MINGRAD

水鳥（@mizuoooon）

テーマ

勝敗を気にせずやりたい事をやる

やりたい事

微分可能レンダリングしてぇ～

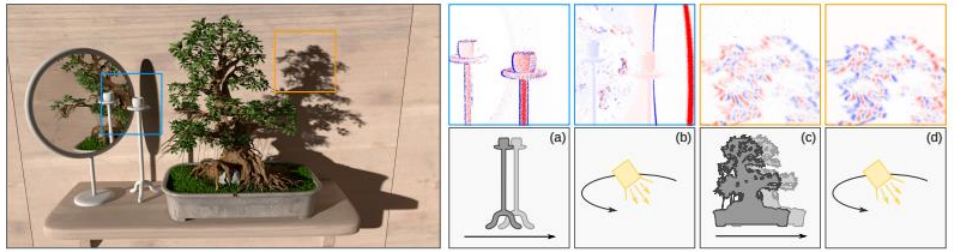
手法

Reparameterizing Discontinuous Integrands for Differentiable Rendering [Loubet, 2019]

- エッジの微分を定義
- シーンパラメーターについて微分可能になるよう積分パラメーターを再定義

Reparameterizing Discontinuous Integrands for Differentiable Rendering

GUILLAUME LOUBET, École Polytechnique Fédérale de Lausanne (EPFL)
NICOLAS HOLZSCHUCH, Inria, Univ. Grenoble-Alpes, CNRS, LJK
WENZEL JAKOB, École Polytechnique Fédérale de Lausanne (EPFL)



A scene with complex geometry and visibility (1.8M triangles)

Gradients with respect to scene parameters that affect visibility

Fig. 1. The solution of inverse rendering problems using gradient-based optimization requires estimates of pixel derivatives with respect to arbitrary scene parameters. We focus on the problem of computing such derivatives for parameters that affect visibility, such as the position and shape of scene geometry (a, c) and light sources (b, d). Our renderer re-parameterizes integrals so that their gradients can be estimated using standard Monte Carlo integration and automatic differentiation—even when visibility changes would normally make the integrands non-differentiable. Our technique produces high-quality gradients at low sample counts (64 *spp* in these examples) for changes in both direct and indirect visibility, such as glossy reflections (a, b) and shadows (c, d).

Differentiable rendering has recently opened the door to a number of challenging inverse problems involving photorealistic images, such as computational material design and scattering-aware reconstruction of geometry and materials from photographs. Differentiable rendering algorithms strive to estimate partial derivatives of pixels in a rendered image with respect to scene parameters, which is difficult because visibility changes are inherently non-differentiable.

We propose a new technique for differentiating path-traced images with respect to scene parameters that affect visibility, including the position of cameras, light sources, and vertices in triangle meshes. Our algorithm computes the gradients of illumination integrals by applying changes of variables that remove or strongly reduce the dependence of the position of discontinuities on differentiable scene parameters. The underlying parameterization is created on the fly for each integral and enables accurate gradient estimates using standard Monte Carlo sampling in conjunction with automatic differentiation. Importantly, our approach does not rely on sampling silhouette edges, which has been a bottleneck in previous work and tends to produce high-variance gradients when important edges are found with insufficient probability in scenes with complex visibility and high-resolution geometry. We show that our method only requires a few samples to produce gradients with low bias and variance for challenging cases such as glossy reflections and shadows. Finally, we use our differentiable path tracer to reconstruct the 3D geometry and materials of several real-world objects from a set of reference photographs.

CCS Concepts: • **Computing methodologies** → **Rendering**; **Ray tracing**.

Additional Key Words and Phrases: differentiable rendering, inverse rendering, stochastic gradient descent, discontinuous integrands, path tracing

ACM Reference Format:
Guillaume Loubet, Nicolas Holzschuch, Wenzel Jakob, and . 2019. Reparameterizing Discontinuous Integrands for Differentiable Rendering. *ACM Trans. Graph.* 38, 6, Article 228 (November 2019), 14 pages. https://doi.org/10.1145/3355089.3356510

1 INTRODUCTION

Physically based rendering algorithms generate photorealistic images by simulating the flow of light through a detailed mathematical representation of a virtual scene. Historically a one-way transformation from scene to rendered image, the emergence of a new class of differentiable rendering algorithms has enabled the use of rendering in an inverse sense, to find a scene that maximizes a user-specified objective function. One particular choice of objective leads to *inverse rendering*, whose goal is the acquisition of 3D shape and material properties from photographs of real-world objects, alleviating the tedious task of modeling photorealistic content by hand. Other kinds of objective functions hold significant untapped potential in areas

Authors' addresses: Guillaume Loubet, École Polytechnique Fédérale de Lausanne (EPFL), g.loubet.research@gmail.com; Nicolas Holzschuch, Inria, Univ. Grenoble-Alpes, CNRS, LJK, nicolas.holzschuch@inria.fr; Wenzel Jakob, École Polytechnique Fédérale de Lausanne (EPFL), wenzel.jakob@epfl.ch;

0730-0301/2019/11-ART228 $15.00
https://doi.org/10.1145/3355089.3356510

基本構成

- C++
- LibTorch (C++版 PyTorch)
- CPUのみ使用

熱い割り切り

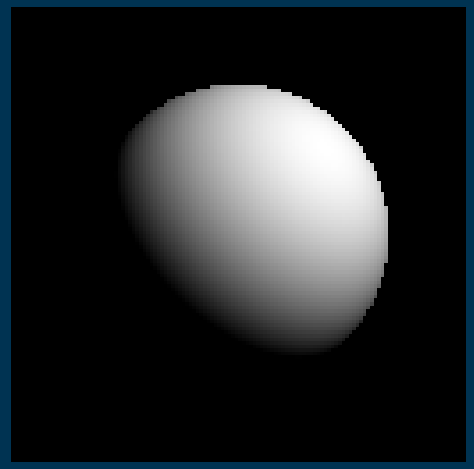
- プライマリレイのみ
- Ortho カメラ
- Acceleration Structure なし
- BRDF なし
- 低解像度 (128x128)

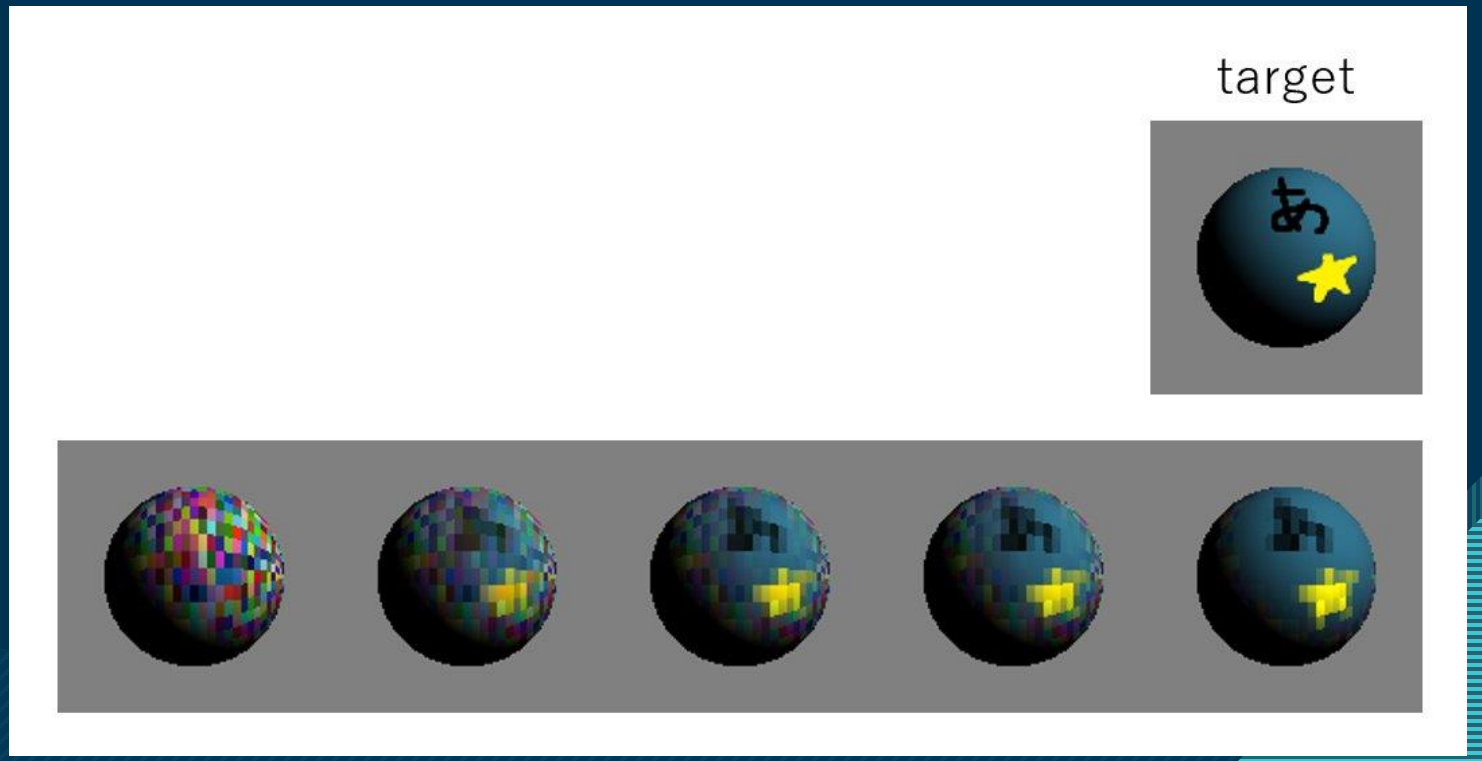
よく考えるとLibTorchじゃなくてよくね

- 別にPyTorchより速くない
- テンソルの型は結局ランタイム
- PyTorchと微妙に違う方言
  - マイナーなので検索しづらい
- C++の既存実装が再利用しやすいかと思ったがそうでもない
  - 結局ベクトル型とかLibTorchのテンソルに置き換えるので
- エラーがまともに出ない
  - （やり方がまずかっただけかも）

とりあえず動かす編

- テンソルを使っていろいろ演算
- 計算グラフを構成しつつ球を描画
- 勝手に微分可能になった





それなりに難しい

- 計算グラフ破壊しないように気を付けて書く
- SoA的な書き方
- テンソルの形状をちゃんと把握してないとエラーになる
  - [1, 2, 3] と [[1], [2], [3]] は違う

例：N個の二次元座標上の点を入力として、それぞれで二次元正規分布の値を評価

やはすぎる

サイズが(N,2)である必要

サイズが(2)である必要

```
Tensor calcWeight(const Tensor& x, const Tensor& sigma) {
    return ((1 / sqrt(2.0f * M_PI * pow(sigma, 2))).unsqueeze(0) * torch::exp(-pow(x, 2) / (2 * pow(sigma, 2)).unsqueeze(0))).prod(1);
}
```

この部分の計算結果は (2) なので
unsqueeze で (1,2) にして次元数を揃える

(N,2)

ここも (2) を (1,2) にする

unsqueeze したおかげで、
サイズは合ってないものの次元数が合うので
broadcast により (N, 2) になる

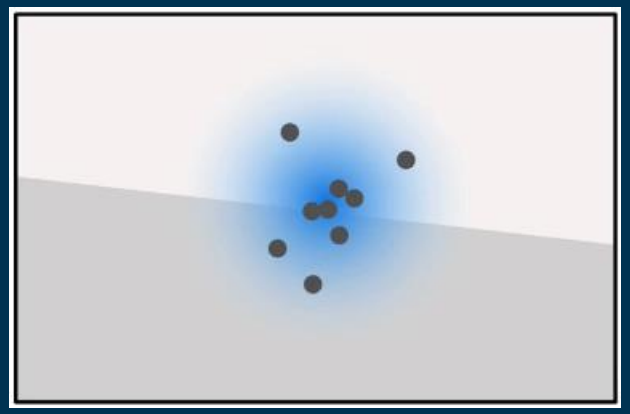
最後にxy方向の
各結果を乗算して
(N) の float が
得られる

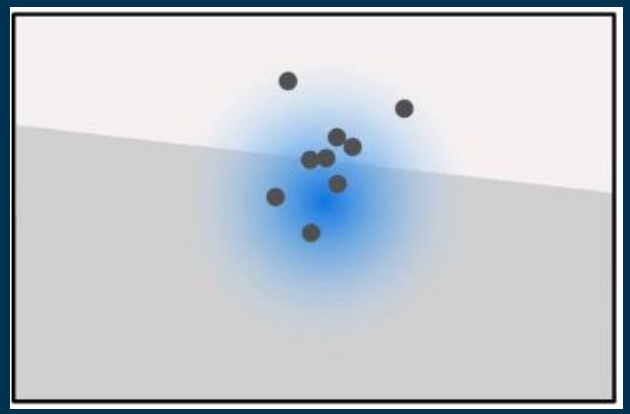
Reparameterizing Discontinuous Integrands for Differentiable Rendering

- ジオメトリのエッジ上での微分を定義
- パラメーターについて微分可能になるように積分をReparameterize
- もともと球面上の積分の話だが、Orthoの画像空間用にアレンジして使用

手法概要

- エッジが移動したらそれに追従してサンプリング点も移動させる

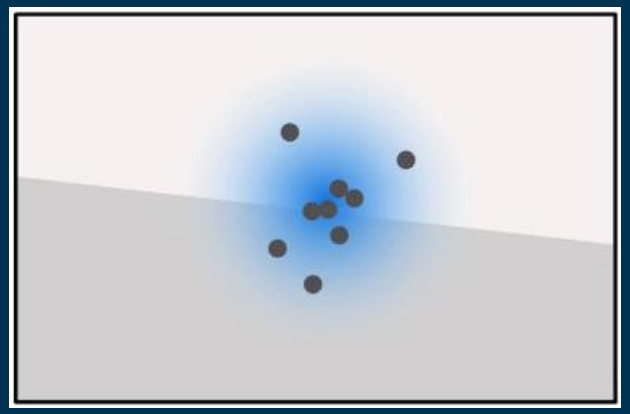


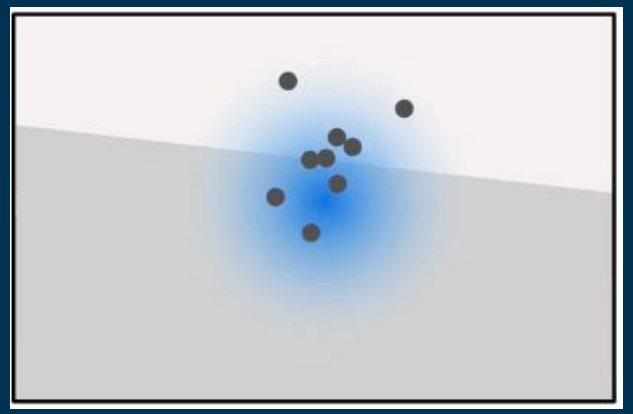


図は論文中より引用

手法概要

- エッジが移動したらそれに追従してサンプリング点も移動させる
- 各サンプリング点の微分の平均がエッジでの微分

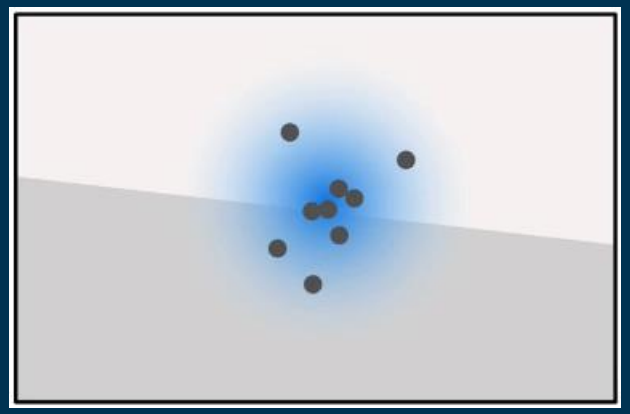


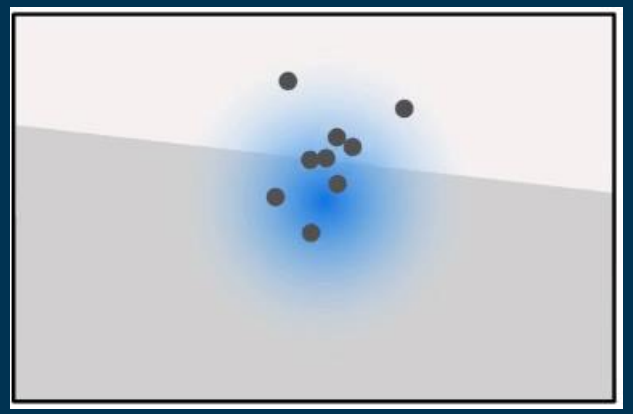


図は論文中より引用

手法概要

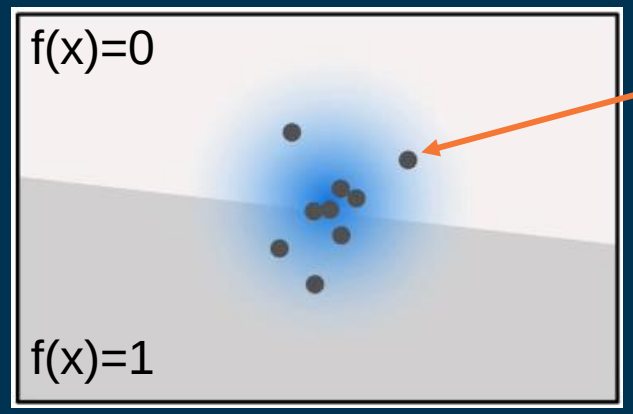
- エッジが移動したらそれに追従してサンプリング点も移動させる
- 各サンプリング点の微分の平均がエッジでの微分





図は論文中より引用

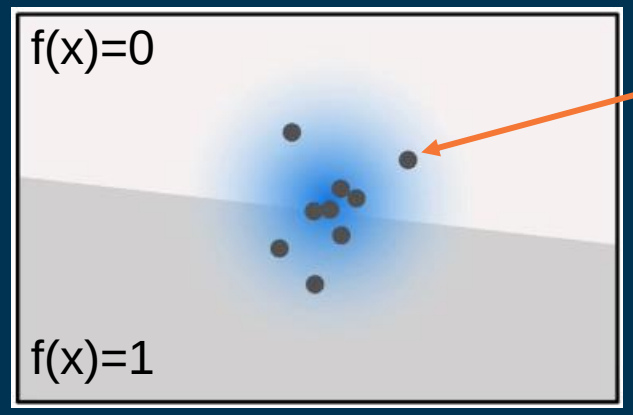
は？と思う気持ち



それぞれの点は移動しても f(x) の値が変わらないので
移動に関する f(x) の微分はゼロになるんでは？
期待値とってもエッジの微分はゼロになるんでは？

図は論文中より引用

は？と思う気持ち

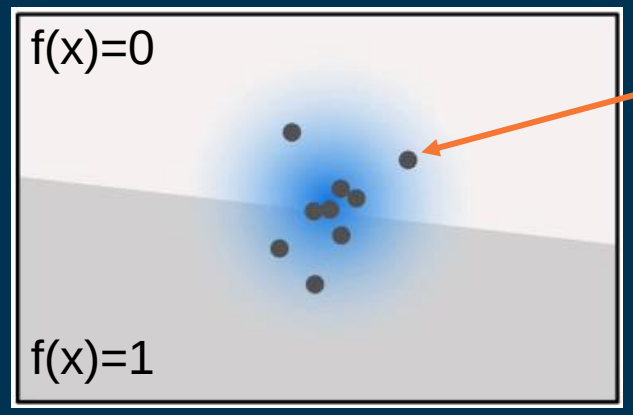


それぞれの点は移動しても f(x) の値が変わらないので
移動に関する f(x) の微分はゼロになるんでは？
期待値とってもエッジの微分はゼロになるんでは？

球面積分領域全体を小さな vMFカーネルの集合の畳み込みで表現された空間とみなす。
モンテカルロサンプリング時は点をサンプリングすると同時にカーネルをサンプリングする。
目標とする積分値は変えないままに、f(x) ではなく
f(x) とカーネルの積に評価対象を変更できる。

図は論文中より引用

は？と思う気持ち



それぞれの点は移動しても f(x) の値が変わらないので
移動に関する f(x) の微分はゼロになるんでは？
期待値とってもエッジの微分はゼロになるんでは？

球面積分領域全体を小さな vMFカーネルの集合の畳み込みで表現された空間とみなす。
モンテカルロサンプリング時は点をサンプリングすると同時にカーネルをサンプリングする。
目標とする積分値は変えないままに、f(x) ではなく
f(x) とカーネルの積に評価対象を変更できる。

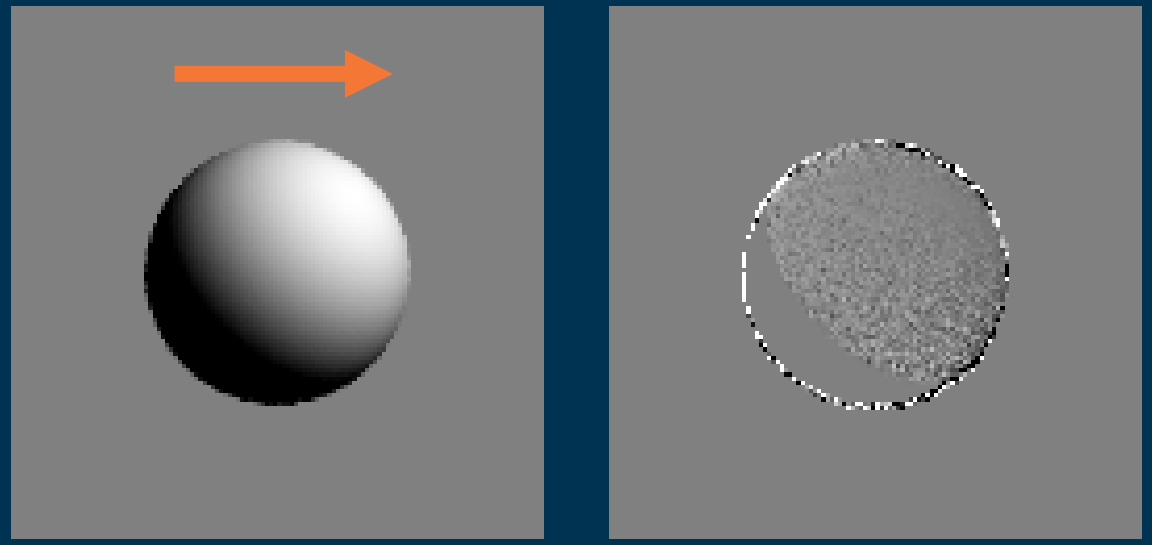
f(x) がエッジ内部で一定値の場合でも、
f(x) とカーネルの積の微分の平均はエッジ付近で値を持つようになる。

図は論文中より引用

細かいテクニック

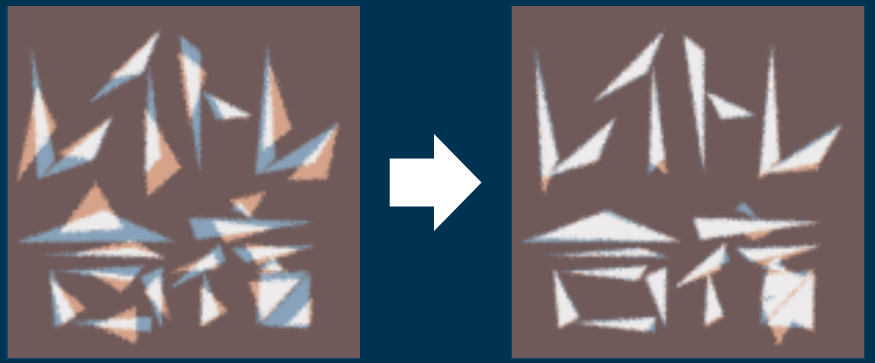
- Control Variants で分散を抑えたり

結果



結果

Target



最適化の難しさ

- 所詮傾きを見てるだけなので最初から三角形の初期値とtargetが
  まったく重なってないと動いてくれない
- 複数の三角形同士が重なったりすると
  頂点動かしても見た目に反映されないので中々抜け出さない

感想

- やりたいことはできてよかった
- 微分可能レンダリング書いたことのある側の人間として生きていきます