Report on 'Differentiable Monte Carlo Ray Tracing through Edge Sampling' by TZU-MAO LI, MIIKA AITTALA, FRÉDO DURAND, JAAKKO LEHTINEN et al.

SIDDHESH KALEKAR

Computer graphics, machine learning, and computer vision all place an increasing emphasis on gradient-based techniques. Deep learning, inverse problems, and optimisation all need the capacity to calculate gradients. The gradient is necessary for rendering when considering elements like camera settings, light sources, scene geometry, or material characteristics. However, since the rendering integral contains visibility elements that are not differentiable, determining the gradient of rendering is difficult. A general-purpose differentiable ray tracer is introduced in the paper that can calculate the derivatives of scalar functions over a rendered picture with respect to any scene parameter, including camera position, scene geometry, materials, and lighting parameters. The stochastic solution is based on Monte Carlo ray tracing. A new techniques to explicitly sample edges of triangles in addition to the usual solid angle sampling is introduced. Running time varies from a second to a minute depending on the required precision, for an overhead of roughly 10-20 times compared to rendering an image alone.

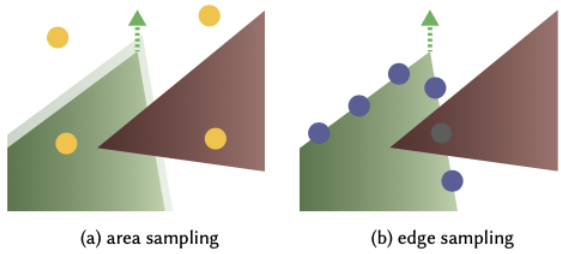
Φ is the set of parameter set like camera pose, scene geometry, material and lighting parameters. Now, goal is to find gradient of scalar function like loss function, with respect to these parameters in Φ. Pixel colour is determined by integrating over all light paths that pass the pixel filter. Monte Carlo Sampling is used to estimate both the values of integral and gradient of the integral.

Pixel colour is integration over all light paths, monte carlo is sampling is used to estimate both integral and it's gradient. Gradient is discontinuous due to edges and occlusions. Thus, integrand will have dirac-delta term with it. Method is divided into two parts, for smooth functions like Phong shading, traditional area sampling algorithms are used. For discontinuous parts, novel edge sampling algorithm is used to capture the changes at boundaries. Assumption in novel approach is that triangles have been preprocessed such that there is no interpenetration. This can be done by breaking triangles or meshes into smaller traingles or meshes.

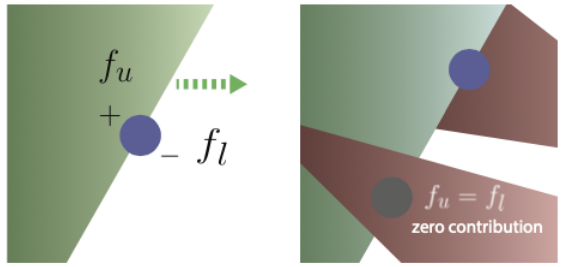
Pixel colour I is given by the integral $I = \iint k(x,y)L(x,y)\,dx\,dy = \iint f(x,y)\,dx\,dy$. We want gradient respect to some scene parameter Φ, $\nabla I = \iint f(x,y;\Phi)\,dx\,dy$. These integrals don't usually have closed form solutions, thus we rely on Monte Carlo Estimations. $f(x,y)$ might not be differentiable with respect to scene parameters Φ, thus to calculate gradient we might not be able to use Monte Carlo methods. Here, $f(x,y)$ is broken into two halves by the plane $\alpha(x,y) = ax+by+c$ which is discontinuity plane and now $f(x,y)$ can be rewritten using heavyside step function $\theta(\alpha(x,y))f_u(x,y) + \theta(-\alpha(x,y))f_l(x,y)$. These discontinuities happen at the edges of the triangles. Now, since there can be multiple edges, there will be multiple alpha functions involved. $I = \iint f(x,y)\,dx\,dy = \Sigma_i \iint \theta(\alpha_i(x,y))f_i(x,y)\,dx\,dy$. Now, gradient can be written as, $\nabla I = \Sigma_i \iint (\delta(\alpha_i(x,y))\nabla\alpha_i(x,y)f_i(x,y)\,dx\,dy + \theta(\alpha_i(x,y))\nabla f_i(x,y)\,dx\,dy)$. Second term can be computed with automatic differentiation. But, first term is modified to be integrated only along the edge of discontinuity as it's zero everywhere else. After this monte carlo estimator is used on this term. More, detailed mathematics is in the paper.

This method handles occlusion correctly, since if the sample is blocked by another surface, (x,y) will always land on the continuous part of the contribution function $f(x,y)$. Such samples will have

Author's address: Siddhesh Kalekar.



(a) area sampling (b) edge sampling



(a) half-spaces (b) occlusion

zero contribution to the gradients. These equations can be easily modified, to involve secondary effects like shading as they also contain a dirac-delta term over some boundary.

The paper also discusses the challenge of efficiently sampling edges in rendering and proposes a method to selectively sample important edges for rendering while overcoming inefficiencies in uniform sampling. Sampling edges from hundreds of thousands or millions of triangles in a scene is impractical, as most edges don't significantly contribute to the final image. The proposed importance sampling strategy addresses this by selecting edges based on geometric foreshortening, material response, and radiance incoming, prioritizing those likely to have the highest impact.

To streamline edge selection, the authors introduce two hierarchical data structures. The first hierarchy encompasses edges linked to a single face or meshes without smooth shading, organized using a 3D bounding volume hierarchy. The second hierarchy includes remaining edges, employing a 6D bounding volume hierarchy based on endpoint positions and face normals. During traversal, edges overlapping with the light source cone at the shading point are emphasized, and the results are combined using multiple importance sampling. A technique for importance sampling a single edge is outlined, considering highly-specular BRDFs. Evaluations demonstrate that the proposed method significantly reduces image variance compared to uniform edge sampling, particularly in capturing rare events such as shadows from small light sources and highly specular reflections.

The "Results" section elaborates on the implementation, verification, and application of the proposed method for differentiable rendering. The authors have developed a C++ renderer with an interface to the PyTorch automatic differentiation library, enabling users to construct scenes using PyTorch tensors, and generating rendered images as PyTorch tensors in the forward pass. The system facilitates backpropagation to optimize scene parameters based on a scalar loss computed from the rendered image, demonstrating the effectiveness of their method in the context of gradient-based optimization.

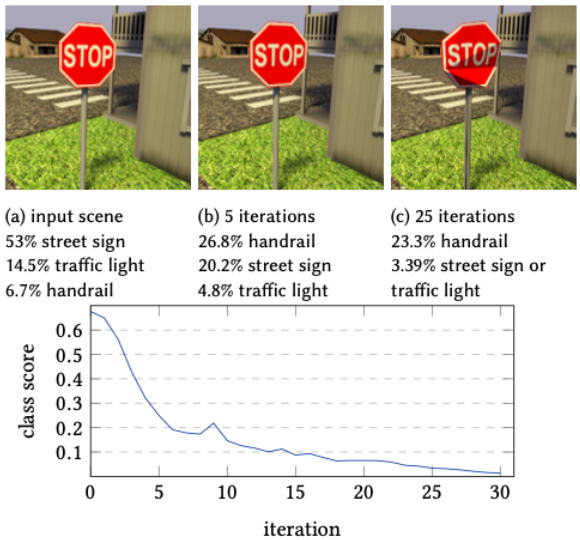
Implementation and Verification: The renderer leverages Embree library for ray casting operations and supports a variety of rendering features such as different BRDFs, textures, area light sources, and projection types. The authors verify the correctness and effectiveness of their method

by testing it on synthetic scenes that encompass various effects like occlusion, non-Lambertian materials, and global illumination. They optimize parameters to minimize the L2 difference between the rendered image and a target image, demonstrating that their renderer accurately generates gradients for optimizing scenes even in complex light transport scenarios.

Comparison with Other Differentiable Renderers: The proposed method is compared with two existing differentiable renderers, OpenDR and Neural 3D Mesh renderer. The comparison highlights the superiority of the proposed method in supporting arbitrary non-Dirac materials, shadows, and global illumination. Unlike previous methods, the proposed approach accurately handles brightness variations due to lighting, providing unbiased gradients.

Inverse Rendering Application: The authors showcase the applicability of their method in an inverse rendering task, involving the fitting of camera pose, material parameters, and light source intensity. The results exhibit successful optimization of scene parameters, underscoring the practical utility of their method in realistic rendering scenarios.

3D Adversarial Example: The paper demonstrates the application of their approach to generate adversarial examples for 3D scenes. By optimizing various scene parameters, they manipulate the rendering of a stop sign to mislead a VGG16 classifier, showcasing the potential for using their gradients to explore interesting scene configurations and potentially mine training data.



In summary, the proposed method not only provides an effective and accurate approach for rendering and optimization but also showcases its versatility through inverse rendering and adversarial example generation. The comparisons with existing methods underscore the advancements and capabilities offered by the proposed differentiable rendering technique.

Limitations:

Performance: The current CPU implementation is relatively slow, taking seconds to minutes to generate small resolution images with a low number of samples. Enhancements in sampling algorithms and compiler techniques for optimization are suggested for future work to address this limitation.

Light Transport Phenomena: The method assumes static scenes without participating media and does not handle motion blur, interpenetrating geometries, parallel edges, or shader discontinuities. Incorporating these aspects and handling interpenetration through mesh splitting are identified as areas for improvement.

Shader Discontinuities: The method assumes differentiable BRDF models and shaders, not accounting for shader discontinuities. This limitation can be addressed through compiler techniques for band-limiting BRDFs.

Conclusion: The paper presents a novel differentiable Monte Carlo ray tracing algorithm capable of generating accurate gradients for a wide range of input parameters in rendering, including scene geometry, camera, lights, and materials. The introduced edge sampling and hierarchical sampling algorithms enable efficient handling of geometric discontinuities and emphasize important edges.