An Approach to Point Based Approximate Color Bleeding With Volumes

Christopher J. Gibson[⋆] and Zoë J. Wood Ph.D[⋆⋆]

California Polytechnic State University

Abstract. Achieving realistic or believable global illumination in scenes with participating media is expensive and commonly avoided. This paper introduces an extension to the commonly used point-based color bleeding technique which allows fast, believable in- and out-scattering while utilizing existing data structures and paradigms. . . .

1 Introduction

This paper will outline how we added a light-voxel (lvoxel) representation to an existing global illumination algorithm which leverages a point cloud representation of a scene. In section 2 we discuss the fundamentals behind radiance and the volume rendering equation. Section 3 discusses extending the existing point cloud representation to include transparent surfaces. We then use the results we achieved to compare our extension to the original point based color bleeding implementation as well as a traditional monte-carlo renderer.

2 Background

2.1 Radiance

Irradiance is the change of flux (radiant power) over an area, denoted by $E = \frac{d\Phi}{dA}$ [1]. Another way to look at this problem involves the relationsip between the surface and surrounding radiometric quantities. This can be represented by the following:

$$E = \int L(\mathrm{p} \leftarrow w)\cos\theta dw. \tag{1}$$

$L(\mathrm{p} \rightarrow w)$ represents the radiance leaving point p and $L(\mathrm{p} \leftarrow w)$ represents incoming radiance given a direction w. Note that radiance along a straight path is invariant. For example:

$$L(x \rightarrow y) = L(y \rightarrow x). \tag{2}$$

[⋆] e-mail: cgibson@calpoly.edu
[⋆⋆] e-mail: zwood@calpoly.edu

2.2 Volumes

One of the unique features of participating media is that they must be represented with a more complex data-structure than solid geometric objects which are usually polygonalized in most rendering processes. Light interacts with the particles of a volume, creating complex radiance patterns (increasing the necessary computational complexity exponentially.) The following equations are based off of the Volume Scattering chapter in [2].

Absorbtion: As light passes through a participating media, light will become absorbed based on its absorbtion probability density σ_a. Radiance invariance (2) allows us to estimate the amount of light absorbed or scattered given point p and direction w:

$$e^{-\int_0^d \sigma_a(p+tw,w)dt}, \tag{3}$$

where σ_a represents the probability density that light will be absorbed over a distance dt.

Scatter Out In addition to being absorbed by the medium, light can be scattered based on a scatter probability density σ_s. This reduces the energy of the ray passing through the density.

$$\mathrm{d}L_o(\mathrm{p}, w) = -\sigma_s(\mathrm{p}, w)L_i(\mathrm{p}, -w)\mathrm{d}t. \tag{4}$$

Transmittance Both (3) and (4) involve the reduction of energy through a volume, reducing its transmittance, which can be combined into the following representation:

$$\sigma_t(\mathrm{p}, w) = \sigma_a(\mathrm{p}, w) + \sigma_s(\mathrm{p}, w). \tag{5}$$

Then, integrating the transmittion of the volume from (5) over a ray gives us the following:

$$T_r(\mathrm{p} \to \mathrm{p}') = e^{-\int_0^d \sigma(p+tw,w)dt}. \tag{6}$$

Phase Functions When dealing with particles in volumes that may scatter light, a distribution function or *phase function* describes the angular distribution of light scattered, described as $phase(w \to w')$. The probability that light may scatter from direction w to w' is described using this function.

Scatter In Although σ_s may reduce the energy of a ray passing through a volume, radiance from other rays may contribute to the original ray's radiance. Before we can integrate incoming radiance, we must take into account the *source term* $\mathbb{S}$, or total added radiance per unit distance, where the following constraint must hold true:

$$\int_{\mathbb{S}^2} phase(w \to w')\mathrm{d}w' = 1.$$

This normalization makes sure that the phase function actually defines the probability distribution for a particular direction.

Finally, we can integrate the total radiance scatter based on the normalized phase function $phase(w \to w')$ over all directions w' to get our total scatter in a direction w:

$$S(\mathrm{p}, w) = L_{\mathrm{ve}}(\mathrm{p}, w) + \sigma_{\mathrm{S}}(\mathrm{p}, w) \int_{\mathbb{S}^2} phase(\mathrm{p}, -w' \to w)L_i(\mathrm{p}, w')\mathrm{d}w'.$$

$L_{\mathrm{ve}}(\mathrm{p}, w)$ represents the emission coefficient of a volume and is not discussed in this paper.

3 Algorithm

This algorithm acts as an extension to the point cloud techniques described in [3] and [4], specifically building off of the point based color bleeding technique by Christensen. Because surfels represent opaque materials within the point cloud, an additional representation had to be implemented in order to handle the scatter and absorbtion properties of participating media. We chose to call these lvoxels.

3.1 Point Cloud Generation

We will refer to the octree representation of direct light as a "point cloud" for simplicity, as each point represents some small portion of of a surface or element in a three-dimensional scene. Surfels differ from lvoxels only in that surfels represent a flat, solid geometry while lvoxels represent a transparent, volumetric medium. Both have radii and position so both can be placed within the same point cloud.

Lvoxels are generated by marching through the volume at preset intervals, sampling scatter and absorbtion coefficients in order to get an average throughout the area an lvoxel will occupy.

3.2 PBCB Gather

The gather stage in PBCB calculates the irradiance at a point, given the radiance of the scene around it.

In order to approximate the integral in (1) we sample across a hemisphere oriented along the normal N of the sample surface at point p. Each sample cast out, evaluating $L(\mathrm{p} \leftarrow w)$ which is then multiplied by $w \cdot N$ in order to represent $cos\theta$.

3.3 Adding Scatter-Out

Modifications to the previously mentioned irradiance sampling technique in order to allow scatter-out effects with volumes are few. The biggest changes are to the point cloud octree and its traversal.

Section 3.1 describes the process of adding lvoxels to the octree scene, but if the traversal algorithm were to be left untouched, absorbtion and transmittance would not be taken into account and the traversal would stop at the first lvoxel encountered.

In order to properly evaluate transparent and opaque surfaces within the point cloud, we made changes to node-level octree traversal. Each branch traverses its children from closest to farthest, guaranteeing that closer leaf nodes are evaluated first.

Leaf nodes then use the pre-evaluated scatter (σ_s) and absorbtion (σ_t) coefficients for each lvoxel to appropriately alter the sample ray's transmittence and contribute to the final resulting radiance value. Once a surfel is hit, there is no need to continue traversing the octree.

Even on sparse octrees without volumes, our worst case tests are unaffected, but our best case samples were dramatically improved because further nodes were never traversed, leading to an average performance increase of over 18%.

3.4 Adding Scatter-In

Once the modifications described in Section 3.3 are put in place, the only modifications necessary for scatter-in are to the volume rendering equation. Recall in Section 2.2 that in-scattering required integrating over all directions. Sending out monte-carlo sample rays through the volume and into the scene would be computationally expensive. Instead, for each sample we send out rays into the point cloud, iterating through a much less dense dataset. This helps us replace expensive $S(\mathrm{p}, w)$ evaluations with traversals into the octree.

4 Results

The following test cases were run on a commodity-class Intel i5 3 GHz machine with 4 Gb of ram. Because of the disparity between college-level versus production-class ray tracer implementations, we tested and compared our results against a naive implementation of monte-carlo global illumination not using the point cloud representation. We then compared the resulting images and the time it took to render each. A low difference between images and an increase in efficiency is an ideal result.

4.1 Test Case

The scene tested involved a 60k triangle Sponza Atrium including only vertex and normal information for simplicity. CT scan data of the Stanford Bunny was used in order to test scatter in/out contributions by complex participating media.

4.2 Image Difference

Fig. 2 shows the bunny and sponza stadium showing traditional monte-carlo scattering. At first glance these two images are very similar, however there are a number of small artifacts present in the image rendered with the point cloud representation, and the indirect lighting is slightly darker overall.

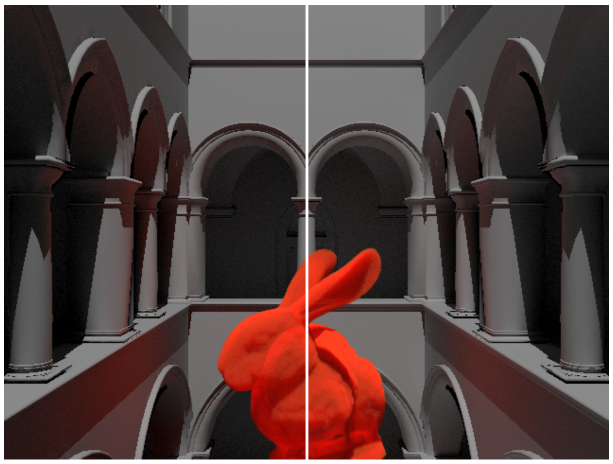

Fig. 1. Comparison of the PBCB extension (left) and traditional monte carlo results (right.)

A closer look at the two results exemplifies the great similarity between the two images, as shown in Fig. **??**.

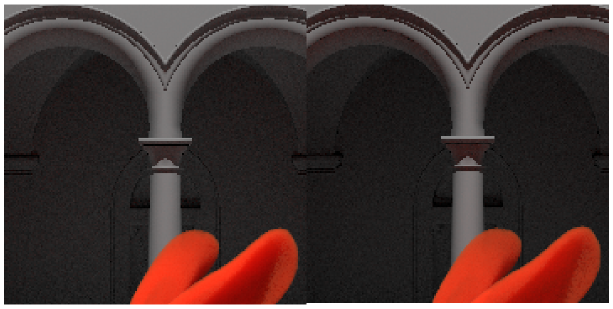

Fig. 2. Zoomed image showing PBCB extension (left) and monte carlo (right.)

4.3 Data Comparison

Scene	Render Time (s)	Image Delta	Memory Overhead
Monte Carlo w/o PBCB	3351 sec	NONE	NONE
Traditional PBCB	???? sec	??%	390 Mb (4.0%)
Extended PBCB	0441 sec	??%	395 Mb (4.1%)

4.4 Analysis

Memory When using traditional PBCB, the real benefit to its surfel representation is shown in more complex scenes. In the sponza atrium, the scene generated over 2.5M surfels for a 60k triangle scene. Adding volume data to the scene does not add an objectionable amount to the point cloud, but for scenes with large volumes the costs could quickly add up without some form of multi-resolution light caching. In this regard, adding yet another representation of the volumes may be expensive, but not prohibatively so. Additionally, larger scenes would benefit from this representation, as it would be significantly simpler than entire scene and can be moved to another system for out-of-core evaluation.

Speed Even without volume integration, monte carlo integration without a lighting representation like PBCB is prohibatively slow for even the simplest scenes. Adding a point cloud representation gave us an impressive speedup. That speedup was compounded even more when volume scattering was added into the tests, showing a factor of eight speedup for our test scenes.

Image Quality Comparing the non-PBCB monte carlo image with that of the traditional PBCB renders shows the clear lack of proper in-/out-scattering. With the extended algorithm, however, the scenes look nearly identical.

We would like to note that there are a number of small artifacts in the PBCB renders due to imprecision and incorrect surfel collisions. It is important to note that, as past papers will attest, such issues are easily overcome and our artifacts are more due to implementation and time constraints than limits on the algorithm itself.

5 Future Work

Multi-Bounce Lighting As mentioned in Christensen's point based color bleeding article, surfels can be modified to "gather" light recursively from their position in the point cloud, allowing for simulated multi-bounce lighting. This would require only a small change to the current algorithm, and would apply for volumes as well to allow very realistic scatter approximations in participating media.

Spherical Harmonic Representation of Volumes In our tests, all participating media scatters light equally in all directions. This is rarely the case, as volumes tend to have unique scatter functions. We can simulate more complex surface scattering functions by creating spherical harmonic representations of the radiance at any specific point in the volume. Our current implementation supports such an approach, but remains untested.

Multi-Resolution Representation Typical implementations of the PBCB algorithm include rougher estimations (usually in the form of a series of spherical harmonic coefficients) at higher levels in the octree, to be evaluated depending on that node's solid angle to our sample point. Due to time constraints, we did not implement full multi-resolution representations of each node. Including LVoxel data in that representation would be a trivial process.

Parallelism Our ray tracer runs a number threads to split the image in order to achieve simple parallelism. Before the threads are created, however, we generate surfels and lvoxels sequentially. Due to the nature of our octree implementation, we cannot add elements and still be thread safe, but this would not be a large obstacle. Scenes like the sponza atrium would run a number of times faster if we were to properly parallelize our implementation more effectively.

6 Conclusion

The addition of the lvoxel paradigm to the already successful point based color bleeding algorithm is shown to be a cost effective method of approximating and evaluating complex scatter functions based on participating media. The speedups are clear, and the memory footprint is easily manageable. The ability to evaluate the irradiance at a point in the scene by using only the point cloud representation is a clear win for out-of-core renderers.

References

1. Philip Dutre, Kavita Bala, P.B.: Advanced Global Illumination. A K Peters (2006)
2. Matt Pharr, G.H.: Physically Based Rendering: From Theory to Implementation. 2 edn. Morgan Kaufmann (2010)
3. Tabellion, E., Lamorlette, A.: An approximate global illumination system for computer generated films. **23** (2004) 469–476
4. Christensen, P.H.: Point-based approximate color bleeding. (2008)

Acknowledgements

A special thanks to Patrick Kelly from DreamWorks Animations for his consistent help and support. Our brainstorming sessions always left me full of great ideas, but also really helped me hone the subject for this project.

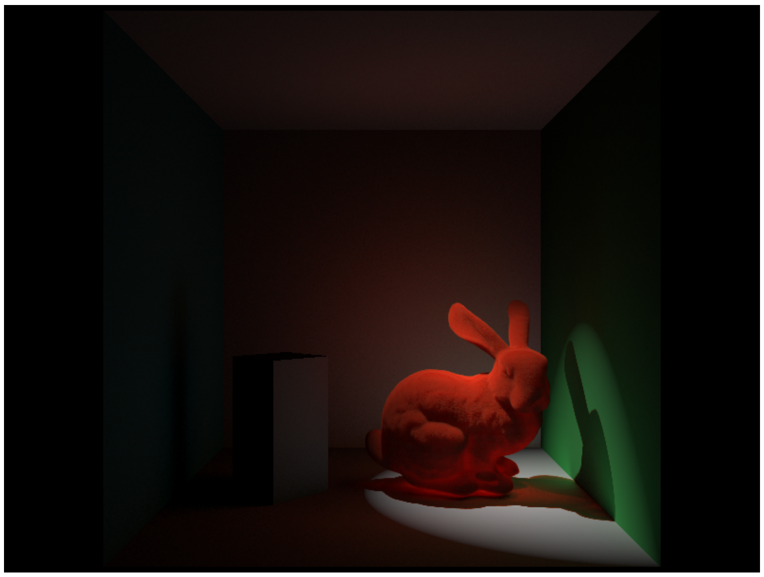

Fig. 3. Image exemplifying clear out-scattering from Stanford bunny volume.

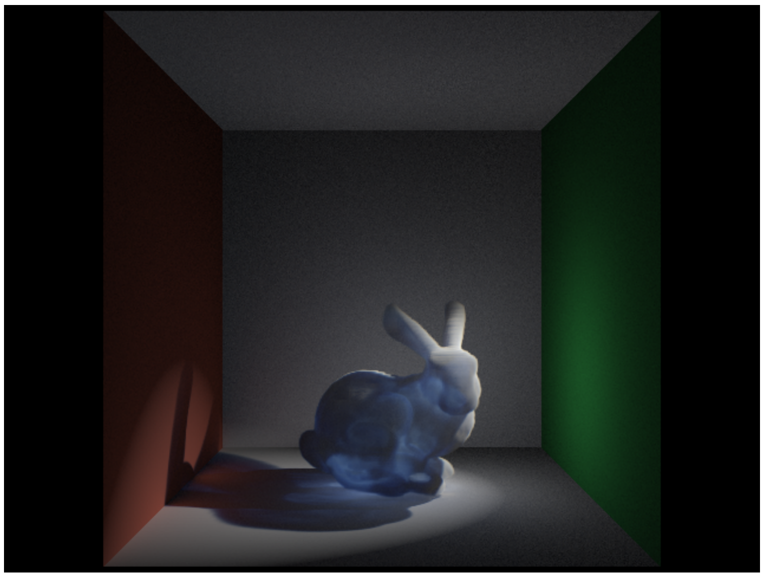

Fig. 4. Image exemplifying clear color bleeding next to the red wall in the bunny's shadow and correct transmittance through the bunny's hollow form.

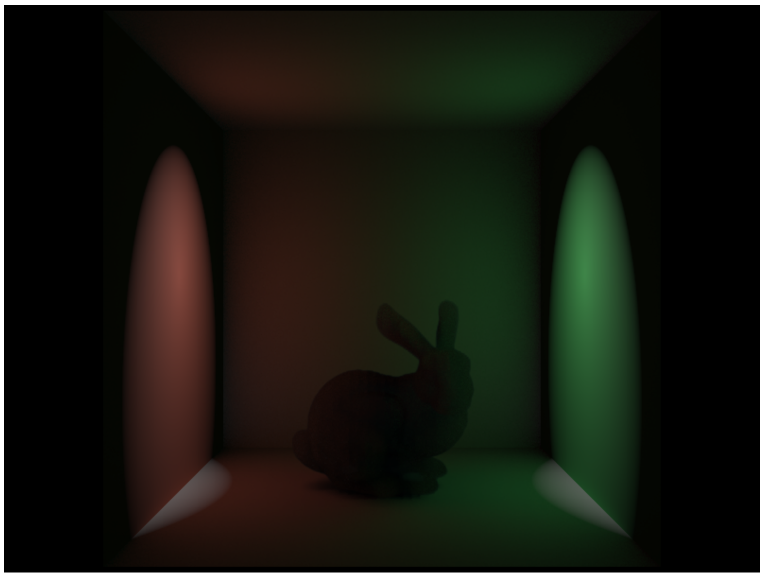

Fig. 5. In-scattering contribution caused by lights pointed at the colored walls.

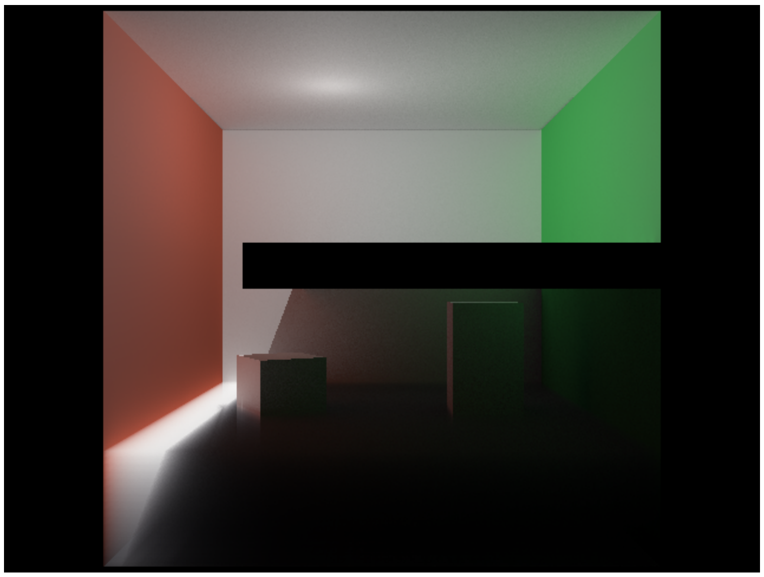

Fig. 6. The black occluding geometry in the center stops all but the light to the left to enter below. In and out scattering on the volume and walls is evident.