Image Synthesis using Adjoint Photons

R. Keith Morley
Princeton University

Solomon Boulos
University of Utah

Jared Johnson
University of Central Florida

David Edwards
University of Utah

Peter Shirley
University of Utah

Michael Ashikhmin
SUNY Stony Brook

Simon Premože
ILM

Figure 1: An image generated using adjoint photon tracing in a scene with millions of instanced polygons, a heterogeneous medium from simulation data, and Rayleigh scattering for the sky. The Sun is the only source of light in the scene (i.e. there is no background radiance or sky map) and uses a spectral radiance emission function.

ABSTRACT

The most straightforward image synthesis algorithm is to follow photon-like particles from luminaires through the environment. These particles scatter or are absorbed when they interact with a surface or a volume. They contribute to the image if and when they strike a sensor. Such an algorithm implicitly solves the light transport equation. Alternatively, adjoint photons can be traced from the sensor to the luminaires to produce the same image. This "adjoint photon" tracing algorithm is described, and its strengths and weaknesses are discussed, as well as details needed to make adjoint photon tracing practical.

Keywords: global illumination, adjoint Monte Carlo methods

1 INTRODUCTION

In this paper we discuss tracing adjoint photons to create images. This method of image synthesis has some theoretical and practical advantages over path tracing [9]; it is simpler to motivate because less of the machinery of radiometry and Monte Carlo integration are needed, and lacking ray integration and explicit direct lighting it deals better with some "difficult" scenes. Our paper develops some of the details of adjoint photon tracing, and discusses where it needs improvement.

Our motivation for this work is three-fold. First, we have found that for predictive engineering applications (e.g. accurately simulating a car headlight modeled as a glass case over a curved, specular reflector in a fog bank), both the geometric and optical complexity can overwhelm most renderers. Second, as complex renderers adopt layers of tricks, approximations, and hacks, both their design elegance and robustness suffer. Finally, we note that in a world of interactive ray tracing, a batch renderer can send trillions of rays in an overnight run. Such a budget obviates the need for many approximations in applications that benefit from higher accuracy. All three of these motivations imply that we should explore simple algorithms with high generality and straightforward implementation.

The particular algorithm we have choosen is to send photons from the sensor (pixel) into the environment and note how likely this is to hit the light source. This is like path tracing in that it sends rays from the sensor into the environment and lets them choose random directions, but it has three important differences from path tracing all of which are advantages in complex scenes where predictive results are desired:

- indirect and direct lighting are not separated so there are no explicit shadow rays making luminaires with reflectors not any more difficult than diffusely emitting surfaces;

- all adjoint photons have exactly one wavelength rather than carrying a spectrum, so bidirectional reflection distribution functions (BRDFs) and phase functions that vary in shape with wavelength present no particular difficulties;

- volumes are probabilistically either hit or not hit by adjoint photons so their transparency is accounted for by the percentage of adjoint photons that hit rather than by accumulation of opacity for each adjoint photon, resulting in a straightforward implementation well-suited to problems requiring many samples per pixel.
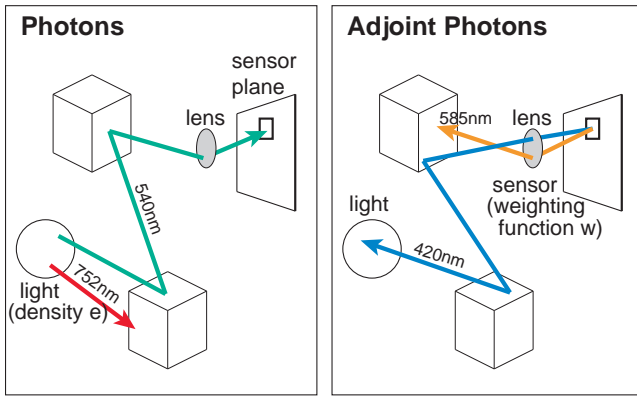
Figure 2: The light emits spectral radiance e and the sensor weights incident light with function w. Left: Photons are traced from the light with cosine-weighted $e(\vec{x},\omega,\lambda,t)$ as the emitted radiance and only recorded with $w(\vec{x},\omega,\lambda,t)$ when they hit the sensor. Right: Adjoint photons are traced from the sensor with cosine-weighted $w(\vec{x},\omega,\lambda,t)$ as the emitted radiance and only recorded with $e(\vec{x},\omega,\lambda,t)$ when they hit the light. These processes both result in the same accumulated response with no need for conversion factors between units. In this example, the adjoint photon case is equivalent to using a planar light source behind the lens and a spherical camera.

2   ADJOINT LIGHT TRANSPORT

While the self-adjoint nature of light transport has been exploited in graphics for over a decade [5, 13, 18], it has a longer history in other transport problems as surveyed in the excellent articles by Arvo [1] and Christensen [4]. However, a physical implication of this self-adjoint property is not as well known: the emission of a light source and the response of a sensor can be exchanged without changing the throughput along a path. This property has been exploited in graphics for the first time in *dual photography* [16] but also makes the design of a rendering algorithm straightforward.

This exchange of source emission and sensor response does not just mean that some kind of "importance" can be emitted from the light, but that a complete exchange can take place without any normalization. That is, suppose we have a luminaire whose emitted spectral radiance is given by a function $e(\vec{x},\omega,\lambda,t)$, where $\vec{x}$ is a 3D location, ω is the emitted direction of light, λ is wavelength and t is time. Further suppose we have a sensor which responds to incident spectral radiance L with the integral:

$$R = \int w(\vec{x},\omega,\lambda,t)L(\vec{x},\omega,\lambda,t)\cos\theta\, dA\, d\Omega\, d\lambda\, dt$$

where R is the response of the sensor and w is the weighting function for the sensor. Note that R depends on both e and the properties of the environment. In a typical environment, e is non-zero on any luminaire surface during whatever time period it is "on". The sensor weight w is non-zero on the sensor element (e.g. a CCD pixel) during the time the sensor is active.

If we were to make the sensor emit light with spectral radiance $w(\vec{x},\omega,\lambda,t)$ and make the luminaire a sensor with weighting function $e(\vec{x},\omega,\lambda,t)$ the resulting response is:

$$R_A = \int e(\vec{x},\omega,\lambda,t)L(\vec{x},\omega,\lambda,t)\cos\theta\, dA\, d\Omega\, d\lambda\, dt.$$

The key property that reversibility implies is that this response is exactly the same as the original case:

$$R_A = R.$$

An immediate implication of this is that any rendering program can be "run in reverse" by emitting light from sensors using the sensor response as an emitted radiance distribution, and sensing that light at luminaires using their radiance distribution as sensitivity. This requires no additional normalization or complicated equations. We exploit this in the next section using photon tracing.

3   ADJOINT PHOTON TRACING

The simplest algorithm is a brute-force simulation of photon-like particles as illustrated in the left of Figure 2. This differs from the family of photon-tracing algorithms that store photon-surface and photon-volume intersections [7] in that nothing is logged but interactions with the sensor array. The tracing of photons (as opposed to how they are logged) is the same as that of Walter et al. [23] where photons have individual wavelengths so wavelength-dependent scattering is straightforward. The algorithm to compute the sensor response R is:

$R = 0$
$Q = \int e(\vec{x},\omega,\lambda,t)\cos\theta\, dA\, d\Omega\, d\lambda\, dt$
for each of N photons **do**
    choose a random $(\vec{x},\omega,\lambda,t) \sim e(\vec{x},\omega,\lambda,t)\cos\theta/Q$
    while photon is not absorbed **do**
        find point $\vec{x}'$ hit by ray
        $R = R + (Q/N)w(\vec{x}',\omega,\lambda,t)$
        $H = \int \rho(\vec{x}',\omega',\omega,\lambda,t)\cos\theta'\, dA'\, d\Omega'\, d\lambda\, dt$
        if random $< H$ **then**
            choose random $\omega' \sim \rho(\vec{x}',\omega',\omega,\lambda,t)\cos\theta'/H$
        else
            absorb photon

Here ρ is the BRDF, e is the emitted spectral radiance as defined in the last section, and w is the sensor response function as defined in the last section. The line $R = R + Qw(\vec{x},\omega,\lambda,t)$ has no cosine term because it is included implicitly by the change in projected area of the sensor which varies linearly with cosine. Note that both e and w are defined over all surfaces, but w is non-zero only on the sensor and e is non-zero only on the luminaires. Thus no photons are generated except at luminaires, and only photons hitting a sensor affect the value of the sensor value R. The constants H and Q simply serve as normalization factors to ensure that e and ρ are valid probability density functions (pdfs). Note that the value R is a scalar, just as for one element in a typical CCD camera. For many pixels or multiple colors there need to be more sensors (e.g. three for each pixel, one red, one green and one blue). In the case of an open environment, the ray will also terminate when it leaves the environment.

Because the transport equation is self-adjoint [13, 18], or can be made so [22], the photon tracing algorithm can be run in reverse with "adjoint photons" as shown in the left of Figure 2. Thus reversing the photon tracing algorithm above involves just one change: **the function w should be exchanged with the function e**. Adjoint photons are emitted from sensors using w as the emission function and are "sensed" by luminaires using e as the "sensor" weighting function.

Operationally, an adjoint photon tracer behaves identically to a traditional photon tracer. An implication of this is that adjoint photon tracing can be implemented by tricking a traditional photon tracer to use w as a spectral emission function and e as the sensor weighting function. We are using the fact that the physically based scattering is self-adjoint to avoid explicitly using the underlying adjoint transport equations. The same principle has been used for decades in other radiation transport problems [10]. We emphasize that no normalization is required when swapping the emission and sensor response functions.
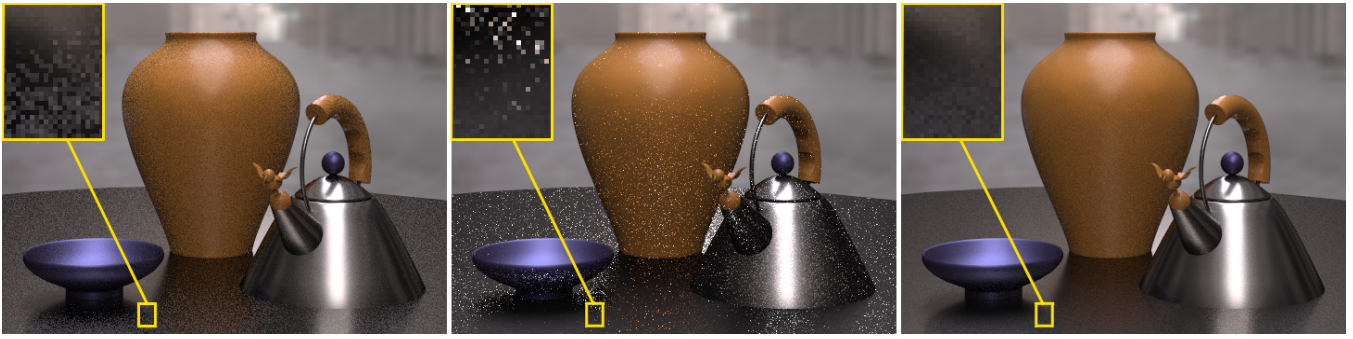
Figure 3: Using a mixture density. Left: Sampling the BRDF. Middle: Sampling the environment map. Right: Mixture density with weighting coefficients 0.75 for the BRDF and 0.25 for the environment map. All images use 289 samples per pixel and have the same run times.

4 IMPORTANCE SAMPLING

Although adjoint photon tracing is clean, it is very noisy in the presence of small bright sources (e.g. the Sun). This can be improved by all the standard techniques of Monte Carlo particle transport [2, 10]. The most critical of these is importance sampling, where instead of using the density $p'(\omega') = \rho(\omega, \omega') \cos\theta'/H$ to choose the direction ω' to scatter the particle at a surface, we use some other density $q(\omega')$. When we do this, some directions will get "too much" energy and some will get "too little" and a correction factor $p'(\omega')/q(\omega')$ is needed to appropriately scale the energy. We use a $q(\omega')$ designed to emphasize directions we expect to be bright. The algorithm proceeds as follows:

$R_A = 0$
$Q = \int w(\vec{x}, \omega, \lambda, t) \cos\theta \, dA \, d\Omega \, d\lambda \, dt$
for each of N adjoint photons **do**
    choose a random $(\vec{x}, \omega, \lambda, t) \sim w(\vec{x}, \omega, \lambda, t) \cos\theta / Q$
    $E = Q/N$
    while photon is not absorbed **do**
        find surface hit by ray
        $R_A = R_A + E e(\vec{x}', \omega, \lambda, t)$
        $H = \int \rho(\vec{x}', \omega', \omega, \lambda, t) \cos\theta' \, dA' \, d\Omega' \, d\lambda \, dt$
        if random $< Hh$ **then**
            choose a random direction $\omega' \sim q$
            $E = (1/h)E\rho(\vec{x}', \omega', \omega, \lambda, t) \cos\theta'/(Hq(\omega'))$
        else
            absorb photon

Here E is the energy of the adjoint photon which is initialized to be the total energy divided by the number of adjoint photons. Due to importance sampling, the energy is rescaled. h is a user-set variable that controls the Russian-roulette of photons [2] and can be set to one for simple operation. In practice we usually use $h = 1$ along with a maximum photon depth.

The key to improving the efficiency of our adjoint photon tracer is to choose a "good" density function p. Most path tracers choose to partition illumination into two separate integrals corresponding to "direct" and "indirect" lighting [17]. One problem with such a separation is that it is not clear where high dynamic range (HDR) environment maps or bright caustics fit into such a partition. We instead only sample directions without the explicit concept of direct lighting.

In practice we use a weighted average of simple probability density functions (pdfs). This averaging of density functions is a "mixture density" where a new pdf is constructed by a weighted average of N pdfs p_1 through p_N [10]. In practice, one p_i is related to the BRDF, one describes directions toward luminaires, and one is proportional to the background luminance. When the background is an HDR environment map, the p_i related to the background can be sampled according to intensity weighted solid angle. Note that we
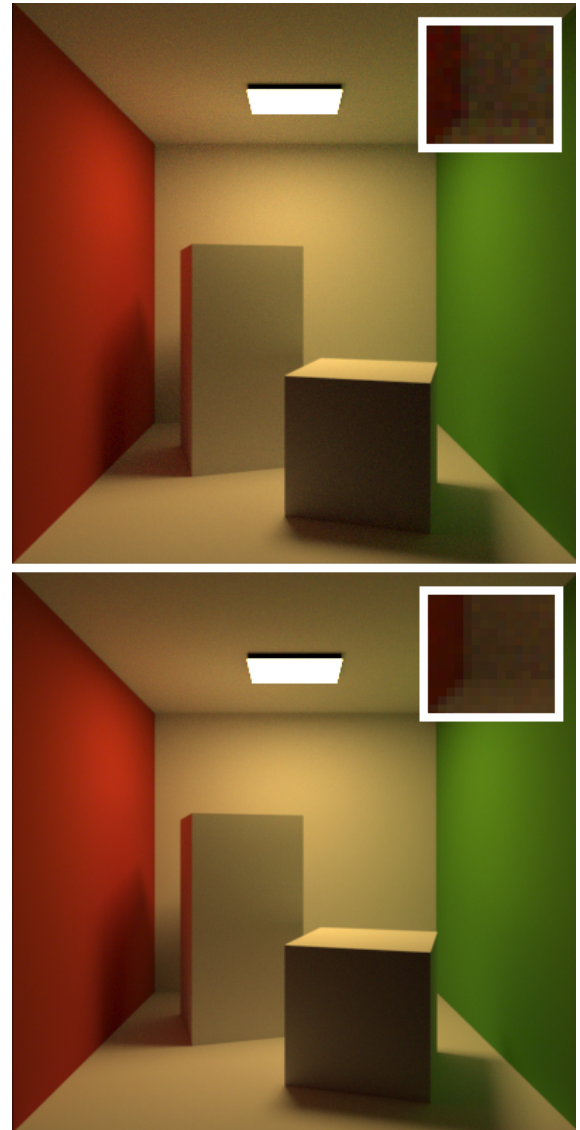

Figure 4: Top: Sampling using a mixture density averaging cosine weighting and directions toward the light. Bottom: Sampling using multiple importance sampling with the power heuristic. Both images were computed in approximately the same amount of time. The insets are magnifications of the pixels near the back left corner.

need to evaluate the values of every p_i regardless of which distribution the sample is taken from. This matters for cases where the p_i have overlapping non-zero regions (e.g. sampling the BRDF and the environment map).

This mixture density architecture can be extended recursively. The density for BRDFs can be a mixture density of a diffuse and a specular lobe. The density for area lights can weight nearby lights more heavily. A density over known caustics could weight directions according to an approximation of subtended solid angle. This emphasizes that any improvement in the weighting coefficients, even through approximations, will be beneficial in improving efficiency.

An example of the power of this mixture density importance sampling technique is shown in Figure 3. Here we use brute force importance sampling of the environment map and the BRDFs, using models and BRDF data from Lawrence et al. [12]. In addition to dealing with sampling problems, the directional sampling method produce a very clean code architecture.

While using mixture densities is robust, it helps little over traditional path tracing for simple scenes. Figure 4 (left) shows an image with depth 6 and a mixture density sampling function that is the average of cosine density and a density that approximates uniform sampling toward the light, using 10,000 paths per pixel. 10,000 paths per pixel is approximately ten times the paths needed for the same level of convergence as for a conventional path tracer in our experience (and approximately five times the total ray segments because no explicit shadow rays are used). So adjoint photon tracing is not a good choice for simple diffuse scenes. However, this method pays off in the presence of complex lights or participating media.

In addition to changing the density we can split adjoint photons as long as the expected weights add to one. This allows not only traditional splitting [10], but also any of Veach's multiple importance sampling (MIS) techniques to be integrated into adjoint photon tracing [21]. The big difference is that a sample (and thus a ray) is needed for each density combined with MIS, producing a branching factor above one. This is shown for the power heuristic with exponent 2 in the right of Figure 4 for a similar runtime to the left figure (using about half as many screen samples to make up for the extra branching). As can be seen, if there is an advantage over the branchless case, it is slight. In practice we usually avoid splitting for the sake of simplicity, but some MIS splits early in the ray tree does improve convergence for some scenes.

For traditional path tracers that use direct lighting, samples are generated on the surface of a luminaire in order to query properties of the light source. Many direct lighting computations require a direction and distance for a shadow ray as well as an estimate of the emitted radiance from the light source if the point is unoccluded [17]. While choosing a point towards a complex reflector would be fairly straightforward, it is not clear how one would cleanly estimate the emitted radiance when the "light" is unoccluded. Our method instead considers important directions such as those towards light sources or a complex reflector (Figure 5) and avoids determining whether or not something is occluded.

As another simple example of this distinction, consider a glass cover over a light source. For direct lighting calculations, the glass cover occludes the light source and the computation produces a shadow. In a system without direct lighting, we will simply send samples towards the light source more frequently and would produce the appropriate image.
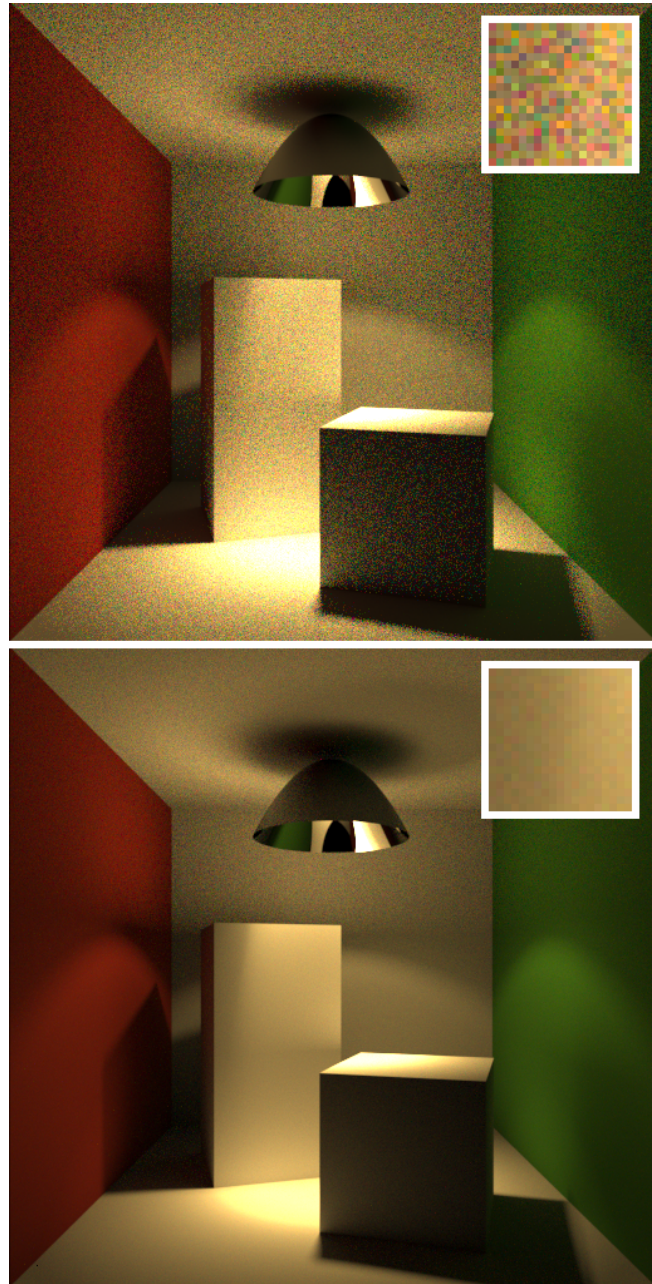

Figure 5: Top: A small spherical light inside a metal parabolic reflector, where the sphere itself is sampled, along with a magnification of the front of the taller box. Bottom: The opening of the parabolic reflector is sampled instead of the sphere. The same number of samples are used in each and the runtimes are approximately the same.
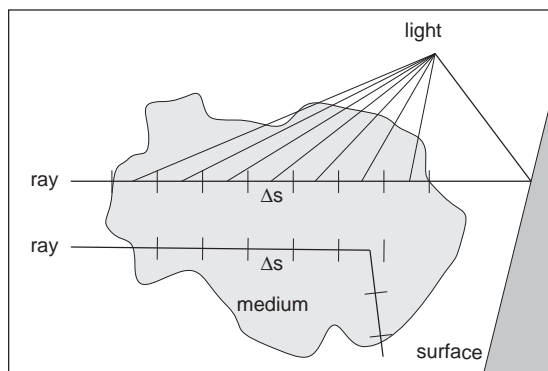
Figure 6: Top ray: a traditional ray-march through a heterogeneous medium, and at each step a shadow ray is also marched. Bottom ray: an adjoint photon also steps through the medium (unless the intersection can be done analytically) and if it has an interaction with the volume it generates a new ray only at that point.
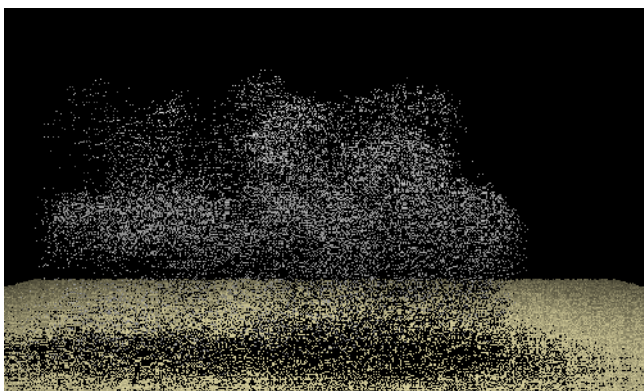

Figure 7: At one sample per pixel each pixel either hits the medium completely or is all background.

5 PARTICIPATING MEDIA

Path tracers have traditionally handled participating media by performing shading at many points along each ray which in turn generates many visibility rays each of which requires many samples in turn as shown for the top ray in Figure 6. For each of the n steps along the ray, an estimate of the radiance reaching that point is computed by scattering according to a phase function, sampling light sources or some combination of the two resulting in an $O(n^2)$ algorithm (each step generates $O(n)$ steps for the radiance estimate and there are $O(n)$ steps). This is just for the single scattering. The multiple scattering is done separately. However, adjoint photon tracers operate as shown for the bottom ray in Figure 6 and only $O(n)$ steps are needed.

The $O(n)$ approach arises naturally in our framework. We treat the participating medium as a probabilistic scatterer/absorber rather than as an attenuating body. An adjoint photon traveling through the medium will either "hit" a particle and cause its shader to be executed, or it will pass through the medium unaffected. Although this results in more randomness for a given ray, it allows us to implement a volume traversal using an $O(n)$ algorithm with the same interface as a surface intersection.

Not surprisingly, the $O(n)$ approach also arises naturally for traditional photon tracers [8] as well as for bidirectional methods [11, 14]. Our approach is almost identical to the first pass of photon mapping, although our single-wavelength implementa-
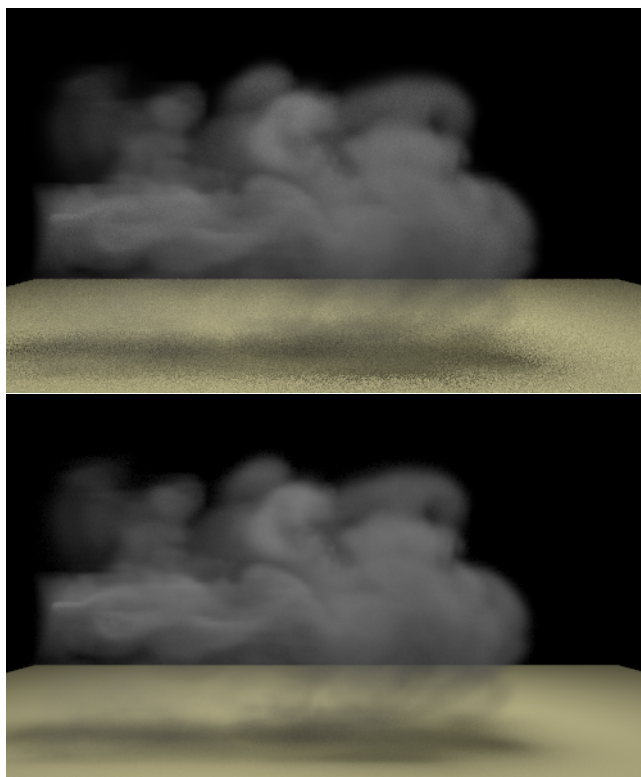

Figure 8: Top: Traditional ray marching, 4 samples/pixel. Bottom: New probabilistic intersection, 400 samples/pixel. These images are computed in approximately the same time.

tion has important simplicity and efficiency advantages for volumes such as air that absorb or scatter different wavelengths with very different probabilities without adding large correction weights. Unlike the second (visualization) pass of photon mapping, we never engage in traditional $O(n^2)$ single-scattering computation.

Figure 7 shows adjoint photon tracing with one sample per pixel and a ray depth of one, with $q(\omega')$ sampling only the luminaire (usually we run it with a mixture density defined over the hemisphere but for this example we hardcode the density for the single scattering case for the purpose of comparison). This effectively does direct lighting within our framework and it demonstrates the probabilistic nature of our intersection routine: adjoint photons either hit the volume or pass through it completely. Figure 8 compares ray marching with our probabilistic intersection method. Note that for 4 samples per pixel, ray marching has mostly converged except for the shadow, which has variance because its samples are distributed across the light source. Our volume shading method runs in linear time, compared to the quadratic time required for traditional ray marching. This increase in efficiency allows us to utilize more samples per pixel with similar computational time. This higher sampling density leads to lower variance in other portions of the scene, such as shadows.

The probability of an adjoint photon scattering is allowed to vary with wavelength. For example, in Rayleigh scattering light is scattered inversely proportional to λ^4. This causes blue light (shorter wavelength) to be preferentially scattered when compared to red light (longer wavelength). With a single wavelength per adjoint photon, implementing Rayleigh scattering is straightforward (see Figure 9). Subsurface scattering is also straightforward to model as a medium inside a dielectric boundary. An example of this is shown in Figure 10.
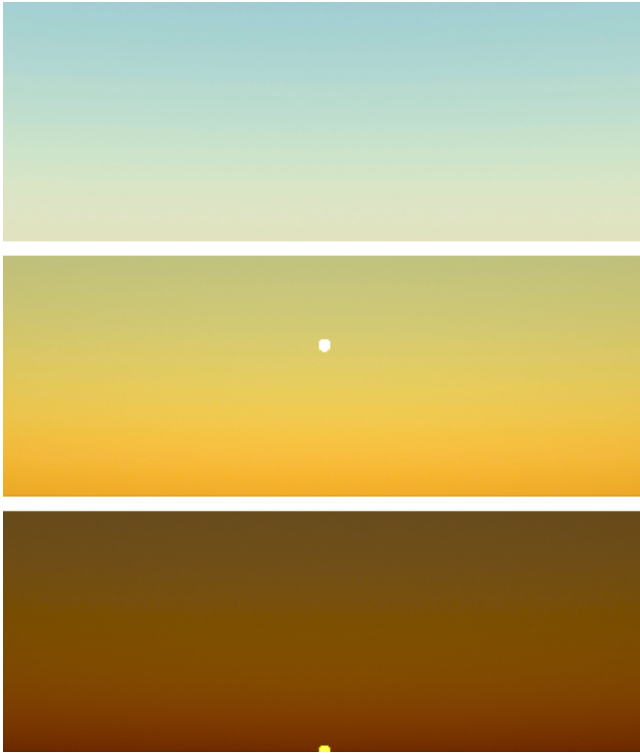
Figure 9: A sky at three different times of day done with a participating media and a sun. The background is black and all color comes from Rayleigh Scattering within the sky participating medium.

Figure 10: Subsurface scattering done automatically: the object is modeled as a polygonal mesh that has a participating medium inside it.

6 IMPLEMENTATION

We implemented adjoint photon tracing in our rendering framework *galileo* using C++ and many of its details are the same as any ray tracer. For each pixel sample, we generate an adjoint photon with a random wavelength, time, lens position, etc. Instead of importance sampling the sensor response function, we simply choose our wavelength uniformly in the visible spectrum. If our sensor response had significant variation with respect to wavelength (e.g. a green CCD pixel that mostly responds to green wavelengths only) this would be fairly inefficient. For the images in this paper, however, we simply use the standard XYZ response curves. When an adjoint photon hits a sensor element, we increment three quantities R_X, R_Y, and R_Z, so the same photons are used in parallel for all three sensor computations for all of the pixels.

Our *galileo* renderer supports a "Surface" class that has a hit routine. The probabilistic media we described in this paper are implemented as a subclass of Surface and are probabilistically hit or not using a media intersection seed (this is in addition to the standard seeds used for lens position, motion blur time, etc). The "SurfaceShader" class in *galileo* has a "kernel" function that computes either a BRDF multiplied by $\cos(\theta)$ or a phase function and scattering albedo depending on whether it is meant for surfaces or media respectively. The SurfaceShader class also supports two sampling functions: "sampleKernel" and "pdfDirection". This pair of functions provide the ability to generate samples according to some density and return the probability of generating a sample for any arbitrary direction. This allows for implementing importance sampling of BRDFs and phase functions and allows for the use of mixture densities and multiple importance sampling. For verification purposes, the default implementation for sampleKernel and pdfDirection represent uniform domain sampling. This allows us to check that our importance sampling converges to the same result as uniform sampling and may also be used for functions that cannot be analytically sampled, such as measured data.

When shading an intersection point, we use a mixture density that combines kernel sampling, luminaire sampling, environment sampling and uniform domain sampling (either hemisphere for BRDFs or full sphere for phase functions) to determine a new directional sample. We evaluate the probability of generating this direction for each of the densities in our mixture and compute the full probability using the weighted average. We also evaluate the kernel value for the chosen direction regardless of which sampling method generated it; for example, if the chosen density is over luminaires the kernel function must still return an answer (even if it is 0). We then weight the kernel value by the pdf and evaluate the spectral radiance of the new ray. This process continues until we reach the background or until a max depth is achieved.

Environments and luminaire emission functions are described using "LightShaders" which provide an "emittedRadiance", "sampleDirection" and "pdfDirection" function. The emittedRadiance function is the function e from before and behaves as it does in any rendering framework, while sampleDirection and pdfDirection behave similarly to the SurfaceShader functions (the major difference being that the luminaire should attempt to choose an important direction relative to the shading point). Efficient environment map sampling is achieved by using a Lambert cylindrical equal-area projection; this causes each pixel in the environment map to have the same solid angle allowing for simple importance sampling of pixel luminance using a summed area table without the need for a correction factor.

A complex scene rendered using our approach is shown in Figure 1. While adjoint photon tracing is relatively slow for simple scenes such as the Cornell box, its merits are evident in dealing with photometrically complex scenes where its robustness and lack of simplifying assumptions allow it to work without special changes.

7 DISCUSSION

Adjoint photon tracing is in some sense a rehash of neutron transport work from the 1940s and 1950s, and most (and possibly all) of the details of its implementation have appeared in different places in the graphics literature. However, we have found that our implementation as a whole seems quite different and is simpler than anything we have seen in the graphics literature or encountered in discussions with other rendering researchers. We now explicitly address several questions about adjoint photon tracing.

Isn't adjoint photon tracing just path tracing? It is similar but has three important differences. First it abandons traditional direct lighting which fundamentally changes the architecture of a path tracer. Second it handles media differently by avoiding traditional ray marching. Finally, it avoids the more complex math of Monte Carlo integration and replaces it with Monte Carlo simulation. Imagine teaching both derivations to a freshman computer science class and that advantage is quickly apparent.

Is per-wavelength sampling inefficient? An immediate question that concerned us is whether associating individual wavelengths is the main source of our variance. As in Walter et al. [23], we have not found color noise to be a significant problem as illustrated by the test shown in Figure 11. Adding spectral effects such as dispersion to such a renderer requires no special handling or complicated code.

Why strive for code simplicity? While the theoretical principles of rendering are fairly well-understood [6, 20], much less is known about how to field systems that deal with complex scenes under complex illumination, with the most well-documented effort being *pbrt* [15]. The *pbrt* system is designed for flexibility and the resulting complexity is managed using careful software engineering. We instead make all software design decisions by choosing the simplest thing that could possibly work, and introducing complexity above that only when simpler solutions have been tried and empirically shown not to work.

Isn't that too many rays? There will always be an important role for batch rendering in both engineering and artistic applications. With advances in CPU resources and ray tracing efficiency structures, approximately one trillion ray-scene intersections can be performed in an overnight run. That is more than enough for most scenes. For the most challenging scenes, more than one CPU can be used as pixel-based techniques easily scale to thousands of CPUs.

Is importance sampling effective? Importance sampling is only as effective as its ability to not miss hugely important paths. This is why bidirectional methods are important for some scenes. However, this paper has used very crude importance sampling and we think we have demonstrated that adjoint photon tracing is an approach worth exploring, especially for complex scenes. More sophisticated importance sampling such as sampling the product of the illumination field and the BRDF could further increase efficiency [3, 19].

Aren't adjoint photons really importons? Christensen has categorized many photon-like particles that might be sent from the sensor [4], but we really are switching the roles of lights and sensors and sending photons without normalization. Once the principle advocated by dual photography is accepted, no knowledge of "importance" is needed.
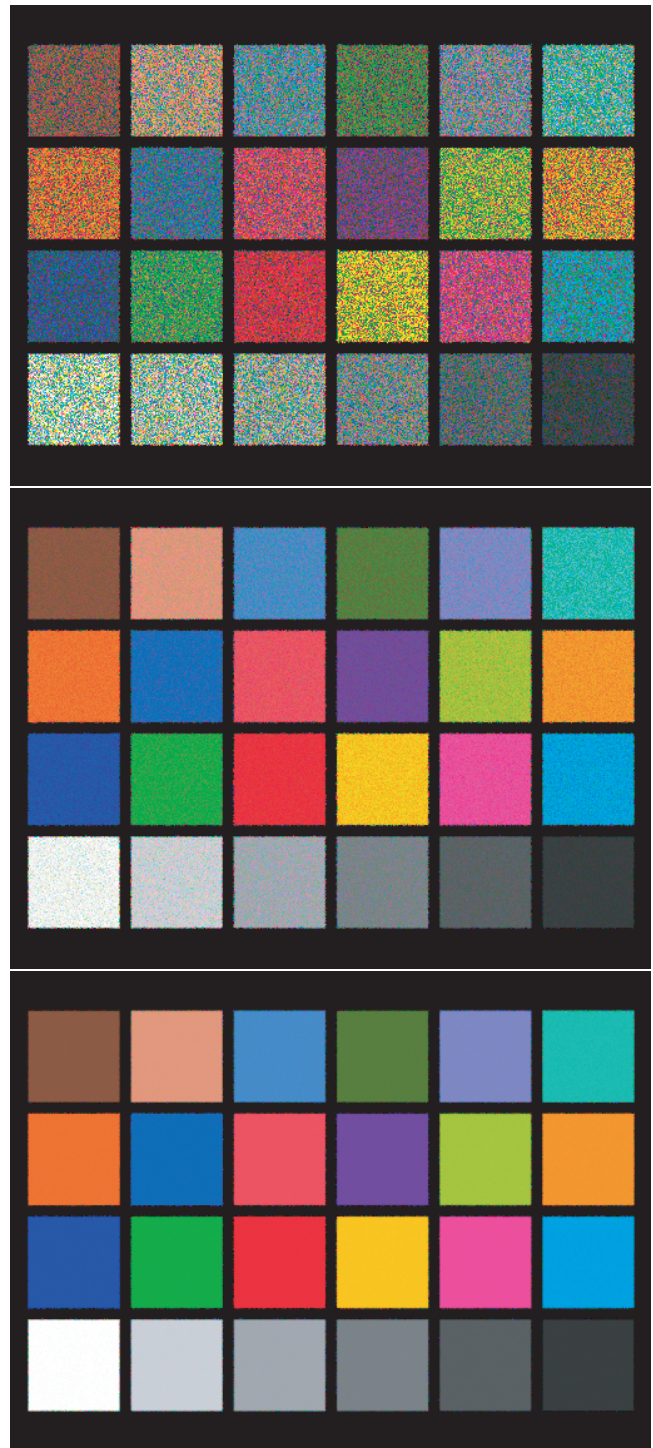

Figure 11: Sampling the Macbeth Color Checker Chart with one random wavelength per ray (adjoint photon) under ambient illumination. Convergence is fairly rapid. Top: 4 samples/pixel. Middle: 16 samples/pixel. Bottom: 49 samples/pixel. All images are computed using one wavelength per adjoint photon.

8 CONCLUSION

An adjoint photon tracer sends adjoint photons from the sensor elements into the environment and records hits at luminaires. The implementation of an adjoint photon tracer is straightforward, and has the important software engineering benefit of being able to implement participating media as randomized surfaces. From a practical standpoint this is very important because, in our experience, supporting media typically uglifies a rendering system to the point where it is hard to maintain it, and in adjoint photon tracing the complexity associated with media is cleanly hidden. Importance sampling can be used to make adjoint photon tracing practical for many scenes, including those with spectral effects and complex participating media. At a high level, an adjoint photon tracing program operates much like a Kajiya-style path tracer. However, in its details it has several advantages, and is based on a more intuitive formulation where the details of Monte Carlo integration are hidden in the importance sampling rates.

REFERENCES

[1] James Arvo. Transfer equations in global illumination. Global Illumination, SIGGRAPH Conference Course Notes, 1993.

[2] James Arvo and David B. Kirk. Particle transport and image synthesis. In *Proceedings of SIGGRAPH*, pages 63–66, 1990.

[3] David Burke, Abhijeet Ghosh, and Wolfgang Heidrich. Bidirectional importance sampling for direct illumination. In *Rendering Techniques*, pages 147–156, 2005.

[4] Per H. Christensen. Adjoints and importance in rendering: An overview. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):329–340, 2003.

[5] Per H. Christensen, David H. Salesin, and Tony D. DeRose. A continuous adjoint formulation for radiance transport. In *Eurographics Workshop on Rendering*, pages 95–104, 1993.

[6] Philip Dutre, Philippe Bekaert, and Kavita Bala. *Advanced Global Illumination*. AK Peters, 2003.

[7] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001.

[8] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of SIGGRAPH*, pages 311–320, 1998.

[9] James T. Kajiya. The rendering equation. In *Proceedings of SIGGRAPH*, pages 143–150, 1986.

[10] Malvin H. Kalos and Paula A. Whitlock. *Monte Carlo methods. Vol. 1: basics*. Wiley-Interscience, New York, NY, USA, 1986.

[11] Eric P. Lafortune and Yves D. Willems. Rendering participating media with bidirectional path tracing. In *Eurographics Rendering Workshop*, pages 91–100, 1996.

[12] Jason Lawrence, Szymon Rusinkiewicz, and Ravi Ramamoorthi. Efficient brdf importance sampling using a factored representation. *ACM Transactions on Graphics*, 23(3):496–505, 2004.

[13] S. N. Pattanaik and S. P. Mudur. The potential equation and importance in illumination computations. *Computer Graphics Forum*, 12(2):131–136, 1993.

[14] Mark Pauly, Thomas Kollig, and Alexander Keller. Metropolis light transport for participating media. In *Eurographics Workshop on Rendering*, pages 11–22, 2000.

[15] Matt Pharr and Greg Humphreys. *Physically Based Rendering*. Morgan Kaufmann, 2004.

[16] Pradeep Sen, Billy Chen, Gaurav Garg, Stephen R. Marschner, Mark Horowitz, Marc Levoy, and Hendrik P. A. Lensch. Dual photography. *ACM Transactions on Graphics*, 24(3):745–755, 2005.

[17] Peter Shirley, Changyaw Wang, and Kurt Zimmerman. Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, January 1996.

[18] Brian E. Smits, James R. Arvo, and David H. Salesin. An importance-driven radiosity algorithm. In *Proceedings of SIGGRAPH*, pages 273–282, 1992.

[19] Justin Talbot, David Cline, and Parris Egbert. Importance resampling for global illumination. In *Rendering Techniques*, pages 139–146, 2005.

[20] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, December 1997.

[21] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of SIGGRAPH*, pages 419–428, 1995.

[22] Eric Veach and Leonidas J Guibas. Metropolis light transport. In *Proceedings of SIGGRAPH*, pages 65–76, 1997.

[23] Bruce Walter, Philip M. Hubbard, Peter Shirley, and Donald F. Greenberg. Global illumination using local linear density estimation. *ACM Transactions on Graphics*, 16(3):217–259, 1997.