Gradient-Domain Path Tracing

Markus Kettunen	Aalto University
Marco Manzi	University of Bern
Miika Aittala	Aalto University
Jaakko Lehtinen	Aalto University and NVIDIA Research
Fredo Durand	MIT CSAIL
Matthias Zwicker	University of Bern

To appear in ACM Transactions on Graphics (Proc. SIGGRAPH 2015)

CG技術の実装と数理 2015

第二回

2015/10/03

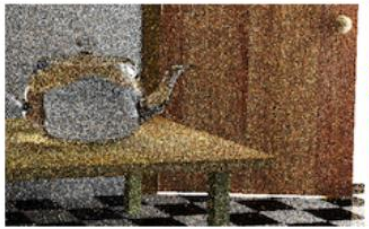
Introduction

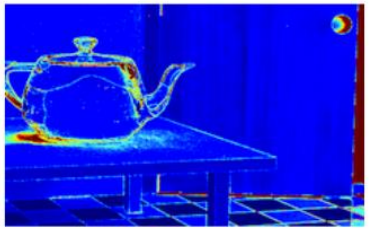
Introduction

- Gradient-Domain Rendering


Horiz. differences I^{dx}


Vertical differences I^{dy}


Coarse image I^{g}

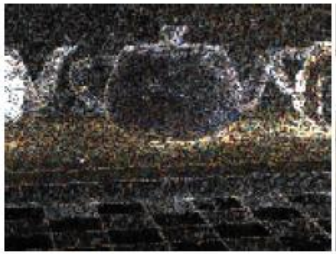

Sample density


Result

Gradient-Domain Metropolis Light Transport [Lehtine 2013]


Coarse image


L_2 (OURS)


Gradients (OURS)


L_2 (GDMLT)


Gradients (GDMLT)

Improved Sampling for Gradient-Domain Metropolis Light Transport [Manzi 2014]

Introduction

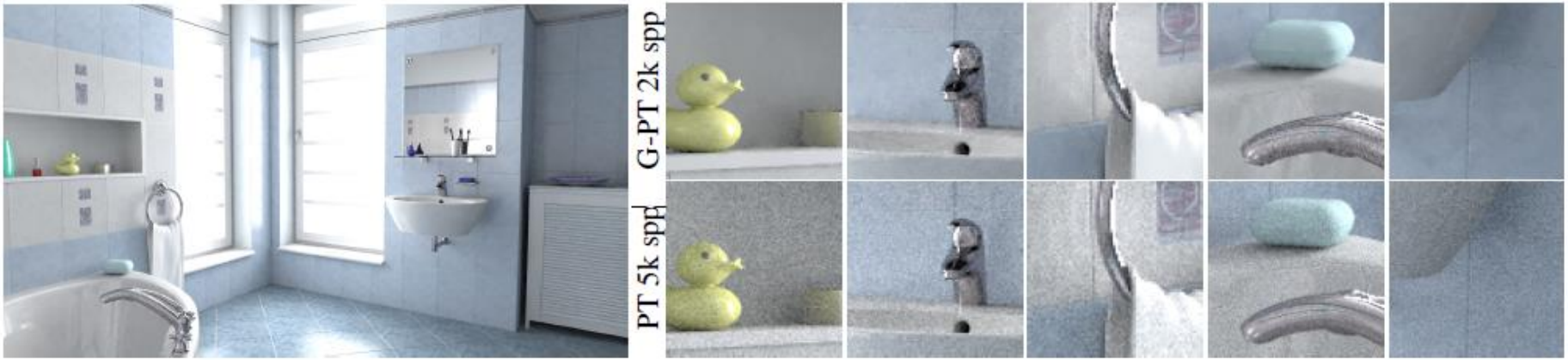
- 高度なレンダリング手法は実装が大変

"Metropolis light transport is notoriously hard to implement, however, and to the best of our knowledge, Mitsuba [Jakob 2012] is the only publicly-available implementation of Veach's original algorithm [Veach and Guibas 1997]."

"メトロポリス光輸送法は実装が超難しいし、自分の知る限り、Veachのオリジナルの手法を実装してあるオープンなレンダラはMitsubaだけだよ!"

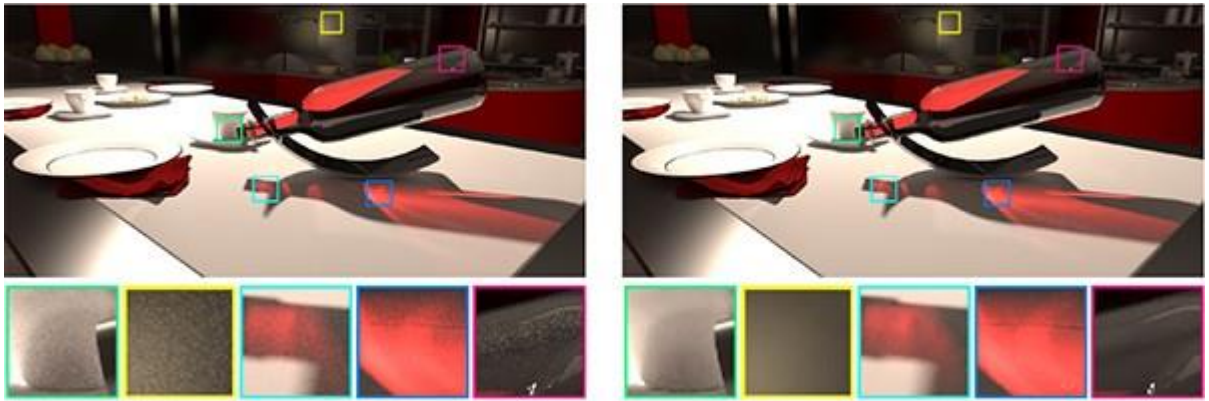
Introduction

- 提案手法
  - Gradient-Domain Rendering → 高効率！
  - 普通のPath Tracingを利用 → 実装が簡単！



Introduction

- Gradient-Domain Rendering



Gradient-Domain Bidirectional Path Tracing [Manzi 2015]

Overview

Overview

1. 勾配画像を直接Path Tracingでレンダリングして得る。（モンテカルロ積分）

Horiz. differences $I^{\mathrm{d}x}$

Vertical differences $I^{\mathrm{d}y}$

Overview

1. 勾配画像を直接Path Tracingでレンダリングして得る。（モンテカルロ積分）
2. 普通にPath Tracingでレンダリング。（モンテカルロ積分）

Coarse image I^g

Overview

1. 勾配画像を直接Path Tracingでレンダリングして得る。（モンテカルロ積分）
2. 普通にPath Tracingでレンダリング。（モンテカルロ積分）
3. 2の結果をヒントにしつつ、1の結果に対するPoisson Problemを解いて元の画像を復元。

直接得られた
勾配画像

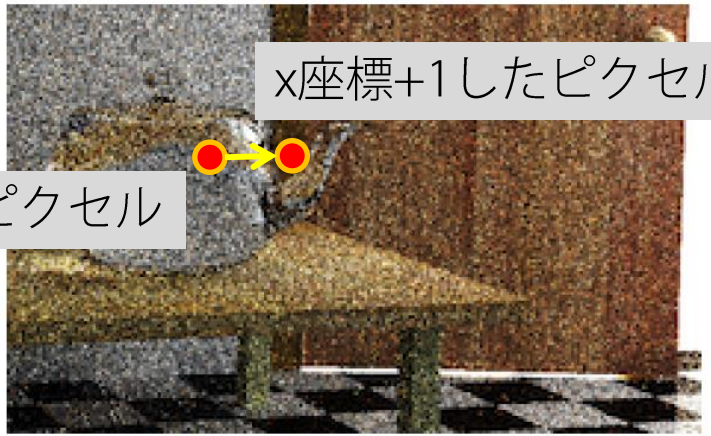
$$\underset{I}{\operatorname{argmin}} \left(\left\| \begin{pmatrix} H^{\mathrm{d}x} I \\ H^{\mathrm{d}y} I \end{pmatrix} - \begin{pmatrix} I^{\mathrm{d}x} \\ I^{\mathrm{d}y} \end{pmatrix} \right\|_2^2 + \left\| \alpha \left(I - I^g \right) \right\|_2^2 \right).$$

復元画像の勾配

復元画像と
Path Tracing画像の差

Overview

1. 勾配画像のレンダリング

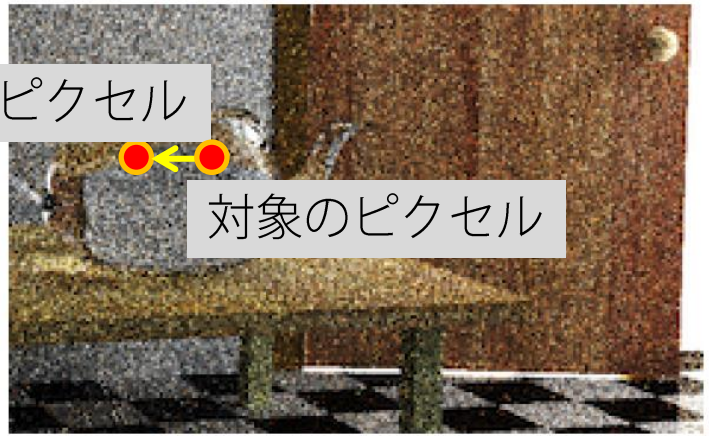


Coarse image I^g

差をとる
（有限差分法）

Overview

1. 勾配画像のレンダリング

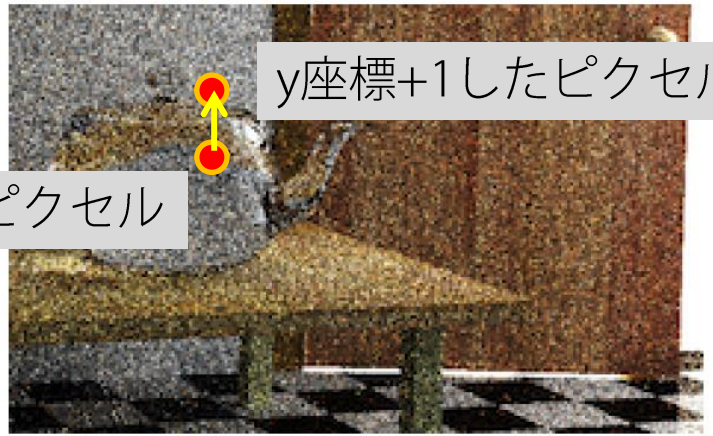


Coarse image I^g

差をとる
（有限差分法）

Overview

1. 勾配画像のレンダリング

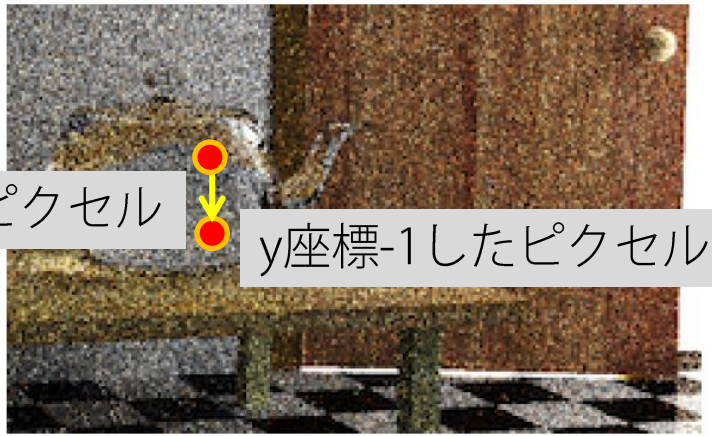


Coarse image I^g

差をとる
（有限差分法）

Overview

1. 勾配画像のレンダリング

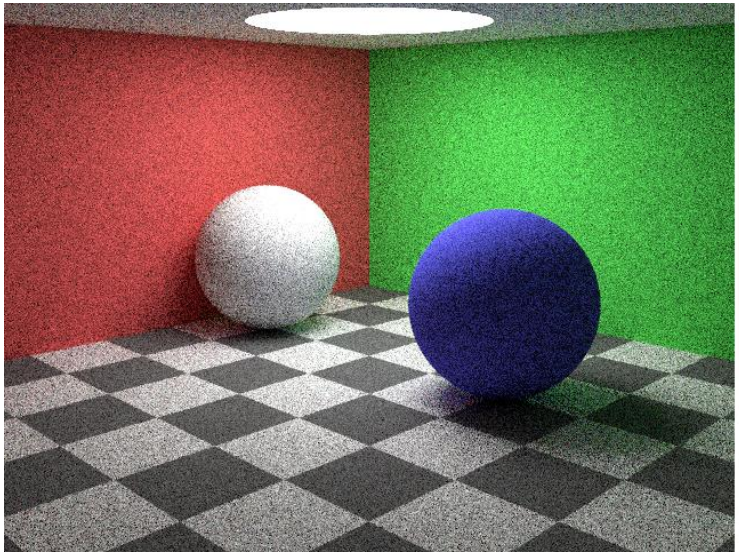


Coarse image I^g

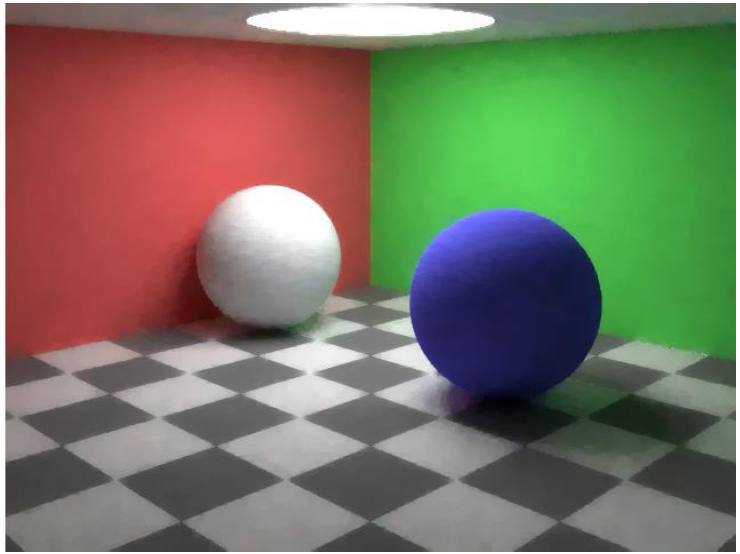
差をとる
（有限差分法）

Implementation

前回のImplementation



100 spp Path Tracing



GD-PT (L1)

なんかボケてる

Reconstruction

- Reconstructionは、L2ノルムなら単純な最小二乗法に帰着する。
- 勾配画像を得る部分にバグ。

$$\underset{I}{\operatorname{argmin}} \left(\left\| \begin{pmatrix} H^{\mathrm{d}x} I \\ H^{\mathrm{d}y} I \end{pmatrix} - \begin{pmatrix} I^{\mathrm{d}x} \\ I^{\mathrm{d}y} \end{pmatrix} \right\|_2^2 + \left\| \alpha \left(I - I^g \right) \right\|_2^2 \right).$$

smallgdpt

- https://gist.github.com/BachiLi/4f5c6e5a4fef5773dab1
- By Tzu-Mao Li
- Durand先生（GD-PTの著者の一人）のところの学生。
- **＋リファレンス実装（？）**
- **＋高速なL2ノルムReconstruction**
  - Fourier Analysis of the 2D Screened Poisson Equation for Gradient Domain Problems [Bhat 2008]によるScreened Poisson Solver

smallgdpt

- https://gist.github.com/BachiLi/4f5c6e5a4fef5773dab1
- By Tzu-Mao Li
- Durand先生（GD-PTの著者の一人）のところの学生。
- **＋リファレンス実装（？）**
- **＋高速なL2ノルムReconstruction**
  - Fourier Analysis of the 2D Screened Poisson Equation for Gradient Domain Problems [Bhat 2008]によるScreened Poisson Solver
- **－スペキュラ周りのShift処理が簡易化されている**
- **－L1ノルム版は無し**

自分の実装

- https://github.com/githole/gdpt

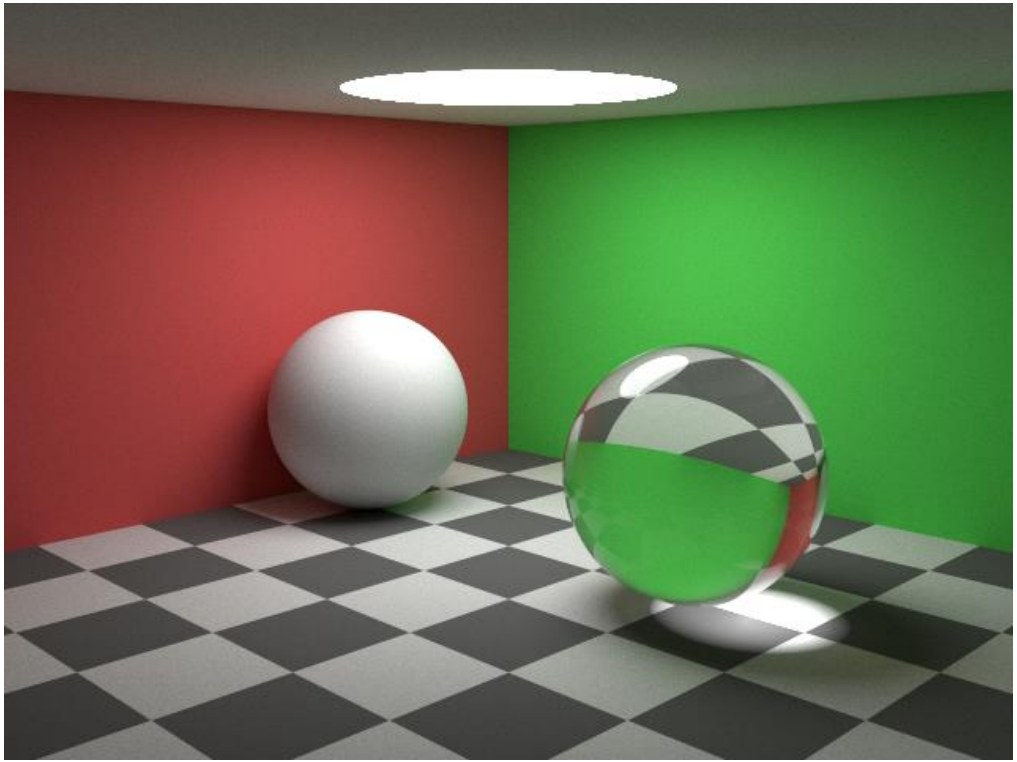
- **＋なるべく論文に忠実っぽい**
- **＋L1ノルムReconstructionも実装**
- **＋ボリュームレンダリング版を一応実装**

自分の実装

- https://github.com/githole/gdpt

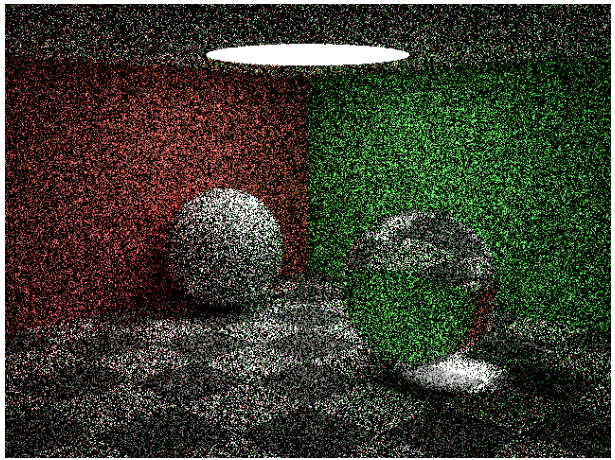
- **＋なるべく論文に忠実っぽい**
- **＋L1ノルムReconstructionも実装**
- **＋ボリュームレンダリング版を一応実装**
- **－スペキュラ周りがちょっと怪しい**
- **－Reconstructionがちょっと遅い**
- **－ボリュームレンダリング版は普通にバグっている**

テストシーンA

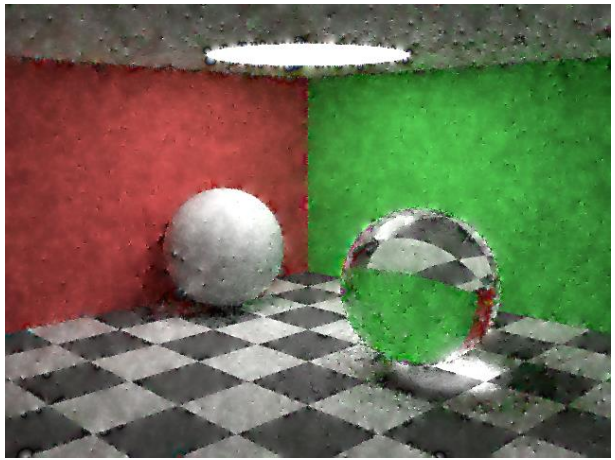


テストシーンA

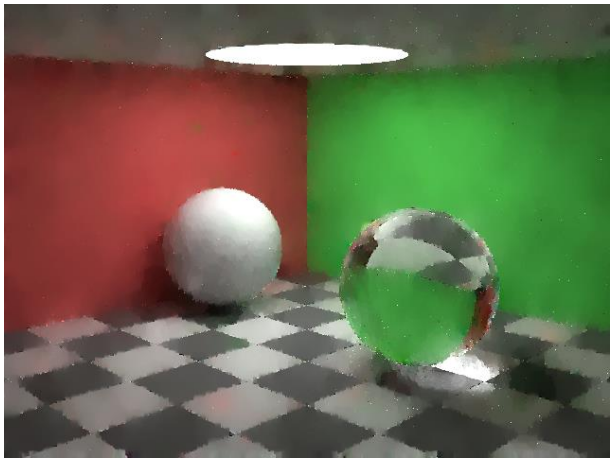
4サンプル/ピクセル



パストレーシング



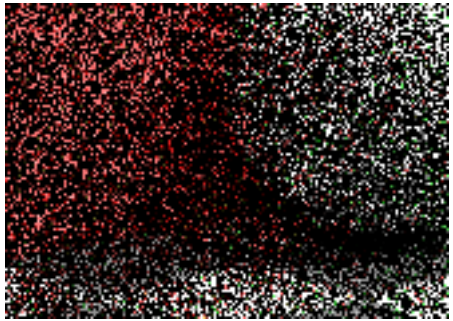
GD-PT (L2)



GD-PT (L1)

テストシーンA

4サンプル/ピクセル



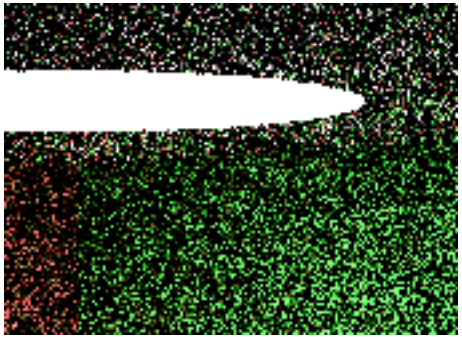
パストレーシング

GD-PT (L2)

GD-PT (L1)

テストシーンA

4サンプル/ピクセル



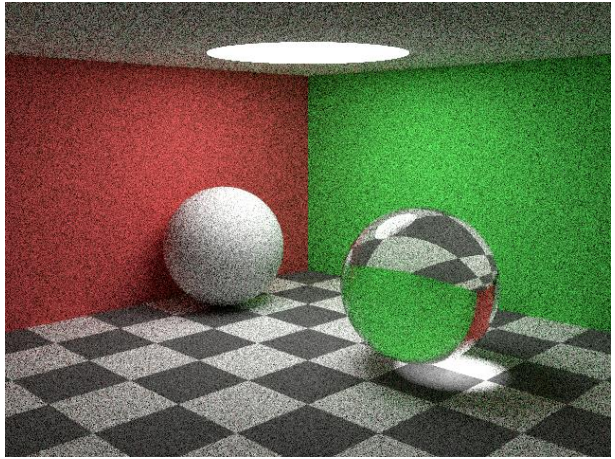
パストレーシング

GD-PT (L2)

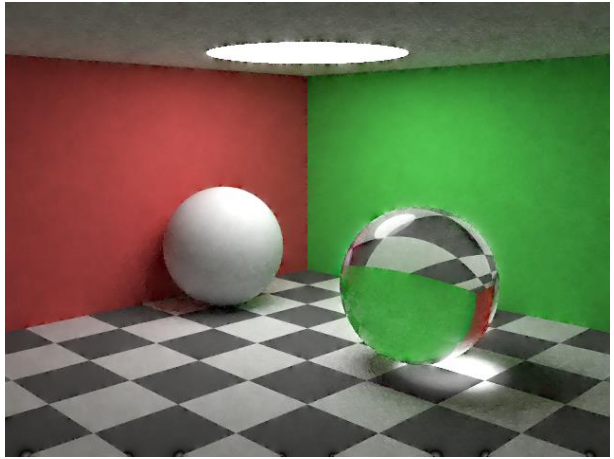
GD-PT (L1)

テストシーンA

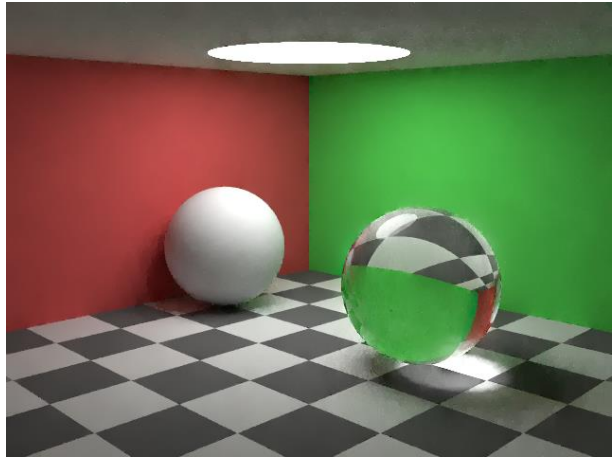
64サンプル/ピクセル



パストレーシング



GD-PT (L2)



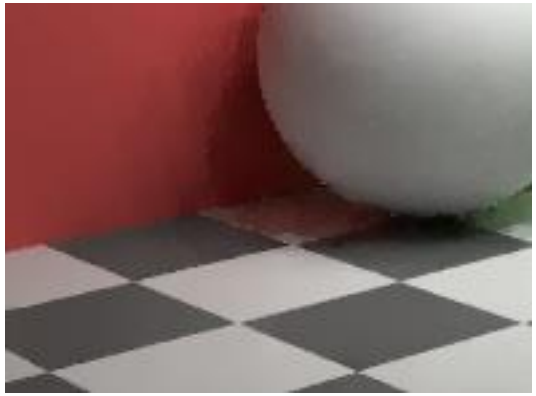
GD-PT (L1)

テストシーンA

64サンプル/ピクセル

パストレーシング

GD-PT (L2)



GD-PT (L1)

テストシーンA

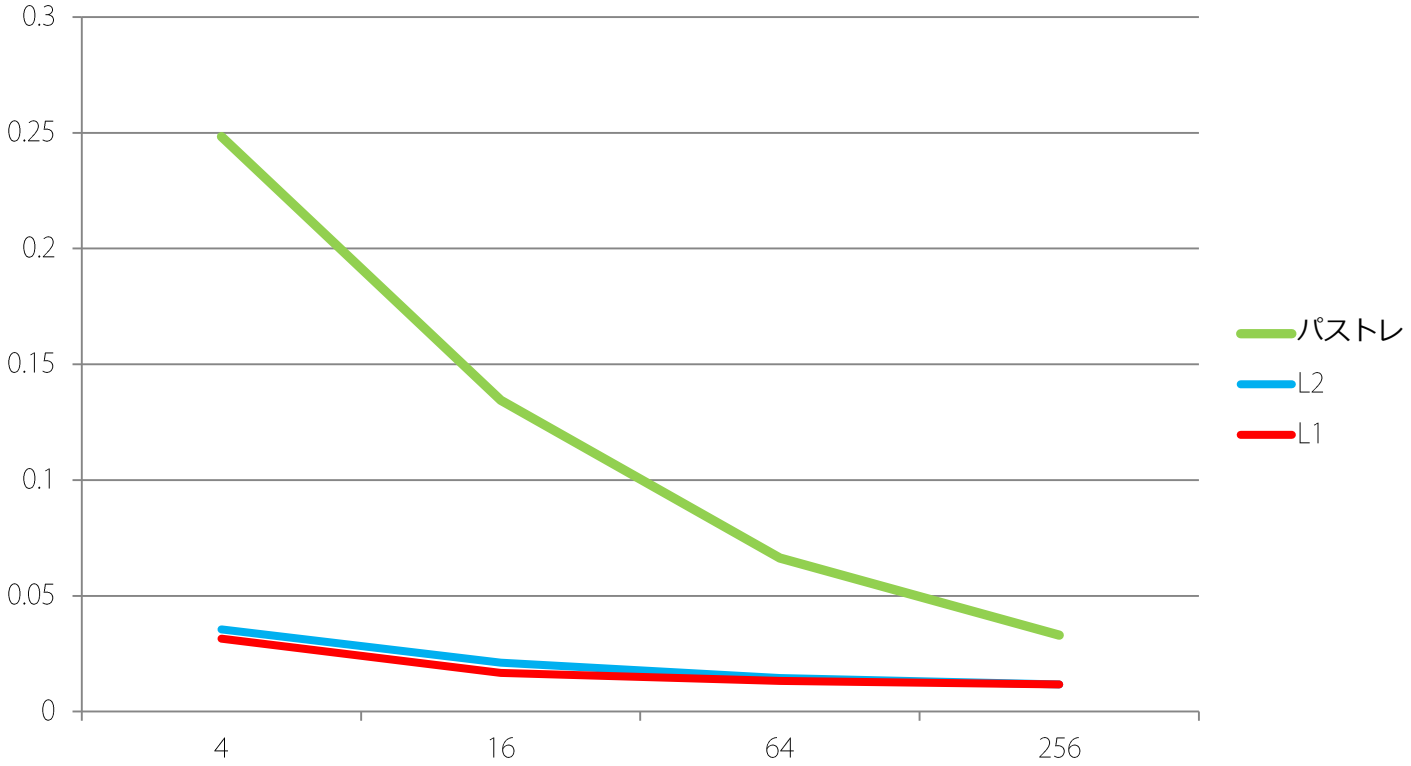
64サンプル/ピクセル

パストレーシング

GD-PT (L2)

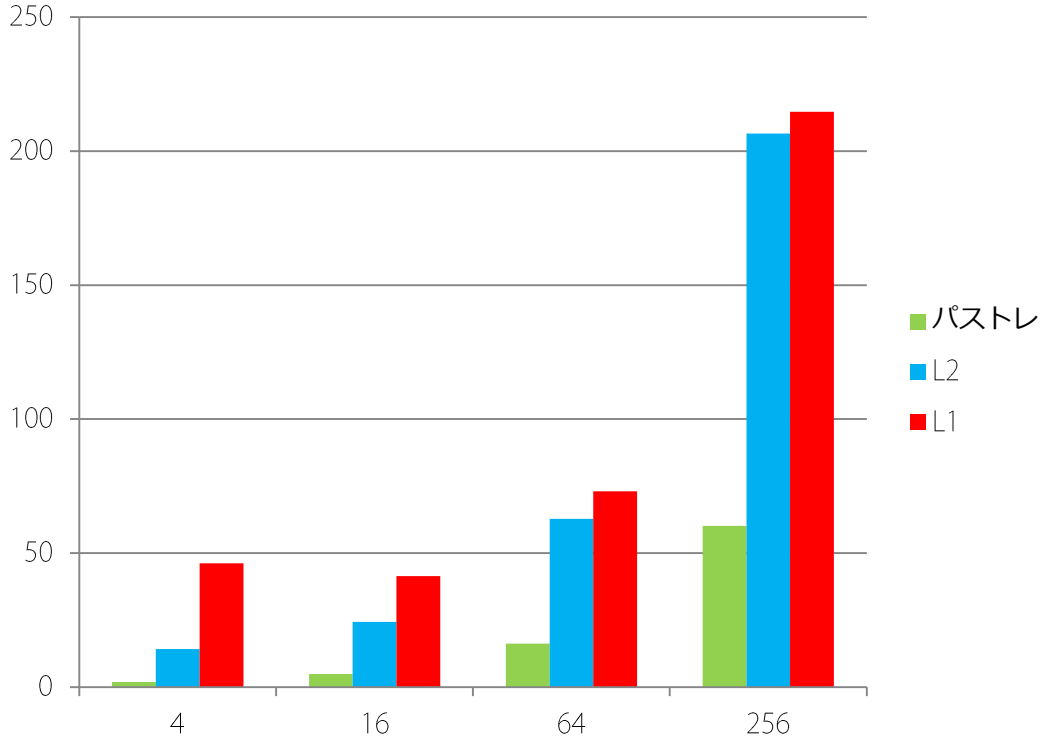
GD-PT (L1)

テストシーンA



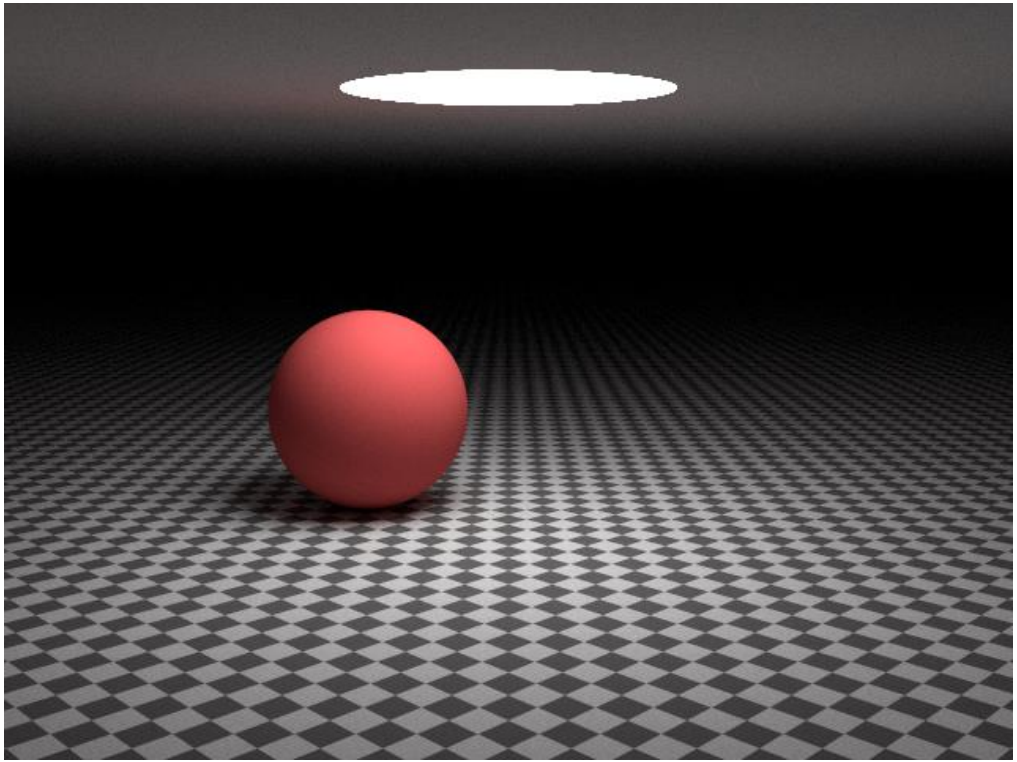
リファレンス画像に対するRMSE

テストシーンA



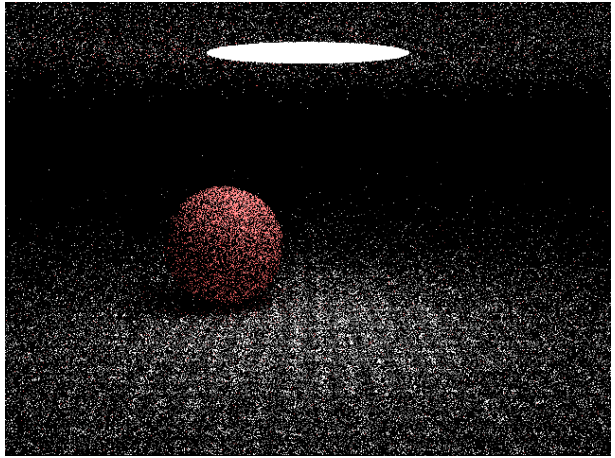
総レンダリング時間（秒）

テストシーンB

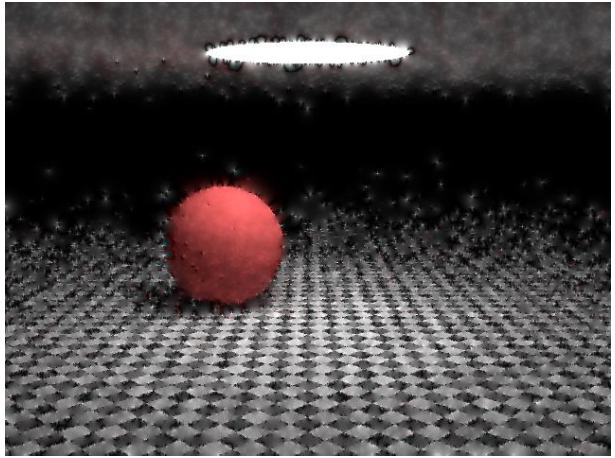


テストシーンB

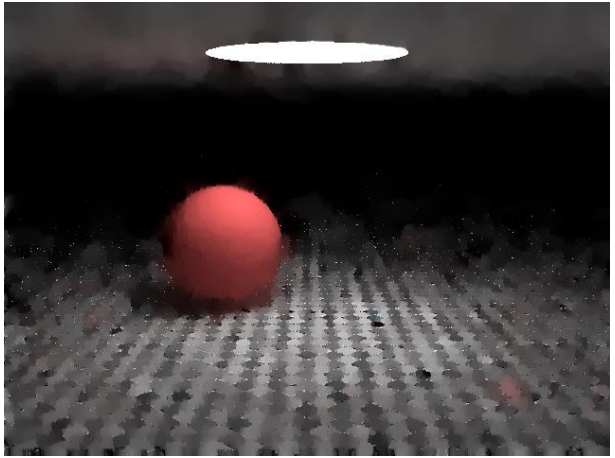
4サンプル/ピクセル



パストレーシング



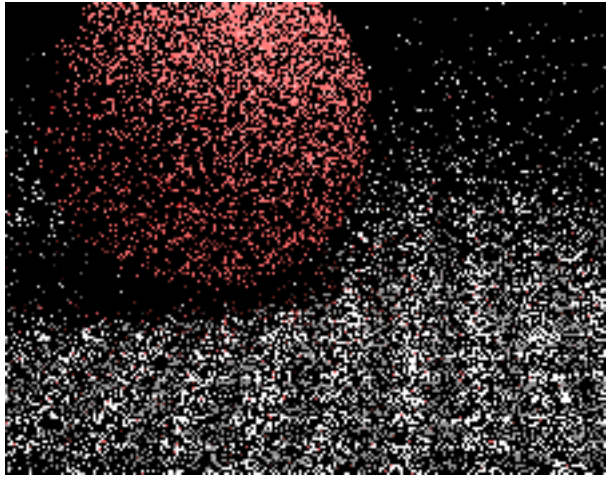
GD-PT (L2)



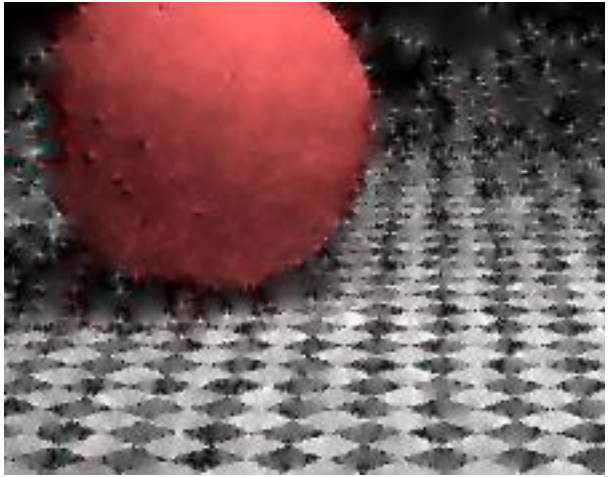
GD-PT (L1)

テストシーンB

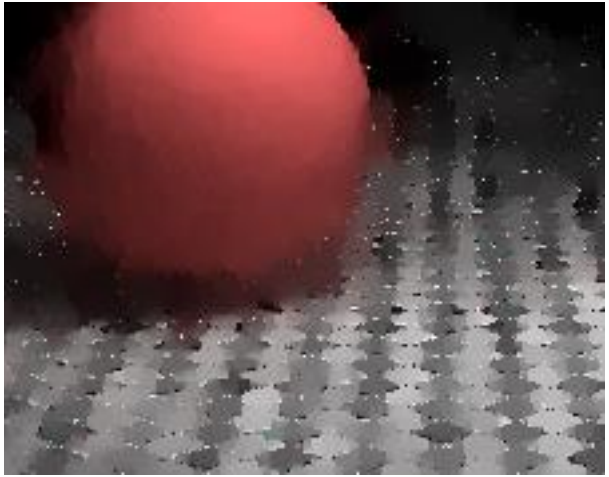
4サンプル/ピクセル



パストレーシング



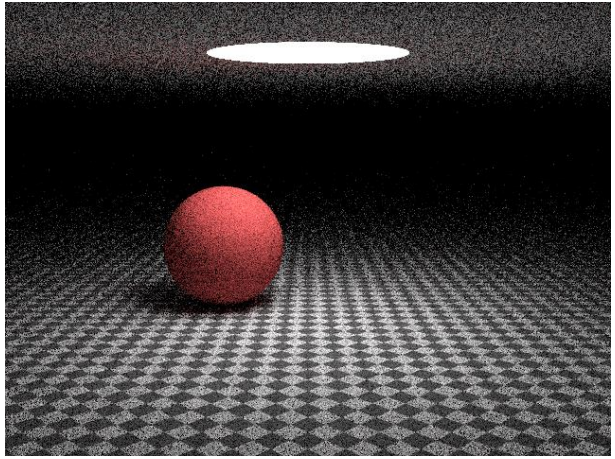
GD-PT (L2)



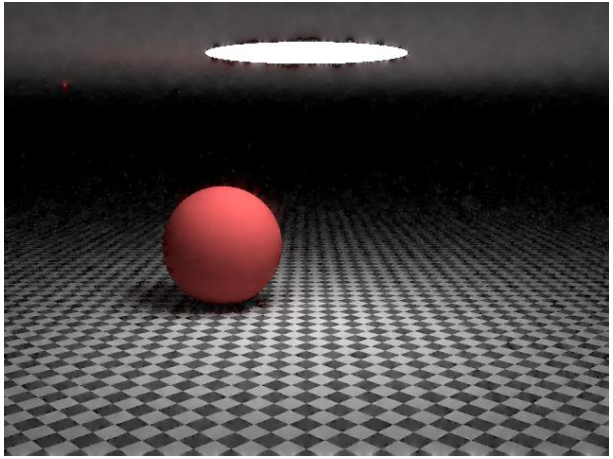
GD-PT (L1)

テストシーンB

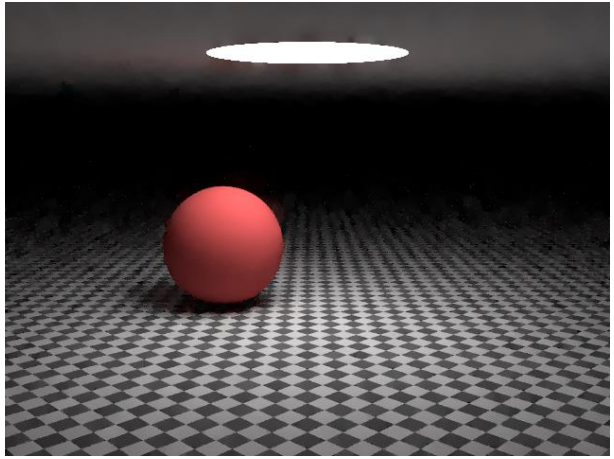
64サンプル/ピクセル



パストレーシング



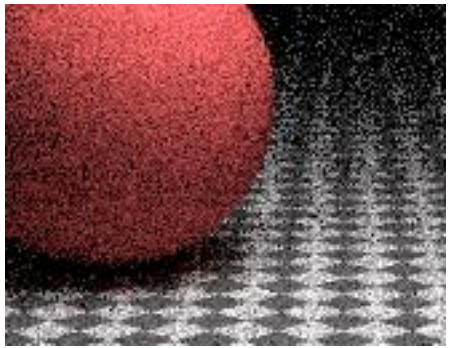
GD-PT (L2)



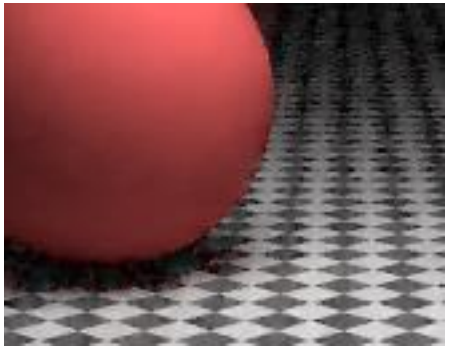
GD-PT (L1)

テストシーンB

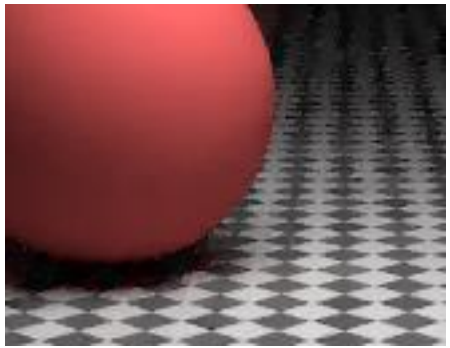
64サンプル/ピクセル



パストレーシング

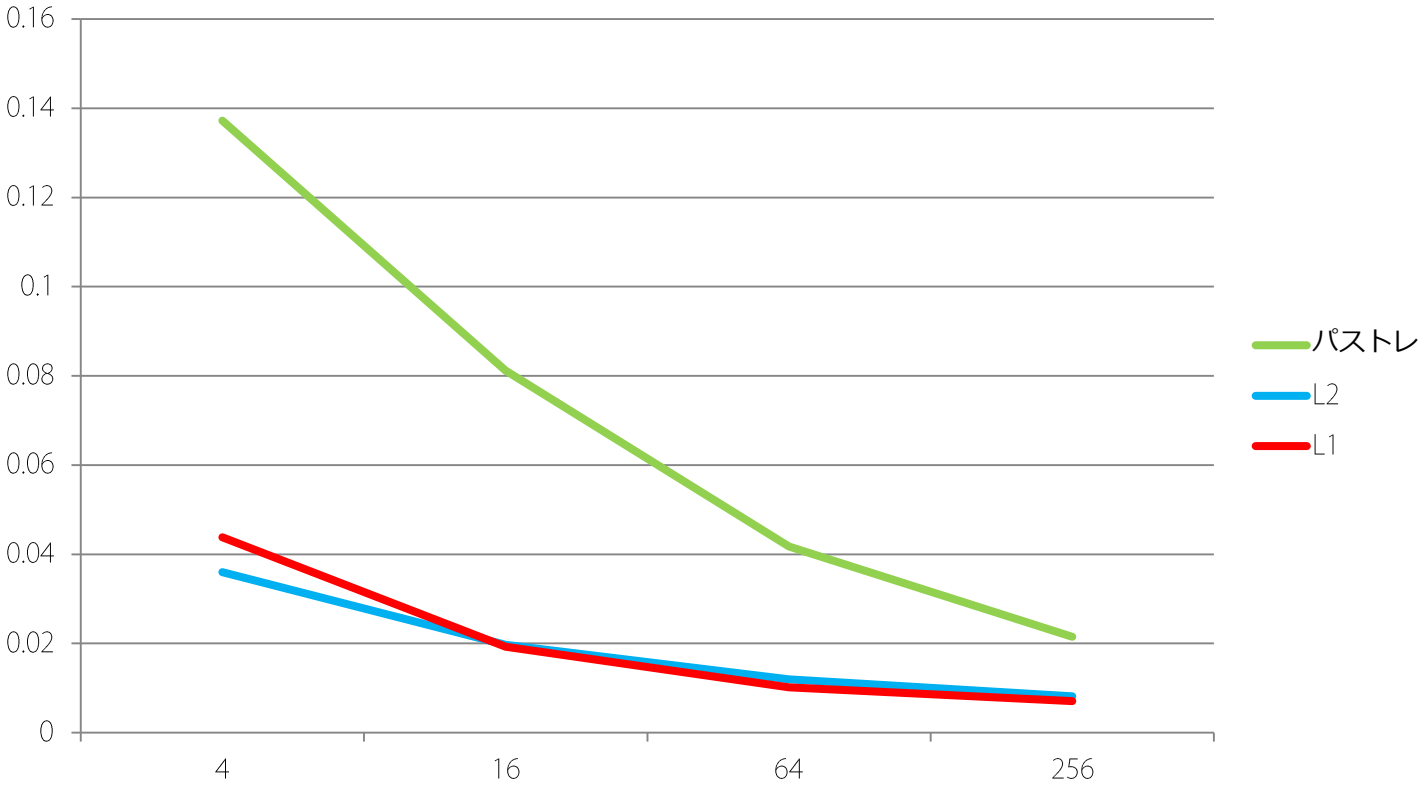


GD-PT (L2)



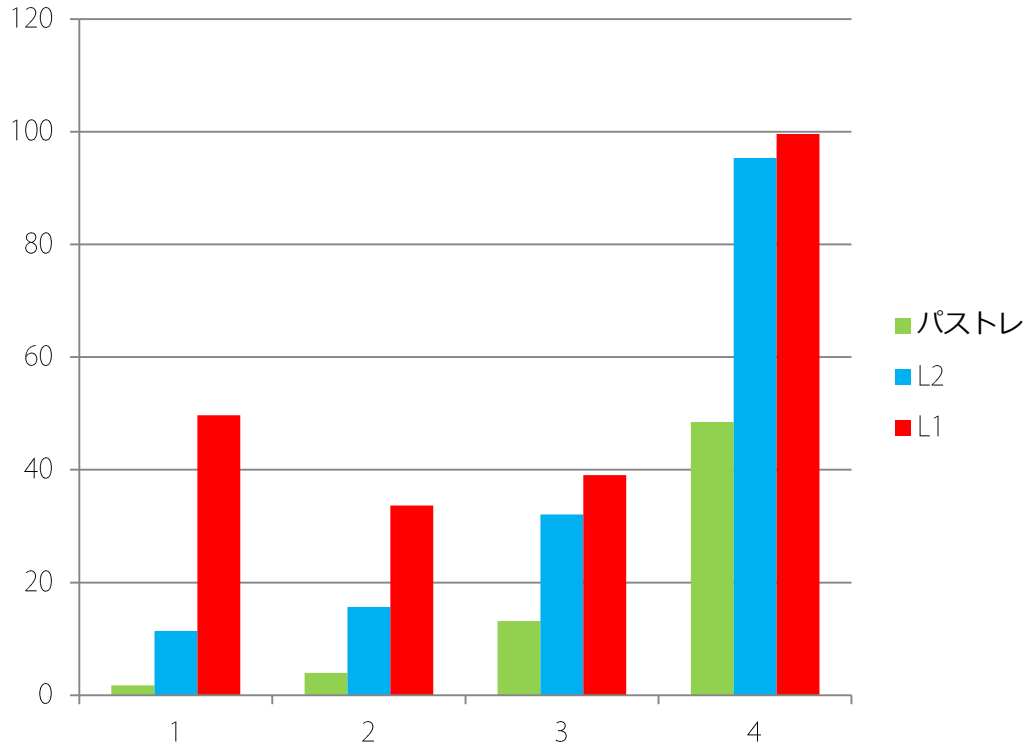
GD-PT (L1)

テストシーンB



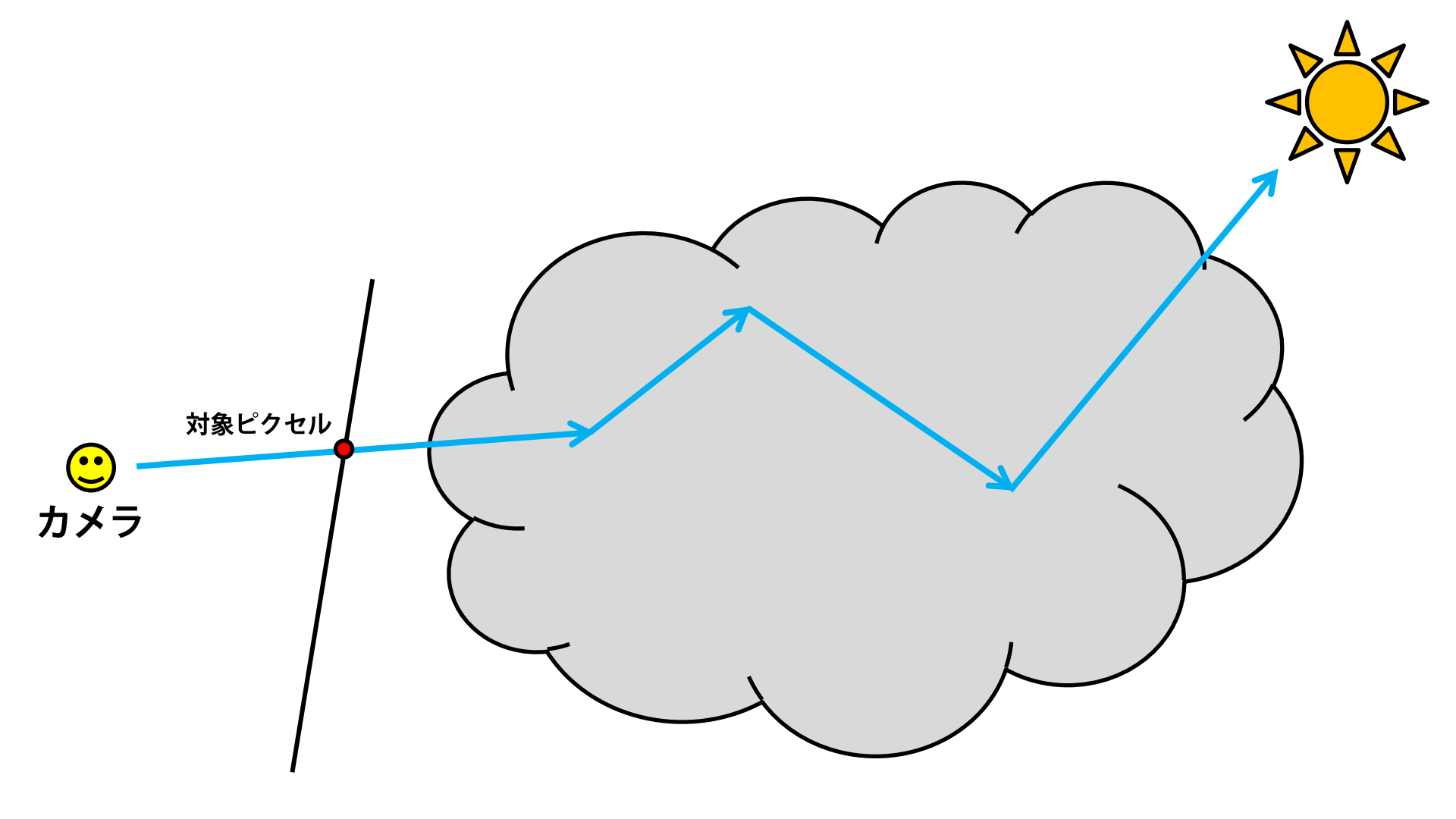
リファレンス画像に対するRMSE

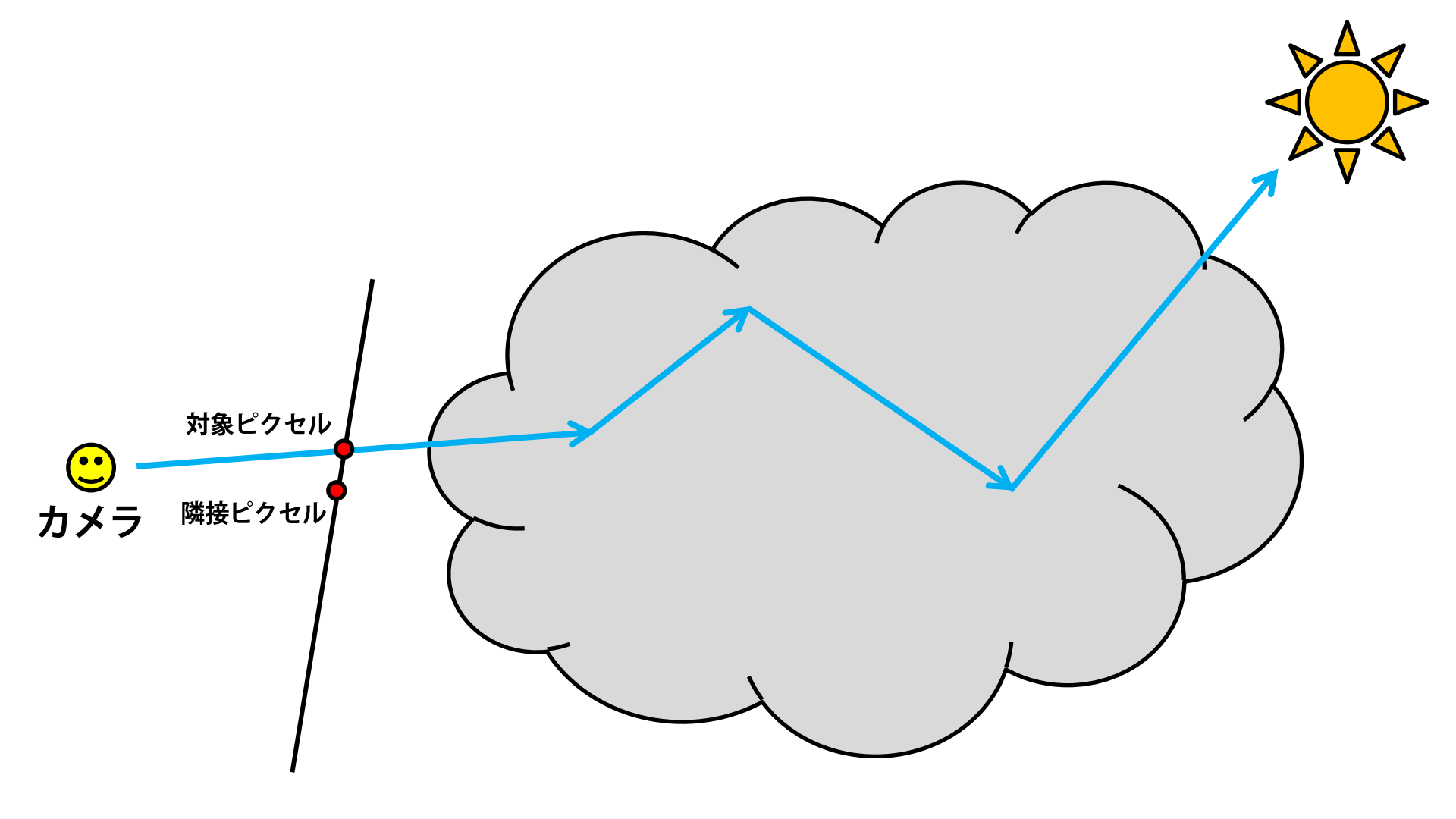
テストシーンB

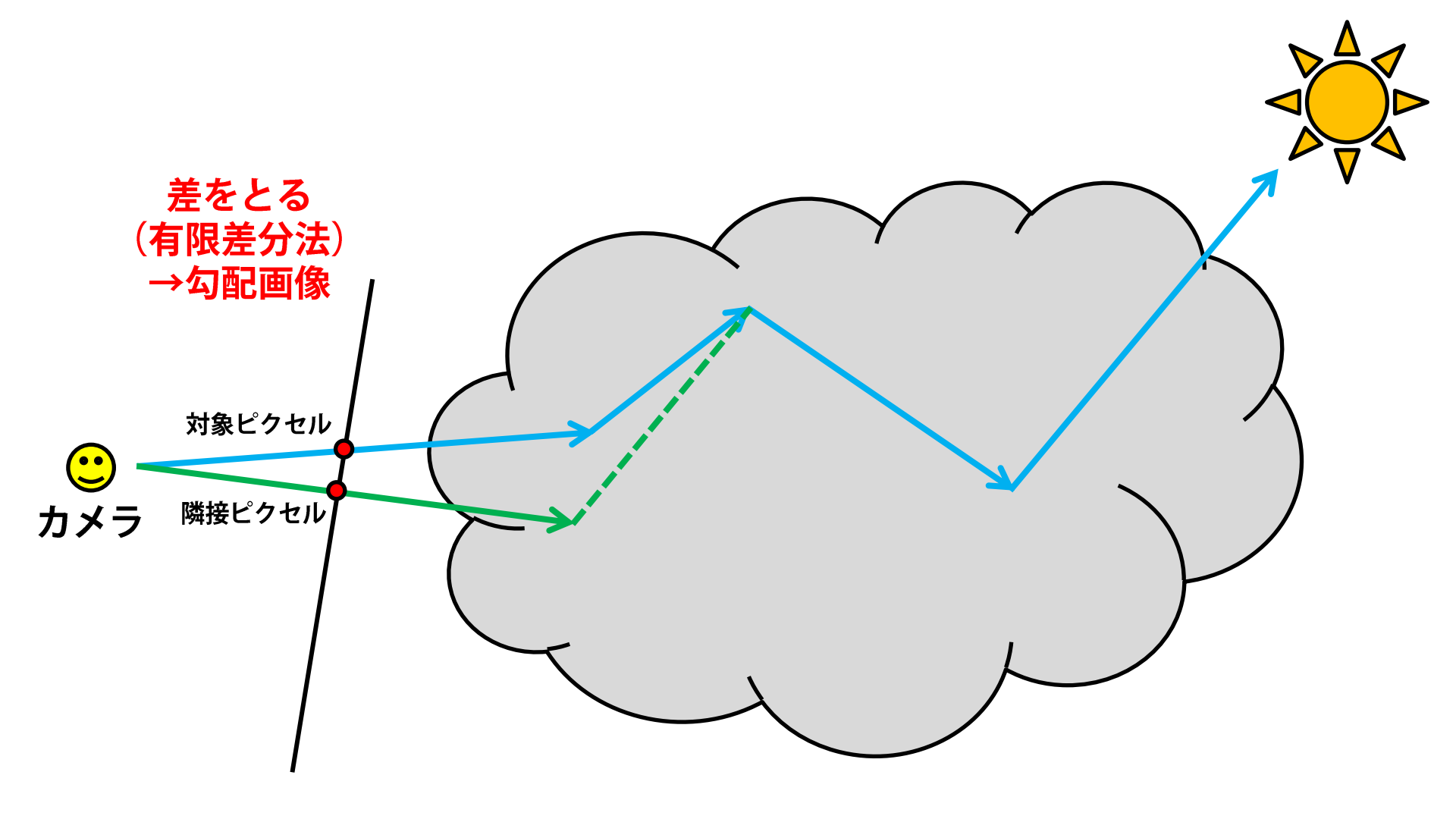


総レンダリング時間（秒）

ボリュームレンダリング


対象ピクセル
カメラ

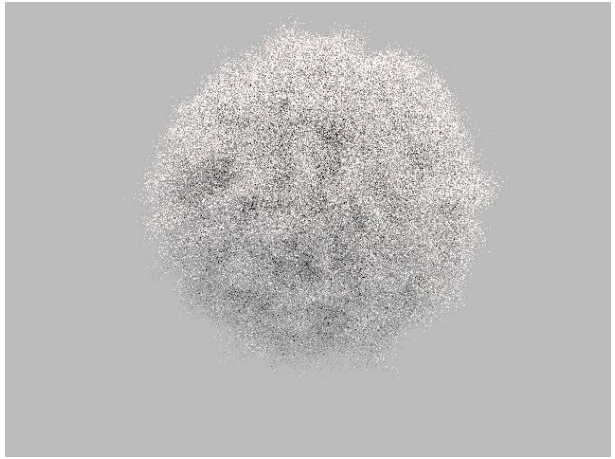

対象ピクセル
隣接ピクセル
カメラ


差をとる
（有限差分法）
→勾配画像
対象ピクセル
隣接ピクセル
カメラ

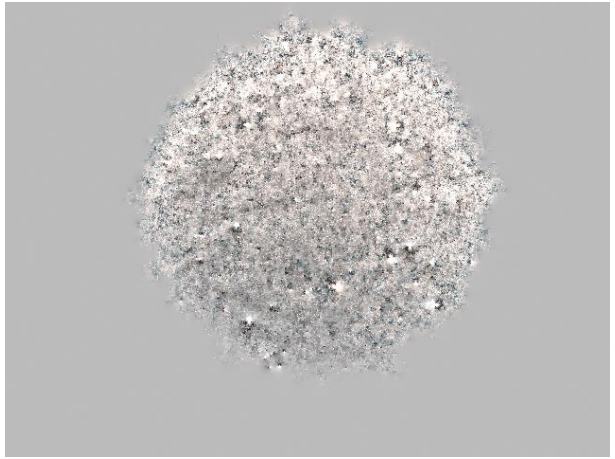
テストシーンC

テストシーンB

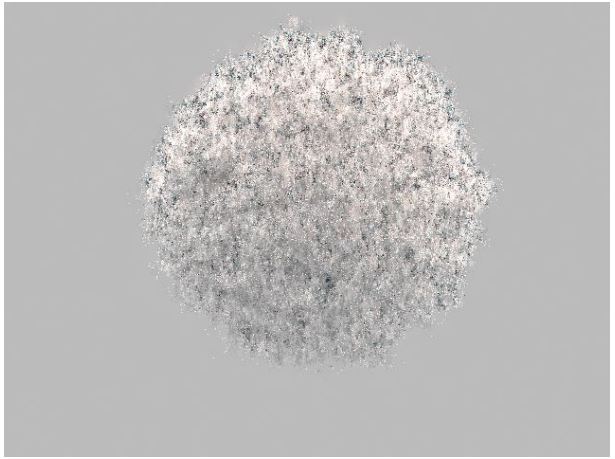
4サンプル/ピクセル



パストレーシング



GD-PT (L2)

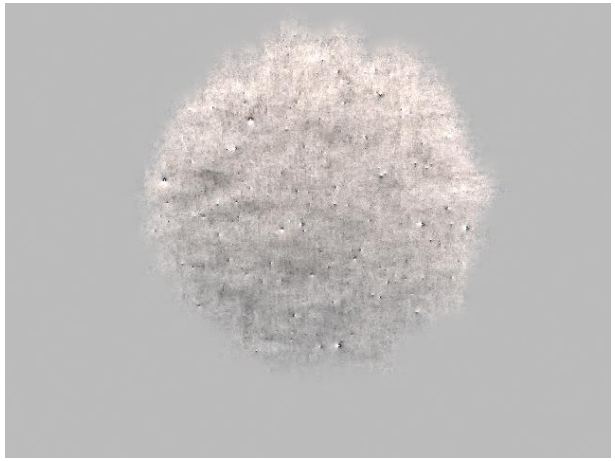


GD-PT (L1)

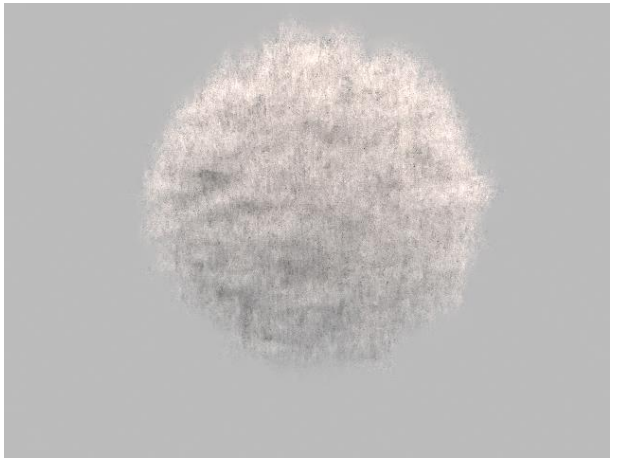
テストシーンB

64サンプル/ピクセル

パストレーシング



GD-PT (L2)



GD-PT (L1)

なんかバグってる

まとめ

まとめ

- **Gradient Domain Path Tracing**
  - 実装は簡単…といえば簡単だけどハマりどころも多い。
    - Diffuseだけなら良いが、いつも通りSpecularが絡むと大変。
  - 結果は結構良い。コストも低め。
    - パストレ比、2~3倍時間が増えてノイズが消える。

- **ボリュームへの拡張**
  - 良いShift戦略を作るのは結構大変かも・・・？
  - 低周波ほど不利になる手法なのでそもそも相性が悪いのかもしれない。
  - バグを修正したい。