Extending Differentiable Rendering Towards Participating Media

Master's Thesis Presentation

Tim Stullich

Monday 15th June, 2020

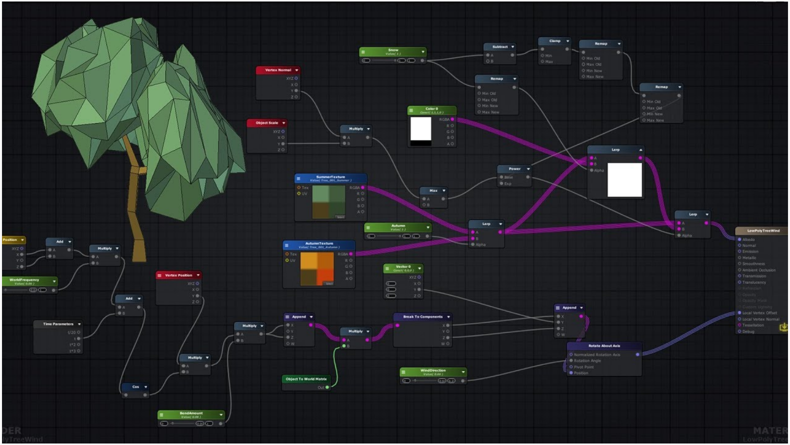
Technical University Berlin - Department of Computer Engineering and Microelectronics

Table Of Contents

1. Introduction
2. Differentiable Rendering
3. Volumetric Scattering
4. Code Implementation
5. Evaluation
6. Conclusion

Introduction

Motivation



Shader node graph for a low polycount tree [Imp18]

Motivation

- Modern shader code is becoming increasingly more complex
- Negatively impacts productivity of 3D application users
- Currently there are no known solutions to this problem
- Can differentiable rendering assist this use case?

Thesis Overview

Thesis Goal

Extend and evaluate differentiable rendering by integrating homogeneous participating media into a differentiable renderer

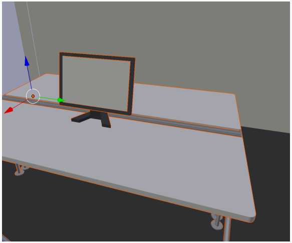
This was accomplished by:

1. Researching differentiable rendering to find a suitable renderer
2. Extending *redner's* [Li+18] functionality to render homogeneous participating media
3. Evaluating how the code implementation solves the shader graph problem by running a number of experiments

Differentiable Rendering

What Is Differentiable Rendering?

3D Scene



Rendering F(π)

Inverse?

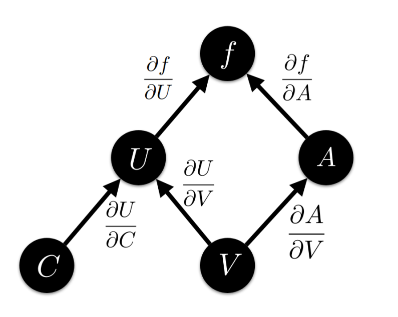
Image

Pixel Color (I)

Parameters (π)
Camera Pose
Geometry
Materials
Lights

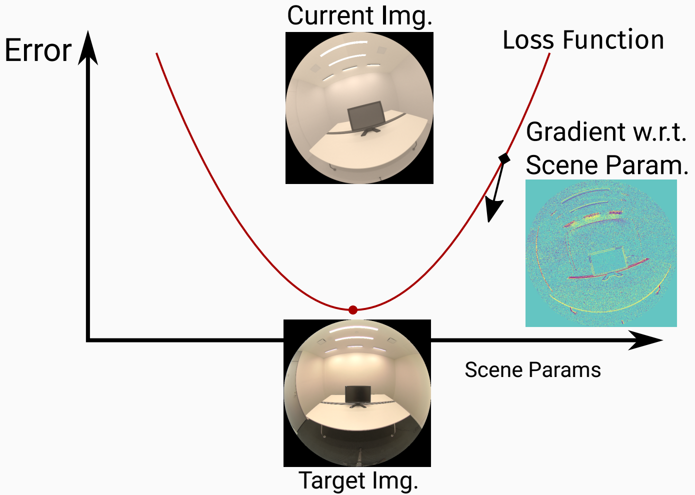
Backpropagation

- Machine learning algorithm to train neural networks
- Uses *gradient vector* ∇ to minimize a loss function
  - $L = \arg\min_{\pi} \sum_{i=0}^{I} (T_i - C_i)^2$
  - Gradient Descent
- Propagates loss to inputs through the *Chain Rule*
- Allows for adjusting of scene parameters according to the pixel color



Gradient vector of f [LB14]

Gradient Descent



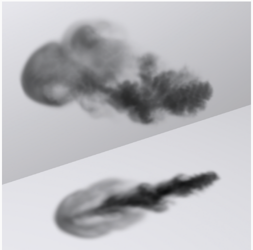
Volumetric Scattering

Volumetric Scattering Terminology

- Occurs when light interacts with particles outside of a vacuum
- Photons collide with particles which are *absorbed*, *scattered* or *emitted*
- Bounded volume is called a *participating medium*
- Homogeneous vs. heterogeneous medium

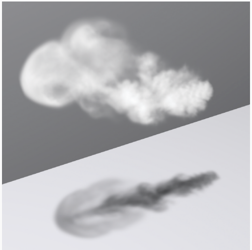
”God rays” shining through fog. [11]

Volumetric Scattering Processes



Absorption ($\boldsymbol{\sigma_a}$)

Scattering ($\boldsymbol{\sigma_s}$)



Emission

Notable properties

- *Absorption coefficient* ($\boldsymbol{\sigma_a}$), *scattering coefficient* ($\boldsymbol{\sigma_s}$) characterize reduction of energy inside medium
- *Attenuation coefficient*: $\boldsymbol{\sigma_t} = \sigma_a + \sigma_s$
  - Overall attenuation: $\frac{\mathrm{d}L_o(p,\omega)}{\mathrm{d}t} = -\sigma_t(p,\omega)L_i(p,-\omega)$
  - Equation can be solved to find the *beam transmittance*

Beam Transmittance

The fraction of radiance transmitted between two points is given by

$$T_r(p \to p') = e^{-\int_0^d \boldsymbol{\sigma}_t(p+t\omega,\omega)\mathrm{d}t}$$

Notes

- The negative exponent is the *optical density* τ
- In a homogeneous medium $\boldsymbol{\sigma}_t$ is constant and the transmittance can be calculated using the Beer-Lambert Law: $\tau = e^{-\sigma_t d}$
- Simpler implementation and better performance when compared to heterogeneous media

Radiative Transfer Equation

$$L_i(p,\omega) = \underbrace{T_r(p_0 \to p)L_o(p_0,-\omega)}_{\text{Surface Term}} + \int_0^{t_{max}} \underbrace{T_r(p+t\omega \to p)L_s(p+t\omega,-\omega)\mathrm{d}t}_{\text{Medium Term}}$$

Overview of the Radiative Transfer Equation [Cha05]

- The RTE describes energy conservation within a medium
- Required for physically-based rendering (Light Transport)
- Not possible to solve analytically
- Solution can be estimated using Monte Carlo (MC) integration

Code Implementation

Redner Overview & Changes

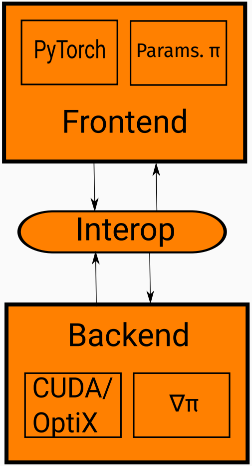
Architecture is split into two parts:

Frontend

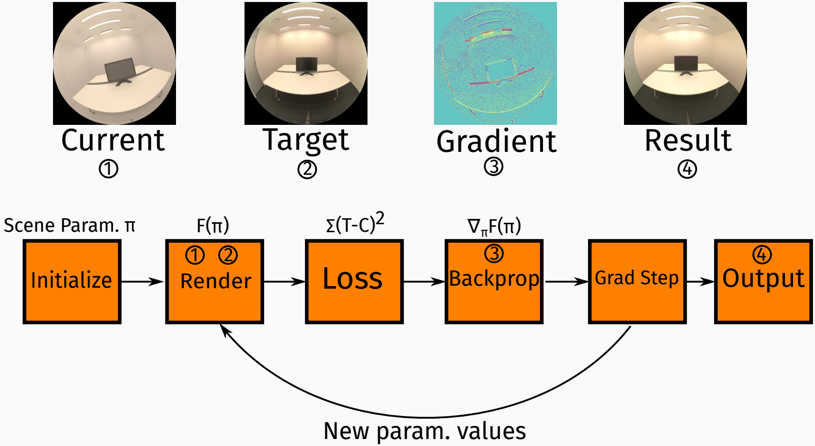
1. Added medium-related Python classes
   - σ_a and σ_s as scene parameters
   - Updated interop layer
2. Wrote test scripts to verify overall implementation

Backend

1. Integrated path tracing logic for homogeneous media
2. Expanded gradient calculation code



Redner's Pipeline

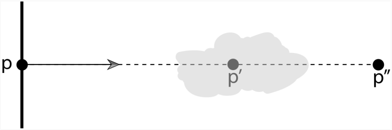


Evaluating Transmittance

$$L_i(p,\omega) = \underbrace{T_r(p_0 \to p)L_o(p_0, -\omega)}_{\text{Surface Term}} + \int_0^{t_{max}} \underbrace{T_r(p + t\omega \to p)L_s(p + t\omega, -\omega)\mathrm{d}t}_{\text{Medium Term}}$$

MC estimation of transmittance:

1. Sample a distance t along ray $f(t) = -\frac{ln(1-\xi)}{\sigma_t}$
2. if $t < t_{max}$: Return beam transmittance
3. else: Not in medium
4. Use transmittance to evaluate surface or medium term



Beam transmittance between two points [PJH16]

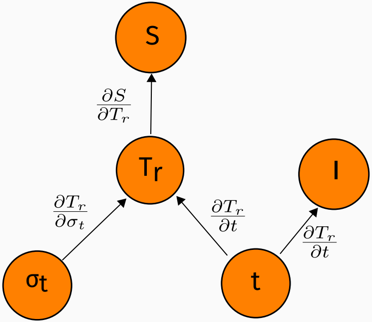
Transmittance Code (Forward Pass)

```
Vector3 transmittance(const Medium &medium, const Ray &ray) {
  if (medium.type == MediumType::homogeneous) {
    // Use Beer-Lambert Law to calculate transmittance
    if (ray.tmax < MaxFloat) {
      return exp(-medium.sigma_t * ray.tmax);
    } else {
        return Vector3{0, 0, 0};
    }
  } else {
    return Vector3{0, 0, 0};
  }
}
```

Transmittance code from redner's codebase [19b]

Deriving Transmittance Code

- Combine derivatives using the Chain Rule
- Transmittance needs derivatives for two parameters (σ_t and t)
- Transmittance propagates to RTE sampling function (S)
- $\frac{\partial T_r}{\partial t}$ also propagates to ray intersection function (I)



Partial derivatives of transmittance() [LB14]

Transmittance Code (Backward Pass)

```
void d_transmittance(const Medium &medium, const Ray &ray,
                     const Vector3 &d_output, DMedium &d_medium,
                     DRay &d_ray) {
  if (medium.type == MediumType::homogeneous) {
    if (ray.tmax < MaxFloat) {
      auto output = exp(-medium.sigma_t * ray.tmax);
      auto d_sigma_t = -d_output * output * ray.tmax;
      d_ray.tmax += -sum(d_output * output * medium.sigma_t);
      // sigma_t = sigma_a + sigma_s;
      atomic_add(&d_medium.sigma_a[0], d_sigma_t[0]);
      atomic_add(&d_medium.sigma_a[1], d_sigma_t[1]);
      atomic_add(&d_medium.sigma_a[2], d_sigma_t[2]);
      atomic_add(&d_medium.sigma_s[0], d_sigma_t[0]);
      atomic_add(&d_medium.sigma_s[1], d_sigma_t[1]);
      atomic_add(&d_medium.sigma_s[2], d_sigma_t[2]);
    }
  }
}
```

Evaluation

Evaluating The Implementation

The method used for evaluating the implementation:

1. Create multiple scenes containing one or more homogeneous media
2. Perturb medium parameter(s) and render target and perturbed scene
3. Run gradient for set amount of iterations optimizing medium parameter(s)
4. Measure loss per iteration $L_i = \sum_{i=0}^{I} (T_i - P_i)^2$
   - Record per-pixel difference as an image
   - Low loss should result in a black image
5. Compare optimized parameters to target parameters

San Miguel Scene

- Chosen because of its complexity
- Tested absorption and scattering coefficients
- Gradient descent optimization of two parameters (σ_a, σ_s) and one homogeneous medium covering the whole scene

San Miguel Test - Two Parameters (σ_a and σ_s)

medium	σ_a			σ_s		
target	0.0001	0.0001	0.0001	0.001	0.001	0.001
perturbed	0.0009	0.0009	0.0009	0.2	0.2	0.2

Perturbed Image

Per-pixel Difference

Target Image

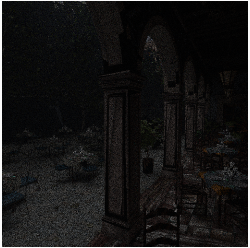
The goal is to remove the fog from the perturbed image by adjusting the input parameters through gradient descent

San Miguel Test - Result

Recovered values for σ_a and σ_s after gradient descent optimization

medium	σ_a			σ_s		
perturbed	0.0001	0.0001	0.0001	0.0022	0.0024	0.0026
target	0.0001	0.0001	0.0001	0.001	0.001	0.001

Perturbed Image

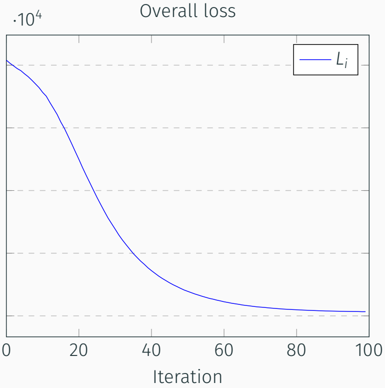


Per-pixel Difference

Target Image

Verifies that the implementation works within a reasonable degree

San Miguel - Measured Loss



The reduction of the loss graphed over 100 iterations of gradient descent

Conclusion

Summary

1. Introduced the theory behind differentiable rendering & volumetric scattering
2. Gained an understanding of the architecture of a differentiable renderer
3. Viewed code samples that demonstrated how to implement parts of the RTE
4. Verified that homogeneous participating media can be differentiably rendered

Conclusion I

Is differentiable rendering useful for the original problem case?

- Differentiable rendering has potential use for assisting in the modeling workflow of 3D application users
- Performance improvements still need to be made
  - San Miguel scene takes minutes for one iteration of rendering + backpropagation
  - An experienced user can most likely manually adjust parameters faster
- For now better suited for offline tasks (shape reconstruction, style transfer, etc.) [Liu+19]

Conclusion II

Potential enhancements:

- Investigate San Miguel test scene data more in depth
  - Perceived differences versus actual parameters
- Expand to heterogeneous media
- Account for discontinuities in Radiative Transfer Equation
- Write a plugin for open-source renderers (Cycles [Fou20], Appleseed [19a], etc.)

Questions?

Notable Differentiable Renderers

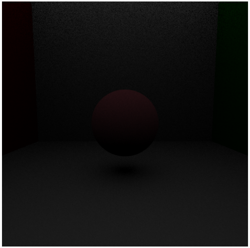
Renderer	Pros	Cons
OpenDR	Easy to use Leverages OpenCV	Approximates gradients Not physically-based
SoftRas	Operates on probability maps	Different use case Not physically-based
Mitsuba 2	Flexible architecture Volumetric scattering	Late release
Redner	Physically-based Mature codebase	No volumetric scattering

Table 1: Redner was chosen in part due to these aspects

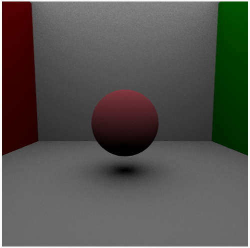
Cornell Box Test - Absorption Coefficient

Gradient Descent Parameters: 100 iterations & learning rate = 0.005

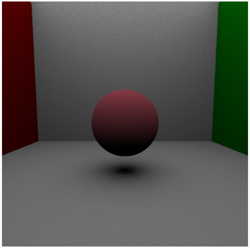
medium	σ_a		
perturbed	0.3	0.3	0.3
target	0.05	0.05	0.05



Perturbed Image



Per-pixel Difference

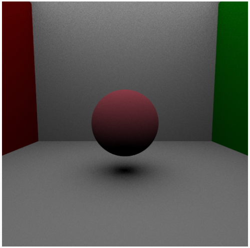


Target Image

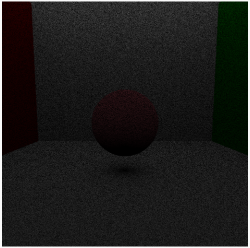
Cornell Box Test - Results

Recovered values for σ_a after gradient descent optimization

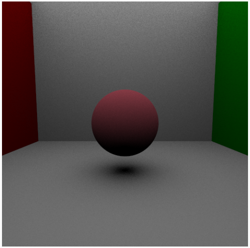
medium	σ_a		
target	0.05	0.05	0.05
final	0.0471	0.0471	0.0471



Final Image

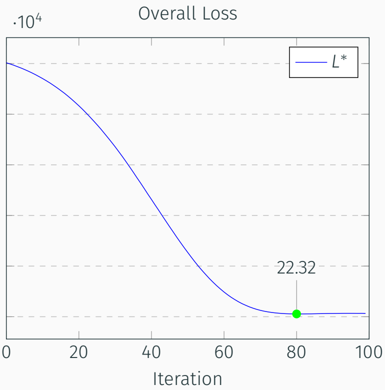


Per-pixel Difference



Target Image

Cornell Box - Measured Loss



The loss steadily declines until a local minimum is reached after 80 iterations.

Surface & Medium Term

$$L_i(p,\omega) = \underbrace{T_r(p_0 \to p)L_o(p_0,-\omega)}_{\text{Surface Term = } \beta_{surf}} + \int_0^{t_{max}} \underbrace{T_r(p+t\omega \to p)L_s(p+t\omega,-\omega)\mathrm{d}t}_{\text{Medium Term = } \beta_{med}}$$

Monte Carlo Estimators

The respective estimators for β_{surf} and β_{med}:

- $\beta_{surf} = \frac{T_r(p \longrightarrow p+t\omega)}{p_{surf}}$
- $p_{surf} = 1 - \int_0^{t_{max}} p_t(t)\mathrm{d}t$
- $\beta_{med} = \frac{\sigma_s(p+t\omega)T_r(p \longrightarrow p+t\omega)}{p_t(t)}$
- PDF: $p_t(t) = \sigma_t e^{-\sigma_t t}$

Radiative Transfer Equation

$$L_i(p,\omega) = \underbrace{T_r(p_0 \to p)L_o(p_0,-\omega)}_{\text{Surface Term}} + \int_0^{t_{max}} \underbrace{T_r(p+t\omega \to p)L_s(p+t\omega,-\omega)\mathrm{d}t}_{\text{Medium Term}}$$

Terms of the RTE

- Two distinct terms to which MC can be applied
- Only one of the term needs to be estimated at a time

Radiative Transfer Equation

$$L_i(p,\omega) = \underbrace{T_r(p_0 \to p)}_{\text{Transmittance}} L_o(p_0,-\omega) + \int_0^{t_{max}} \underbrace{T_r(p+t\omega \to p)}_{\text{Transmittance}} L_s(p+t\omega,-\omega)\mathrm{d}t$$

Transmittance

- The amount of light that passes through the medium after accounting for absorption, scattering, and emission. [07]
- Will be examined more in-depth later

Radiative Transfer Equation

$$L_i(p,\omega) = T_r(p_0 \rightarrow p)\ \underbrace{L_o(p_0, -\omega)}_{\text{Exitant Radiance}} + \int_0^{t_{max}} T_r(p+t\omega \rightarrow p) L_s(p+t\omega, -\omega)\mathrm{d}t$$

Exitant radiance

- Describes the amount of light that is emitted at the surface of the medium

Radiative Transfer Equation

$$L_i(p,\omega) = T_r(p_0 \to p)L_o(p_0,-\omega) + \int_0^{t_{max}} T_r(p+t\omega \to p)\,\underbrace{L_s(p+t\omega,-\omega)}_{\text{Source Term}}\,\mathrm{d}t$$

Source term

- Accounts for scattering inside the medium
- Scattering direction determined by a *phase function*. [HG41]
- Phase function be thought of as analog to a BSDF

References

transmittance. Oct. 2007. URL: *https://en.wikipedia.org/wiki/Transmittance* (visited on 06/04/2020).

God rays 8 by hanpanman on DeviantArt. Aug. 2011. URL: *https://www.deviantart.com/hanpanman/art/God-rays-8-255417166* (visited on 06/02/2020).

A modern, open source production renderer. Sept. 2019. URL: *https://appleseedhq.net/*.

tstullich/redner - A differentiable Monte Carlo path tracer. 2019. URL: *https://github.com/tstullich/redner* (visited on 06/02/2020).

Subrahmanyan Chandrasekhar. *Radiative Transfer.* Dover Publications, 2005.

Blender Foundation. *blender.org - Home of the Blender project - Free and Open 3D Creation Software.* Feb. 2020. URL: *https://blender.org* (visited on 02/28/2020).

L. G. Henyey and J. L. Greenstein. "Diffuse radiation in the Galaxy.". In: *ApJ* 93 (Jan. 1941), pp. 70–83. DOI: *10.1086/144246*.

Imphenzia. *Lowpoly Tree Shader in Amplify Shader Editor for Unity*. Youtube. July 4, 2018. URL: https://www.youtube.com/watch?v=o0ZlwiXP48I (visited on 06/02/2020).

Matthew Loper and Michael Black. "OpenDR: An Approximate Differentiable Renderer". In: Sept. 2014. DOI: 10.1007/978-3-319-10584-0_11.

Tzu-Mao Li et al. "Differentiable Monte Carlo Ray Tracing through Edge Sampling". In: *ACM Trans. Graph.* 37.6 (Dec. 2018). ISSN: 0730-0301. DOI: 10.1145/3272127.3275109. URL: https://doi.org/10.1145/3272127.3275109.

Shichen Liu et al. "Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning". In: *The IEEE International Conference on Computer Vision (ICCV)* (Oct. 2019).

Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation (3rd ed.)* 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Nov. 2016, p. 1266. ISBN: 9780128006450.