
レイトレ合宿７　with FriendsRender R
by tabochan

FriendsRender R

- やったこと
  - Stochastic Progressive Photon Mapping
  - Depth of Field
  - Textured Light
  - Bump map / Alpha map Texture

  - Bounding Volume Hierarchy (2/4/8 分木)
    ＋ なんちゃってMulti-Thread 構築
  - kd Tree

  - クリスマスツリー飾りつけ　謎プログラム

Stochastic Progressive Photon Mapping

Stochastic Progressive Photon Mapping

Toshiya Hachisuka Henrik Wann Jensen
UC San Diego

Figure 1: *Tools with a flashlight. The scene is illuminated by caustics from the flashlight, which cause SDS paths on the flashlight and highly glossy reflections of caustics on the bolts and plier. The flashlight and the plier are out of focus. Using the same rendering time, our method (right) robustly renders the combination of the complex illumination setting and the distributed ray tracing effects where progressive photon mapping is inefficient (left).*

Abstract

This paper presents a simple extension of progressive photon mapping for simulating global illumination with effects such as depth-of-field, motion blur, and glossy reflections. Progressive photon mapping is a robust global illumination algorithm that can handle complex illumination settings including specular-diffuse-specular paths. The algorithm can compute the correct radiance value at a point in the limit. However, progressive photon mapping is not effective at rendering distributed ray tracing effects, such as depth-of-field, that requires multiple pixel samples in order to compute the correct average radiance value over a region. In this paper, we introduce a new formulation of progressive photon mapping, called stochastic progressive photon mapping, which makes it possible to compute the correct average radiance value for a region. The key idea is to use shared photon statistics within the region rather than isolated photon statistics at a point. The algorithm is easy to implement, and our results demonstrate how it efficiently handles scenes with distributed ray tracing effects, while maintaining the robustness of progressive photon mapping in scenes with complex lighting.

1 Introduction

Efficiently computing global illumination is an active areas of research in computer graphics. The types of lighting can vary significantly in different scenes, and it is important to develop algorithms that can handle this variation robustly.

Global illumination algorithms solve the rendering equation introduced by Kajiya [1986]. Unbiased Monte Carlo methods have been a popular approach for computing global illumination without any approximations in the past few decades [Dutré et al. 2006]. However, unbiased methods are not robust under all illumination settings. There are certain light paths that are problematic. For example, path tracing [Kajiya 1986] works well for a scene with diffuse materials, however, it cannot efficiently handle caustics from a small light source.

Hachisuka et al. [2008] observed out that specular-diffuse-specular paths (SDS paths in the light path notation) are particularly problematic for the existing unbiased methods. An example of an SDS path is a light path due to the combination of specular materials and a light bulb. Photon mapping [Jensen 1996] is a biased method which is robust in the presence of SDS paths. However, the results suffer from bias, which appears as low frequency noise in the rendered images. Moreover, computing the correct solutions requires storing an infinite number of photons in the limit. Progressive photon mapping [Hachisuka et al. 2008] solves this issue by using progressive refinement, and makes it possible to compute a correct solutions without storing any photons. Moreover, the method retains the robustness of photon mapping.

Although each radiance estimate in progressive photon mapping converges to the correct radiance, the algorithm is restricted to computing the correct radiance value at a *point*. This property limits the applications of progressive photon mapping because we often need to compute the correct average radiance value over a *region*. For example, anti-aliasing in ray tracing requires the average radiance value for a pixel footprint. Depth-of-field is another example where each pixel value is the average radiance value for the part of a scene that is visible through the lens. In general, all the effects that can be achieved by distributed ray tracing [Cook et al. 1984] require computing the average radiance over some domain.

In this paper, we present a new formulation of progressive photon mapping that enables computing the correct average radiance value over a region. Our formulation requires a simple algorithmic modification, which consists of adding a distributed ray tracing pass after each photon pass in progressive photon mapping. The main contribution is this new formulation that allows simple, yet effective improvement of the robustness of the progressive photon mapping. We show that our modification allows us to render scenes with distributed ray tracing effects in combination with complex illumination scenarios.

Distributed ray tracing できるようになった PPM

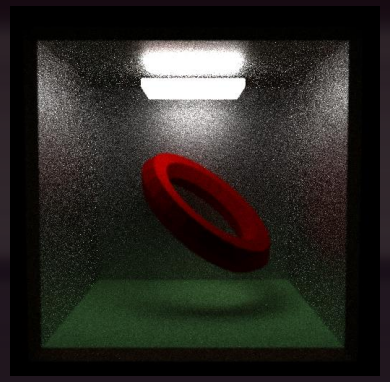
とりわけ DoF / GGX などができるようになる

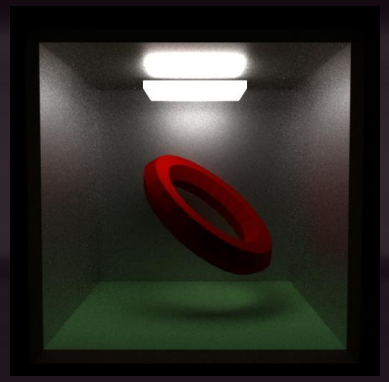
- HitPoint (Photonを集める点) を生成
- Photon ばらまき
- HitPoint での輝度推定

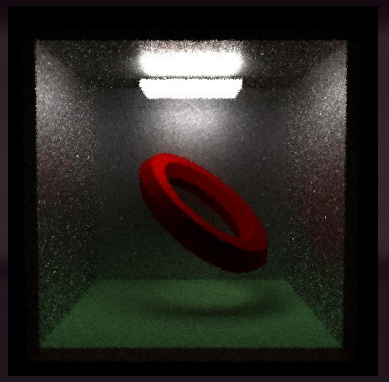
をひたすらループ

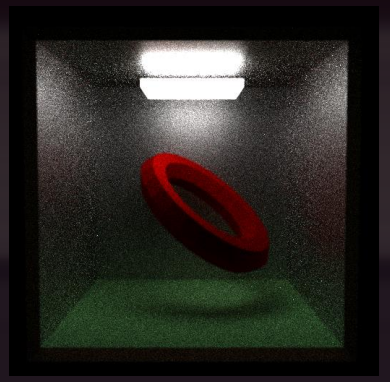
Stochastic Progressive Photon Mapping

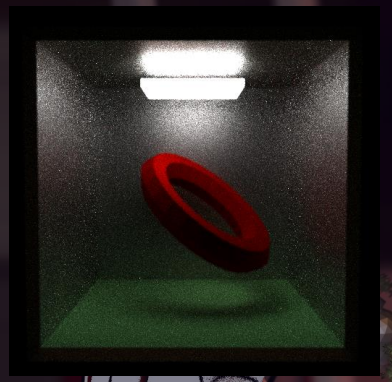
Pathtracing, NEE , PPM , SPPM
ぜんぶ実装して比較


SPPM


Mitsuba


PPM


Path


MIS

BVH & kd木

□　論文を読み直したので、せっかくだからまた実装

子ノードをソートしないでソートする　　qsort よりちょっと早い
- 分割軸の情報で並び替える

2, 4, 8 分木でパフォーマンス比較　　8 分木がわずかに性能よろし
- 4,8 分木は SIMD 実装

なんちゃって Multi Thread 構築　　ちょっと早い
- 8分割を二回行って 64ノード作った後、omp parallel する

□　kd木のMulti Thread 構築を実装する時間がなかった。。。

kd木構築に 1thread 割り当てて、残りは HitPoint の生成をするようにした
効果のほどは不明。。

飾りつけ謎プログラム

木のモデル　と　オーナメント　を入力すると
自動で配置する

が

実際やってみると
↓　みたいに、下にばっかりくっついてキモい

下にはくっつきにくくなるよう、Probability Density をいじって改善する。。
それでもやっぱり、手作業での調整が少し必要

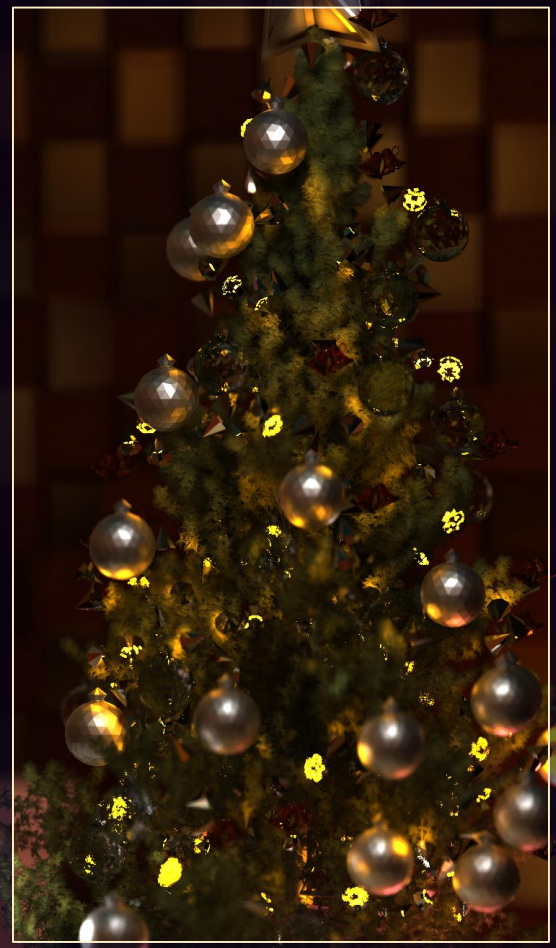
アピールポイント

光源は攻め気味

1. GGX Glass の中に 光源

2. (まあまあ) Many Light

3. Texture Light



アピールポイント

謎シャンデリア(?)

アピールポイント

実は GGX じゃない

高解像度の Bump Mapping がなされた 鏡面

Position-Normal Distributions for Efficient Rendering of Specular Microstructure

Ling-Qi Yan[1] Miloš Hašan[2] Steve Marschner[3] Ravi Ramamoorthi[4]

[1]University of California, Berkeley [2]Autodesk [3]Cornell University [4]University of California, San Diego

Abstract

Specular BRDF rendering traditionally approximates surface microstructure using a smooth normal distribution, but this ignores glinty effects, easily observable in the real world. While modeling the actual surface microstructure is possible, the resulting rendering problem is prohibitively expensive. Recently, Yan et al. [2014] and Jakob et al. [2014] made progress on this problem, but their approaches are still expensive and lack full generality in their material and illumination support. We introduce an efficient and general method that can be easily integrated in a standard rendering system. We treat a specular surface as a four-dimensional *position-normal distribution*, and fit this distribution using millions of 4D Gaussians, which we call *elements*. This leads to closed-form solutions to the required BRDF evaluation and sampling queries, enabling the first practical solution to rendering specular microstructure.

Keywords: rendering, specular highlights, glints, surface microstructure, normal distribution function

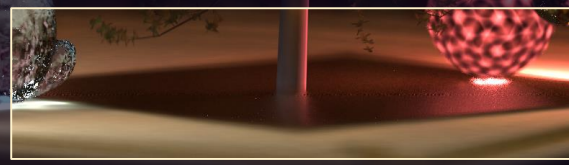
Concepts: •**Computing methodologies** → **Rendering; Reflectance modeling;**

1 Introduction

Sharp lighting exposes a large amount of structure in real-world specular highlights, especially in close-ups (like in Figure 1, left). However, the conventional microfacet BRDF models the detailed surface micro-structure using a smooth normal distribution function (NDF). While this leads to good results under distant views and smooth illumination, the high-frequency glint structure is lost (Figure 1, right). It is certainly possible to *model* explicit surface microstructure using normal mapping, but the resulting *rendering* problem is challenging for straightforward Monte Carlo approaches, because uniform pixel sampling misses the tiny, strong highlights.

Recently, there has been progress on resolving the full detail of real-world specular surfaces, in the work of Yan et al. [2014] and Jakob et al. [2014]. While the results demonstrated in their papers are compelling, these methods still have drawbacks. Yan et al. [2014] computes an accurate solution for a surface defined by an explicit high-resolution normal map. To achieve this, however, a fine tessellation of the normal map into triangular elements is necessary, and the contribution of a single element to the result requires a rather involved computation. Jakob et al. [2014] defines distributions and orientations of specular point scatterers implicitly; this has the advantage of low storage requirements, but only works for random piece-wise flat surfaces, and loses the ability to represent specific desired surfaces with local curvature. Neither method is fast or easy to integrate in a rendering system, which limits their practical impact.

だが実装はできなかった。。。



次回

GPUつかいたい