured at 32 SPP using uniform light selection, locally optimal light selection, and cache points. For cache points, we present RMSE both without and with online learning for visibility estimates enabled:

RMSE (32 SPP)	Us Again: 4881396 Lights	Encanto: 38720 Lights	Encanto: 4406 Lights
Uniform Selection	0.2280	0.0896	0.0675
Optimal Selection	0.1436	0.0321	0.0377
Cache Points (No Visibility)	0.1448	0.0339	0.0407
Cache Points	0.1443	0.0320	0.0369

Table 3: Time To Unit Variance (TTUV) measured at 32 SPP using uniform light selection, locally optimal light selection, and cache points. For cache points, we present TTUV both without and with online learning for visibility estimates enabled:

TTUV (32 SPP)	Us Again: 4881396 Lights	Encanto: 38720 Lights	Encanto: 4406 Lights
Uniform Selection	0.5337	0.0966	0.0873
Optimal Selection	2.5552	0.0184	0.0346
Cache Points (No Visibility)	0.2769	0.0172	0.0324
Cache Points	0.2753	0.0154	0.0267

Table 4: Memory usage for uniform light selection and cache points, the difference between the two being the total size of all data structures required for cache points. We do not separately present memory usage for locally optimal light selection, since those values are the same as for uniform light selection:

Memory Usage	Us Again: 4881396 Lights	Encanto: 38720 Lights	Encanto: 4406 Lights
Uniform Selection	33.62 GB	45.0 GB	85.6 GB
Cache Points	37.51 GB	46.17 GB	87.05 GB
% Increase	11.57%	2.6%	1.69%

not significantly concerning. Since we place a hard maximum limit on the number of cache points that the renderer will generate, we can provide a guaranteed, fixed upper bound for the added memory overhead. We have generally found that these added memory overheads are relatively small when compared with the total memory usage of typical production scenes, and, in all but the simplest scenes (such as Cornell Box), cache points provide a sufficiently large advantage in convergence rate to make the added memory overhead worthwhile.

5.1.2 Determinism and Temporal Coherence. Our cache points system is fully deterministic given the same starting random seed and the same input scene. Because the cache points initialization process is highly parallelized, considerable care has to be taken to make sure that determinism is preserved after parallelized steps. This is especially true of the steps where we merge spatially similar candidate locations, merge neighboring cache points with similar light distributions, and blur light distributions between cache points; we have already detailed in the previous sections how sorts between steps and atomic operations are used to avoid race conditions and maintain determinism.

We currently do not take any additional steps with respect to maintaining temporally coherent distributions between frames; we build a new cache point distribution for every frame. There are two reasons for this. Firstly, since the system is already deterministic with respect to random seed, small changes to the input scene generally only produce small changes to the cache point distribution that correspond directly to the changes in the scene. Generally, this property has been enough to maintain consistency across neighboring frames when rendered using the same starting seed, and since cache points produce unbiased direct light sampling, any noise differences become immaterial as the image converges. Secondly, in production, we want to have independent sampling on adjacent frames, so we seed individual frames in each shot with unique values. The reason for this is because our production lighting workflows rely extensively on our in-house denoiser's [Dahlberg et al. 2019; Vogels et al. 2018] advanced cross-frame denoising capabilities, which benefit from reusing and spreading unique samples per frame across multiple frames [Zimmer et al. 2015].

In production use, temporal coherence has not been a significant concern with regards to our cache points system, and we have not

needed to take further measures to avoid related issues. A rare exception case is described in Section 5.2.1.

5.2 Case Studies

Over the course of many productions, we have occasionally run into interesting edge cases that present challenges to the cache points system. We detail some examples here.

5.2.1 Failure Case: Narrow Spot Lights. On *Strange World*, we ran into some issues with using cache points in production scenes that featured very narrow spotlights inside very large atmospheric volumes (Figure 12). Rendering using cache points produced frames with gaps or breaks in the spotlight beams, and these artifacts were also temporally unstable across many frames. As a result, in these types of scenes, artists sometimes fell back to manually disabling the cache points system, which is a rare situation in our productions.

Upon further investigation, we discovered that the issue was simply that we were missing cache points in those particular areas of the spotlights. For the points inside the artifacted areas, the closest cache points were outside the spotlight beam and so did not contain the spotlight in their light distributions; therefore, the probability of sampling the spotlight from the light distribution was zero and instead the spotlights could only be sampled rarely using the low probability we reserve to lights outside of the cache point's light distribution. Since cache points are initially randomly distributed throughout the object bounding boxes, the chances of placing one right within the spotlight cone was rather low, due to how small the spotlight angle is compared to how large the surrounding volume is. We do not take into consideration light positions or directions when seeding the cache points, so there is no guarantee that we will place cache points in the spotlight cone. This explains why the artifacts were temporally unstable across frames: sometimes we would get lucky with the cache point placement, and sometimes we would get unlucky.

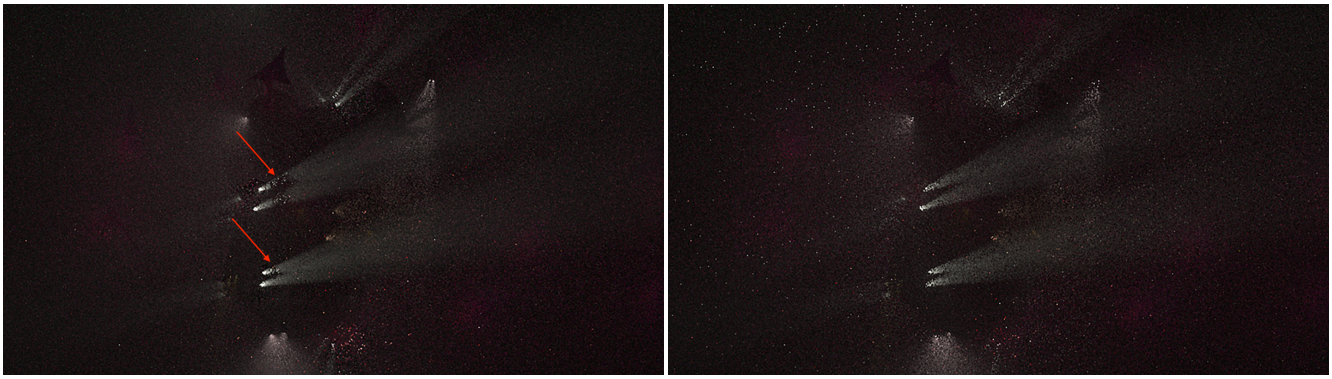
A potential solution to this issue would be to add new cache points at later iterations of the render. We could use ray hits during the later iterations as new cache point candidates and accept them using a modified version of the approach described in Section 3.1.2. We would then want to update the existing cache points' distributions to account for the newly added cache points.

5.2.2 Failure Case: Scaling to Billions of Lights. As mentioned in Section 2, while we do not use a light hierarchy for global many-lights sampling, we do use light hierarchies for selecting individual triangles inside of emissive meshes. The cache points system then treats each emissive mesh as a single light source, as opposed to directly addressing each emissive triangle within the cache point system's light distributions. A major motivating factor for this choice came from an early experiment during *Moana* to place every emissive triangle directly into the cache point system as a uniquely addressable light. While this experiment was not a problem for light sampling at render-time and produced the expected quality with no major impact on the computational workload of light sampling, this experiment did result in extremely long cache point build times at renderer startup. The reason is because we currently simply linearly loop through all lights in the scene when sorting lights into nearby and far distributions and when we determine which lights to place in which cardinal bins in the far distribution. Because we currently carry out these steps in a linear loop, the cache points system's startup time scales poorly as the number of lights in the scene approaches huge numbers, such as billions of lights. An alternative approach would be to use a spatial acceleration structure to winnow down the number of lights that need to be considered when building the light distributions; one possibility is to simply place the centroid of every light into a KD-tree and perform kNN searches, while another possibility is to use a light hierarchy not for many-lights sampling itself, but instead to drive cache point light distribution construction.

5.2.3 Challenges in Volumetric Scattering. Using cache points to improve volumetric in-scattering sampling and volumetric emission sampling has provided a significant improvement in our ability to efficiently render complex and difficult volume setups (Figure 13). As the visual complexity and richness of our films continues to increase, our artists now routinely fill most scenes with some form of thin atmospheric volume to provide additional lighting detail and shaping. Torches or glowing magical effects embedded in thin volumes and dramatic godrays are both commonplace scenarios for us; see Figure 13 for some recent production examples where we made use of cache points for better volume sampling.

However, one major drawback of our current cache points solution for volumetric in-scattering and volumetric emission sampling is that the system can add significantly to render time; each sample per pixel (SPP) is much lower variance but also takes much more time to compute. As a result, while cache points for many-lights sampling are enabled by default, we have not been able to enable cache points for volumetric in-scattering and volumetric emission sampling by default as well. While there are no additional parameters that artists need to set or tune other than simply deciding whether or not to enable this system when rendering volumes, we still would prefer artists to not need to even make the decision to turn the system on or off. Ideally we would either find a way to lower the per-SPP computational workload of this system, or derive a mechanism that would allow the renderer itself to automatically enable or disable the system by automatically detecting at render-time whether or not this system would help in the current scene.

Another minor drawback of our solution for volumes is that this system is piggy-backing off of the cache point locations that already exist for many-lights sampling and doesn't do more to consider additional volumetric parameters when choosing cache point locations. While cache points are placed at locations where paths undergo volumetric scattering, we do not do anything to increase cache point density in optically thin media where large free flight distances mean cache points can be very sparsely distributed, and we do not factor in the volumetric emission field at all when selecting cache point locations. Insufficiently dense cache point coverage for volumetric in-scattering and emission sampling can lead to noise discontinuities similar to the case discussed in Section 5.2.1, while overly dense cache point coverage can lead to unnecessary increased memory usage. For cases like god rays, a potentially useful extension would be to add a third seeding mechanism for cache point locations based on photon tracing from lights. For cases such as volumetric emission, a potentially useful extension would



(a) Scene with cache points enabled **(b) Scene with cache points disabled**

Figure 12: A pared down production scene from *Strange World* with cache points enabled (a) and cache points disabled (b). The main focus in this render is on some spotlights and a large atmospheric volume; all other geometry has been matted out. Note that while the render with cache points enabled generally has less noise than with cache points disabled, there are chunks missing from the two spotlights in the middle (highlighted with red arrows). © 2024 Disney

be to take individual voxel extents and densities into account when placing cache point locations inside the bounding box of a volume.

6 FUTURE WORK

We have now utilized our cache points system in many productions with great success; every production rendered using Disney's Hyperion Renderer to date has done so with the cache points system enabled by default. However, in the spirit of constantly seeking to improve artist workflows, we envision a number of possible improvements to the cache points system.

Interactivity and GPU Implementation. As interactivity, fast artistic iteration times, and optimal time-to-first-pixel becomes increasingly important in all renderer-dependent user workflows, algorithms that require pre-computation time before the renderer can begin tracing rays become less and less attractive; as such, we are interested in progressive formulations of cache points that can work well in interactive use cases. Instead of building the entire cache point data structure upfront, we envision that progressively building the cache point data structure in parallel with the render's first several SPP could be a promising way to speed up time-to-first-pixel while sacrificing only a small amount of noise improvement in the initial set of samples. Also, while our in-house CPU production renderer has used the cache points system for almost the entire history of its existence, our in-house interactive GPU path tracer currently uses a combination of ReSTIR [Bitterli et al. 2020] and a light hierarchy without occlusion estimates [Estevez and Kulla 2018] for light selection; ideally we would prefer to have the same light selection strategy across both renderers. To this end, reformulating the cache points system to work well on the GPU for both build and sampling is a major point of interest for us. We would also like to experiment with using cache points as the initial light selection method used to drive ReSTIR.

Alternate Spatial Data Structures. Our cache points system currently stores points in a KD-tree; we are interested in investigating better spatial data structures that can either reduce memory footprint, improve data structure build times, or improve point search times. While KD-trees are likely to remain part of the initial cache points build process due to the need to perform multiple kNN search operations, at actual path tracing time we only need to look up a singular cache point per path vertex, making spatial data structures such as hash grids for storing the final render-time cache point distribution a promising alternative. For a GPU implementation, a multi-resolution hash grid is especially appealing over a KD-tree [Davidovič et al. 2014].

Improved Sampling Information. Our system currently does not utilize joint sampling to take into account the BSDF when performing light selection [Christensen et al. 2018]; adding this capability could potentially help in eliminating the need to sample large numbers of lights that do not line up with highly glossy BSDFs. In real-world production scenes, we expect that using a joint sampling approach in conjunction with our occlusion estimate approach could lead to significant sampling efficiency improvements. In a similar vein: we currently do not consider surface orientation when merging spatially neighboring cache points during the build process; a potential solution could be to factor in a surface normal for cache points placed on surfaces and allow otherwise nearby cache points to not be merged if they have highly divergent corresponding surface normals.

Combining with Path Guiding. Currently we use two separate systems for guiding direct lighting and guiding indirect lighting; for indirect lighting, we use Practical Path Guiding [Müller 2019; Müller et al. 2017]. Unifying systems for guiding direct and indirect lighting is a worthy goal; to this end, we have investigated combining Practical Path Guiding and the technique from Vevoda et al. [2018] with promising results.

Combining with Photon Mapping. Hyperion includes a photon mapping system for rendering refractive caustics. Since this photon mapping system was implemented after the core of the cache

Figure 13: Production frames *Raya and the Last Dragon* utilizing our cache points system for sampling volumetric in-scattering and volumetric emission. © 2024 Disney

points system was implemented, our cache points system currently does not leverage any of the capabilities of the photon mapping system, but these two systems do seem to have complimentary capabilities. For example, using photons from forward light tracing to generate candidate cache point locations is a natural extension of our existing approach and could help with cases like the one in Section 5.2.1. Our photon mapping system uses an adaptive photon guiding technique that learns photon emission functions across light sources throughout the duration of a render, similar to Estevez and Kulla [2020a; 2020b]; combining this adaptive photon guiding with our learned occlusion estimates in cache points seems promising. Finally, our cache point locations are chosen up-front and not further refined during throughout the course of the render; using a mechanism similar to the ones found in PPM [Hachisuka et al. 2008] and SPPM [Hachisuka and Jensen 2009] to progressively refine cache point locations is a possible approach.

7 CONCLUSION

We have presented Cache Points, a many-lights sampling system used by Disney's Hyperion Renderer to render millions of production frames over the past decade. We have described in detail how cache points are populated and novel aspects of the system such as online learning of visibility estimates, how they are used in light sampling at render-time, and how we have extended them for use in accelerating difficult volumetric scattering cases. We have also discussed some production experiences, real-world success cases, and real-world failure cases for our system, along with potential paths for future improvement.

ACKNOWLEDGMENTS

The techniques presented in this paper have seen continual improvement over the years, with many contributions made by both current and past members of the Hyperion development team. In addition to the authors, other past key contributors to the cache points system include Patrick Kelly, Ralf Habel, Ben Spencer, Benedikt Bitterli, and Matt Jen-Yuan Chiang. The authors are also thankful to Mackenzie Thompson, Andrew Bauer, Brian Green, Mark Lee, and Lea Reichardt from the Hyperion development team for their support of and feedback on this paper.

We thank Jan Novák, Marios Papas and Thomas Müller from Disney Research|Studios and Cliff Ramshaw and Julian Fong from Pixar's RenderMan development team for interesting and helpful discussions on the topics of many-lights sampling and volumetric scattering through optically thin media. We also thank Ivo Kondapaneni for his work implementing Vevoda et al. [2018]'s technique in an experimental branch of Hyperion, which has served as a useful comparison point. We are also thankful to our anonymous paper referees for their invaluable feedback.

Finally, we are especially grateful to the many artists and technical directors that have used Hyperion and whose feedback, suggestions, and partnership over the years have influenced and shaped every aspect of the renderer, including the cache points system presented in this paper.

REFERENCES

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal Reservoir Sampling for Real-Time Ray Tracing with Dynamic Direct Lighting. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 39, 4, Article 148 (July 2020). https://doi.org/10.1145/3386569.3392481

Jakub Boksansky, Paula Jukarainen, and Chris Wyman. 2021. Rendering Many Lights with Grid-Based Reservoirs. *Ray Tracing Gems II* (Aug. 2021). https://doi.org/10.1007/978-1-4842-7185-8_23

Brent Burley, David Adler, Matt Jen-Yuan Chiang, Hank Driskill, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Dan Teece. 2018. The Design and Evolution of Disney's Hyperion Renderer. *ACM Transactions on Graphics* 37, 3, Article 33 (Aug. 2018). https://doi.org/10.1145/3182159

Brent Burley, David Adler, Matt Jen-Yuan Chiang, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Dan Teece. 2017. Recent Advances in Disney's Hyperion Renderer. *Path Tracing in Production Part 1, SIGGRAPH 2017 Course Notes*, Article 13 (July 2017), 26-34 pages. https://doi.org/10.1145/3084873.3084904

Min-Te Chao. 1982. A General Purpose Unequal Probability Sampling Plan. *Biometrika* 69, 3 (Dec. 1982), 653–656. https://doi.org/10.1093/biomet/69.3.653

Matt Jen-Yuan Chiang, Benedikt Bitterli, Chuck Tappan, and Brent Burley. 2016. A Practical and Controllable Hair and Fur Model for Production Path Tracing. *Computer Graphics Forum (Proc. of Eurographics)* 35, 2 (May 2016), 275–283. https://doi.org/10.1111/cgf.12830

Per Christensen, Julian Fong, Jonathan Shader, Wayne Wooten, Brenden Schubert, Andrew Kensler, Stephen Friedman, Charlie Kilpatrick, Cliff Ramshaw, Marc Bannister, Brenton Rayner, Jonathan Brouillat, and Max Liani. 2018. RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering. *ACM Transactions on Graphics* 37, 3 (Aug. 2018), 30. https://doi.org/10.1145/3182162

John Horton Conway and Neil James Alexander Sloane. 1999. *Sphere Packings, Lattices, and Groups* (1st ed.). Springer.

Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. 2014. Scalable Realistic Rendering with Many-Light Methods. *Computer Graphics Forum (Proc. of Eurographics)* 33, 1 (Feb. 2014). https://doi.org/10.1111/cgf.12256

Henrik Dahlberg, David Adler, and Jeremy Newlin. 2019. Machine-Learning Denoising in Feature Film Production. In *ACM SIGGRAPH 2019 Talks (SIGGRAPH '19)*. Article 21, 2 pages. https://doi.org/10.1145/3306307.3328150

Weijing Dai, Joerg Reimann, Dorian Hanaor, Claudio Ferrero, and Yixiang Gan. 2019. Modes of Wall Induced Granular Crystallisation in Vibrational Packing. *Granular Matter* 21, 2, Article 26 (June 2019). https://doi.org/10.1007/s10035-019-0876-8

Tomáš Davidovič, Iliyan Georgiev, and Philipp Slusallek. 2012. Progressive Lightcuts for GPU. In *ACM SIGGRAPH 2012 Talks (SIGGRAPH '12)*. Article 1, 2 pages. https://doi.org/10.1145/2343045.2343047

Tomáš Davidovič, Jaroslav Křivánek, Miloš Hašan, and Philipp Slusallek. 2014. Progressive Light Transport Simulation on the GPU: Survey and Improvements. *ACM Transactions on Graphics* 33, 3, Article 29 (May 2014). https://doi.org/10.1145/2602144

Addis Dittebrandt, Vincent Schüßler, Johannes Hanika, Sebastian Herholz, and Carsten Dachsbacher. 2023. Markov Chain Mixture Models for Real-Time Direct Illumination. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)* 42, 4, Article e14881 (June 2023). https://doi.org/10.1111/cgf.14881

Christian Eisenacher, Gregory Nichols, Andrew Selle, and Brent Burley. 2013. Sorted Deferred Shading for Production Path Tracing. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)* 32, 4 (July 2013), 125–132. https://doi.org/10.1111/cgf.12158

Alejandro Conty Estevez and Christopher Kulla. 2018. Importance Sampling of Many Lights with Adaptive Tree Splitting. *Proc. of the ACM on Computer Graphics and Interactive Techniques (Proc. of HPG)* 1, 2, Article 25 (Aug. 2018). https://doi.org/110.1145/3233305

Alejandro Conty Estevez and Christopher Kulla. 2020a. Adaptive Caustic Rendering in Production with Photon Guiding. In *Proc. of EGSR*. 67–78.

Alejandro Conty Estevez and Christopher Kulla. 2020b. Practical Caustics Rendering with Adaptive Photon Guiding. In *ACM SIGGRAPH 2020 Talks (SIGGRAPH '20)*. Article 17, 2 pages. https://doi.org/10.1145/3388767.3407370

Luca Fascione, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomáš Davidovič, Andrea Weidlich, and Johannes Meng. 2018. Manuka: A Batch-Shading Architecture for Spectral Path Tracing in Movie Production. *ACM Transactions on Graphics* 37, 3, Article 31 (Aug. 2018). https://doi.org/110.1145/3182161

Julian Fong, Magnus Wrenninge, Christopher Kulla, and Ralf Habel. 2017. Production Volume Rendering. In *ACM SIGGRAPH 2017 Courses* (Los Angeles, CA, USA) *(SIGGRAPH 2017)*. ACM, New York, NY, USA, Article 2, 2:1–2:79 pages. https://doi.org/10.1145/3084873.3084907

Blender Foundation. 2024. Light Objects - Blender Manual. (2024). https://docs.blender.org/manual/sl/3.6/render/lights/light_object.html#area-light

Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramón Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, Adrien Herubel, Declan Russell, Frédéric Servant, and Marcos Fajardo. 2018. Arnold: A Brute-Force Production Path Tracer. *ACM Transactions on Graphics* 37, 3, Article 32 (Aug. 2018). https://doi.org/110.1145/3182160

Iliyan Georgiev, Jaroslav Křivánek, Stefan Popov, and Philipp Slusallek. 2012. Importance Caching for Complex Illumination. *Computer Graphics Forum (Proc. of Eurographics)* 31, 3 (May 2012), 701–710. https://doi.org/10.1111/j.1467-8659.2012.03049.x

Petra Gospodnetić. 2017. GSoC 2017 with AppleseedHQ. https://medium.com/@petragospodneti/gsoc-2017-with-appleseedhq-cc5f33d04170

Jerry Jinfeng Guo, Pablo Bauszat, Jacco Bikker, and Elmar Eisemann. 2018. Primary Sample Space Path Guiding. In *Experimental Ideas and Implementations (EGSR 2018: Eurographics Symposium on Rendering)*. 73–82. https://doi.org/10.2312/sre.20181174

Toshiya Hachisuka and Henrik Wann Jensen. 2009. Stochastic Progressive Photon Mapping. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* 28, 5, Article 141 (Dec. 2009). https://doi.org/10.1145/1618452.1618487

Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. 2008. Progressive Photon Mapping. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* 27, 5, Article 130 (Dec. 2008). https://doi.org/10.1145/1409060.1409083

Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, Quang Truong Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, Thi Hoai An Ta, Nam Trung Tran, Thi Diep Trieu, Josef Urban, Ky Bu, and Roland Zumkeller. 2017. A Formal Proof of the Kepler Conjecture. *Forum of Mathematics, Pi* 5, Article e2 (May 2017). https://doi.org/10.1017/fmp.2017.1

Sebastian Herholz, Oscar Elek, Jiří Vorba, Hendrik P.A. Lensch, and Jaroslav Křivánek. 2016. Product Importance Sampling for Light Transport Path Guiding. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)* 35, 4 (July 2016), 67–77. https://doi.org/10.1111/cgf.12950

Sebastian Herholz, Yangyang Zhao, Oskar Elek, Derek Nowrouzezahrai, Hendrik P.A. Lensch, and Jaroslav Křivánek. 2019. Volume Path Guiding Based on Zero-Variance Random Walk Theory. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 38, 3, Article 25 (June 2019). https://doi.org/10.1145/3230635

Wei-Feng Wayne Huang, Peter Kutz, Yining Karl Li, and Matt Jen-Yuan Chiang. 2021. Unbiased Emission and Scattering Importance Sampling For Heterogeneous Volumes. In *ACM SIGGRAPH 2021 Talks (SIGGRAPH '21)*. Article 3, 2 pages. https://doi.org/10.1145/3450623.3464644

Henrik Wann Jensen. 1995. Importance Driven Path Tracing using the Photon Map. In *Proc. of Eurographics Workshop on Rendering Techniques*. 326–335. https://doi.org/10.1007/978-3-7091-9430-0_31

Henrik Wann Jensen. 1996. Global Illumination using Photon Maps. In *Proc. of Eurographics Workshop on Rendering Techniques*. 21–30. https://doi.org/10.1007/978-3-7091-7484-5_3

Henrik Wann Jensen. 2001. *Realistic Image Synthesis Using Photon Mapping* (1st ed.). A.K. Peters.

Alexander Keller, Carsten Wächter, Matthias Raab, Daniel Seibert, Dietger van Antwerpen, Johann Korndörder, and Lutz Kettner. 2017. The Iray Light Transport Simulation and Rendering System. (May 2017). https://arxiv.org/abs/1705.01263

Chrisopher Kulla, Alejandro Conty Estevez, Clifford Stein, and Larry Gritz. 2018. Sony Pictures Imageworks Arnold. *ACM Transactions on Graphics* 37, 3, Article 29 (Aug. 2018). https://doi.org/110.1145/3180495

Christopher Kulla and Marcos Fajardo. 2012. Importance Sampling Techniques for Path Tracing in Participating Media. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)* 31, 4 (June 2012), 1519–1528. https://doi.org/10.1111/j.1467-8659.2012.03148.x

Peter Kutz, Ralf Habel, Yining Karl Li, and Jan Novák. 2017. Spectral and Decomposition Tracking for Rendering Heterogeneous Volumes. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 36, 4 (July 2017), 111:1–111:16. https://doi.org/10.1145/3072959.3073665

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized Resampled Importance Sampling: Foundations of ReSTIR. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 41, 4, Article 75 (July 2022). https://doi.org/10.1145/3528223.3530158

Bailey Miller, Iliyan Georgiev, and Wojciech Jarosz. 2019. A Null-Scattering Path Integral Formulation of Light Transport. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 38, 4, Article 44 (July 2019). https://doi.org/10.1145/3306346.3323025

Thomas Müller. 2019. Practical Path Guiding in Production. *Path Guiding in Production, SIGGRAPH 2019 Course Notes*, Article 18 (July 2019), 37-49 pages. https://doi.org/10.1145/3305366.3328091

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)* 36, 4 (June 2017), 91–100. https://doi.org/10.1111/cgf.13227

Christopher Nichols. 2016. Understanding Vray Adaptive Lights. (Nov. 2016). https://www.chaos.com/blog/understanding-adaptive-lights

Gregory Nichols and Christian Eisenacher. 2015. Hyperion, Disney's New Global Illumination Renderer. *The Path Tracing Revolution in the Movie Industry, SIGGRAPH 2015 Course Notes* (July 2015). https://doi.org/10.1145/2776880.2792699

Jan Novák, Andrew Selle, and Wojciech Jarosz. 2014. Residual Ratio Tracking for Estimating Attenuation in Participating Media. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* 33, 6, Article 179 (Nov. 2014), 179:1–179:11 pages. https://doi.org/10.1145/2661229.2661292

Jacopo Pantaleoni. 2019. Importance Sampling of Many Lights with Reinforcement Lightcuts Learning. (Nov. 2019). https://arxiv.org/abs/1911.10217

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2023. *Physically Based Rendering: From Theory to Implementation* (4th ed.). MIT Press.

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Petr Vévoda, Philipp Slusallek, and Jaroslav Křivánek. 2020. Variance-Aware Path Guiding. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 39, 4, Article 151 (July 2020). https://doi.org/10.1145/3386569.3392441

Lukas Ruppert, Sebastian Herholz, and Hendrik P. A. Lensch. 2020. Robust Fitting of Parallax-Aware Mixtures for Path Guiding. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 39, 4, Article 147 (July 2020). https://doi.org/10.1145/3386569.3392421

Peter Shirley, Changyaw Wang, and Kurt Zimmerman. 1996. Monte Carlo Techniques for Direct Lighting Calculations. *ACM Transactions on Graphics* 15, 1 (Jan. 1996), 1–36. https://doi.org/10.1145/226150.226151

Florian Simon, Johannes Hanika, Tobias Zirr, and Carsten Dachsbacher. 2017. Line Integration for Rendering Heterogeneous Emissive Volumes. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)* 36, 4 (July 2017), 101–110. https://doi.org/10.1111/cgf.13228

Justin Talbot, David Cline, and Parris Egbert. 2005. Importance Resampling for Global Illumination. In *Proc. of EGSR*. 139–146. https://doi.org/10.2312/EGWR/EGSR05/139-146

Yusuke Tokuyoshi. 2021. Tiled Reservoir Sampling for Many-Light Rendering. (Nov. 2021). https://gpuopen.com/download/publications/TiledReservoirSampling21-11-ecdc.pdf

Vaibhav Vavilala. 2019. Light Pruning on Toy Story 4. In *ACM SIGGRAPH 2019 Talks (SIGGRAPH '19)*. Article 44, 2 pages. https://doi.org/10.1145/3306307.3328161

Eric Veach. 1998. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph. D. Dissertation. Stanford, CA, USA. Advisor(s) Guibas, Leonidas J.

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian Online Regression for Adaptive Direct Illumination Sampling. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 37, 4, Article 125 (Aug. 2018). https://doi.org/10.1145/3197517.3201340

Petr Vévoda and Jaroslav Křivánek. 2016. Adaptive Direct Illumination Sampling. In *ACM SIGGRAPH Asia 2016 Posters (SIGGRAPH Asia '16)*. Article 43, 2 pages. https://doi.org/10.1145/3005274.3005283

Ryusuke Villemin and Christophe Hery. 2013. Practical Illumination from Flames. *Journal of Computer Graphics Techniques* 2, 2 (Dec. 2013), 142–155.

Thijs Vogels, Fabrice Rouselle, Brian McWilliams, Gerhard Rothlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. 2018. Denoising with Kernel Prediction and Asymmetric Loss Functions. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 37, 4, Article 124 (Aug. 2018). https://doi.org/10.1145/3197517.3201388

Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 2020. Path Guiding in Production. In *ACM SIGGRAPH 2020 Courses (SIGGRAPH '20)*. Article 18. https://doi.org/10.1145/3305366.3328091

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 33, 4, Article 101 (aug 2014), 101:1–101:11 pages. https://doi.org/10.1145/2601097.2601203

Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. 2005. Lightcuts: A Scalable Approach to Illumination. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* 24, 3 (July 2005), 1098–1107. https://doi.org/10.1145/1073204.1073318

Gregory J Ward. 1991. Adaptive Shadow Testing for Ray Tracing. In *Proc. of Eurographics Workshop on Rendering Techniques*. 11–20. https://doi.org/10.1007/978-3-642-57963-9_2

Chris Wyman, Markus Kettunen, Daqi Lin, Benedikt Bitterli, Cem Yuksel, Wojciech Jarosz, and Pawel Kozlowski. 2023. A Gentle Introduction to ReSTIR: Path Reuse in Real-Time. In *ACM SIGGRAPH 2023 Courses* (Los Angeles, CA, USA) *(SIGGRAPH 2023)*. ACM, New York, NY, USA, Article 1, 1:1–1:61 pages. https://doi.org/10.1145/3587423.3595511

Henning Zimmer, Fabrice Rouselle, Wenzel Jakob, Oliver Wang, David Adler, Wojciech Jarosz, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung. 2015. Path-space Motion Estimation and Decomposition for Robust Animation Filtering. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)* 34, 4 (June 2015), 131–142. https://doi.org/10.1111/cgf.12685