Differentiable Rendering for Local Parameters

YANBO XU, Carnegie Mellon University, USA

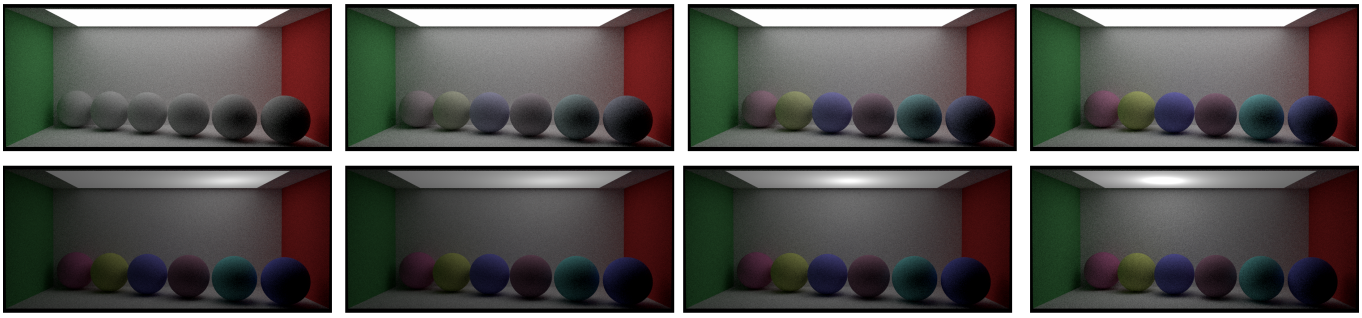

Fig. 1. Optimization using Differentiable Rendering for local parameters. First row: optimize material albedo. Second row: optimize light center.

Physic-based rendering [2] typically refers to the process that outputs an image given the known properties in the scene, such as object location, surface shape, material property lighting condition, etc. However, the assumption that all information about the scene is known is not always valid, which gives rise to the field of inverse rendering [5] that infers the scene attributes from image observations. Differentiable Rendering is one of the methods that can achieve such an objective by utilizing gradients of the scene parameters and optimizing them. In this project, a differentiable rendering framework based on DIRT is implemented, with the capabilities of optimizing local parameters such as BRDF and lighting properties. Support of more advanced optimizers such as Adam [4] is also included. Finally, gradient certification via numerical differentiation (finite difference) is also included.

Additional Key Words and Phrases: Physic-based rendering, Differentiable Rendering

1 INTRODUCTION

Physic-based rendering [2] is wildly used in the fields of science, film, and gaming to produce physically accurate images. It typically refers to the process that outputs an image given the known properties in the scene, such as object location, surface shape, material property lighting condition, etc.

However, the assumption that all information about the scene is known is not always valid, which gives rise to the field of inverse rendering [5] that infers the scene attributes from image observations. For example, given image observations from an unknown material, getting the exact material parameters to describe the object is non-trivial.

Fortunately, many parameters in the forward rendering process are differentiable [3], which makes it possible to optimize the unknown parameters using the gradients. Generally, the parameters can be classified into local parameters and global parameters. The local parameters, such as shading term and BRDF, have gradients that are easier to calculate, whereas the global parameters (object geometry, lighting location, etc) need to account for the discontinuity and boundary conditions during differentiation.

In this project, a differentiable rendering framework based on DIRT is implemented. It can optimize local parameters such as BRDF and lighting properties. It supports more advanced optimizers such as Adam [4] to enable a more stable optimization process. Finally, it supports numerical differentiation (finite difference) to certify the correctness of calculated gradients.

Author's Contact Information: Yanbo Xu, yanboxu@andrew.cmu.edu, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

2 BACKGROUND

2.1 Local Parameters Differentiation

The throughput of a path $\bar{x}$, denoted as $f(\bar{x}, \pi)$, can be calculated as:

$$f(\bar{x}, \pi) = \prod_{b=1}^{B} f_s(x_{b-1}, x_b, x_{b+1}; \pi) * G(x_{b-1}, x_b) \tag{1}$$

, where $f_s(x_{b-1}, x_b, x_{b+1}; \pi)$ is the BRDF term with foreshortening terms, and the $G(x_{b-1}, x_b)$ is the geometry term.

Take the derivative of the throughput w.r.t. local parameters, we get:

$$\begin{aligned}\frac{\partial f(\bar{x}, \pi)}{\partial \pi} &= \sum_{b=1}^{B} \frac{\partial f_s}{\partial \pi}(x_{b-1}, x_b, x_{b+1}) \sum_{i=1, i\neq b}^{B} f_s(x_{i-1}, x_i, x_{i+1}) \\ &= \sum_{b=1}^{B} \frac{\partial f_s}{\partial \pi}(x_{b-1}, x_b, x_{b+1}) \frac{f(\bar{x}, \pi)}{f_s(x_{b-1}, x_b, x_{b+1}; \pi)}\end{aligned}$$

.

Define the score [1] for parameter π_i as:

$$s_i(\bar{x}, \pi) = \sum_{b=1}^{B} \frac{\frac{\partial f_s}{\partial \pi}(x_{b-1}, x_b, x_{b+1})}{f_s(x_{b-1}, x_b, x_{b+1}; \pi)} \tag{2}$$

, one can get the gradient of local parameter π_i as:

$$\frac{\partial f(\bar{x}, \pi)}{\partial \pi_i} = s_i(\bar{x}, \pi) f(\bar{x}, \pi) \tag{3}$$

2.2 Gradient under Image Observation

Recall that ray tracing is a Monte Carlo Estimation estimation of the integral $I_j^{real} = \int_P f(\bar{x}; \pi) d\bar{x}$, defined as:

$$I_j = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\bar{x_i}; \pi)}{p(\bar{x_i})}$$

, here j indicates the pixel location in the sensor, and N is the number of paths for each pixel.

We then can calculate the gradient of the rendering output of pixel j w.r.t the local parameter π_k using:

$$\frac{\partial I_j}{\partial \pi_k} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial f(\bar{x_i};\pi)}{\partial \pi_k}\frac{1}{p(\bar{x_i})} \tag{4}$$

$$= \frac{1}{N}\sum_{i=1}^{N} s_k(\bar{x_i},\pi)\frac{f(\bar{x_i},\pi)}{p(\bar{x_i})} \tag{5}$$

Assume that the loss that evaluates the difference between observed image I^t and rendered image I is

$$L = \frac{1}{2}\sum_{j=1}^{H*W}(I_j - I_j^t)^2$$

, then for the objective that minimizes L, we get

$$\frac{\partial L}{\partial \pi_k} = \sum_{j=1}^{H*W}(I_j - I_j^t)\frac{\partial I_j}{\partial \pi_k} \tag{6}$$

.

Note that in practice, dividing the $\frac{\partial L}{\partial \pi_k}$ by $H * W$ will make the algorithm more robust to different resolutions.

3 DIFFERENTIABLE RENDER BASED ON DIRT

The implementation of this project is based on DIRT , where the term $\frac{f(\bar{x},\pi)}{p(\bar{x})}$ is already evaluated inside the integrator. From eq.(4) and eq.(2), we only need to calculate the score $s_k(\bar{x_i},\pi)$ of each path $\bar{x_i}$ and utilize eq.(6) to get the final gradient.

3.1 Optimization of single surface (material)

Figure. 2 shows the successful optimization of Lambertian albedo value on a single surface.

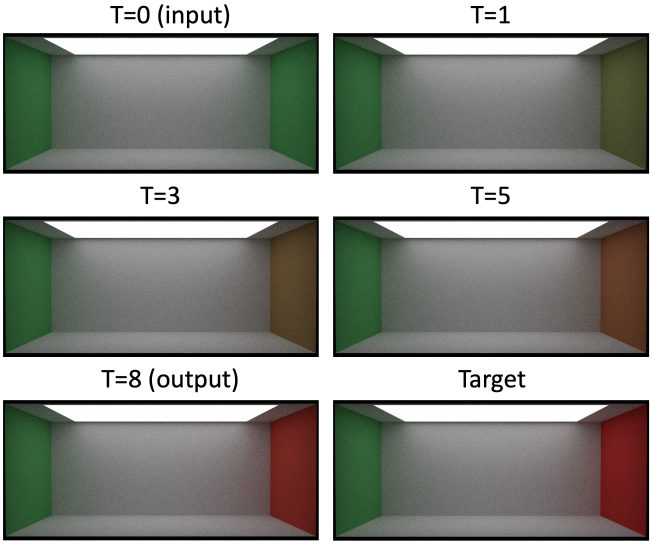

Fig. 2. SGD optimization on Lambertian albedo value, with grad on a single surface. T is the step of the gradient descent process.

Figure. 3 shows the successful optimization of diffuse light emission value on a single surface.

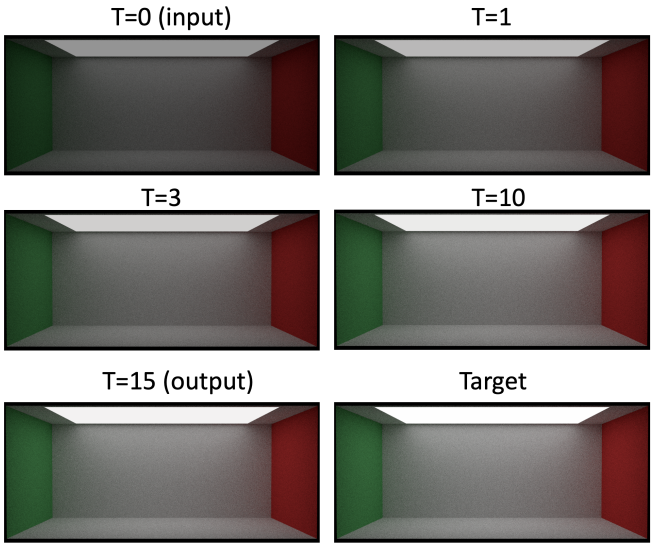

Fig. 3. SGD optimization on diffuse light emission value, with grad on a single surface. T is the step of the gradient descent process.

3.2 Optimization of multiple surfaces (material)

It is possible to optimize multiple surfaces in the scene. To make modifications of materials that can be optimized easier, this framework supports utilizing the original scene JSON file with the property "grad" added to the material JSON config. See Sec. 7 for more detail.

Figure. 4 shows the successful optimization of Lambertian albedo value on a multiple surface.

4 GRADIENT VERIFICATION USING FINITE DIFFERENCE

Although less efficient and potentially noisy, the finite difference (numerical approximation of gradients) could help us verify the correctness of calculated gradients.

In particular, we use gradient images to verify that. The gradient image calculated by our differentiable render can be defined as:

$$I_k^{grad}(h,w) = (I_{h,w}^r(\pi_k) - I_{h,w}^t(\pi_k))\frac{\partial I_{h,w}^r(\pi_k)}{\partial \pi_k}$$

, where hw are the sensor index. The finite difference gradient image is defined as:

$$I_k^{fi}(h,w) = \frac{I_{h,w}^r(\pi_k+\epsilon) - I_{h,w}^r(\pi_k)}{\epsilon}$$

, in other words, the difference of rendered image when changing some parameters by ϵ amount.

Fig. 5 shows the gradient image using our renderer and the value estimated using finite difference. Note that they have similar values at similar locations. The difference might be due to the noise and ϵ selection.

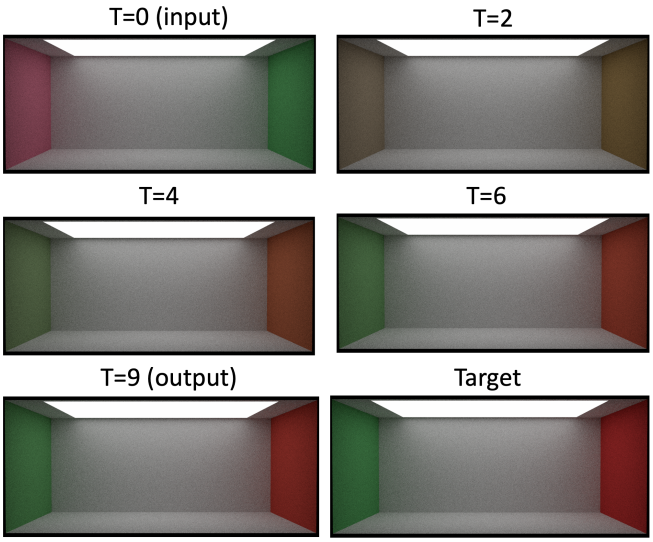

Fig. 4. SGD optimization of Lambertian albedo value, with grad on a multiple surface. T is the step of the gradient descent process.

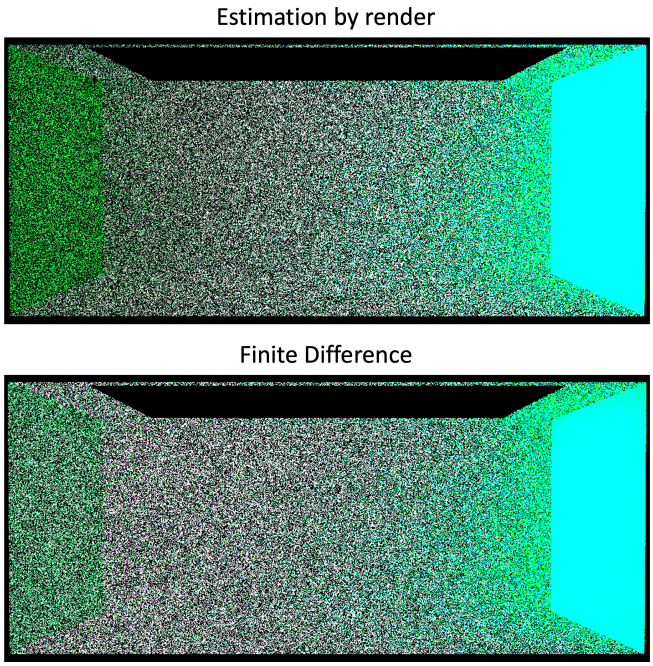

Fig. 5. Gradient image using our renderer and the value estimated using finite difference.

5 SUPPORT OF ADAM OPTIMIZER

The naive gradient descent algorithm, although decent to the correct direction, might be unstable when the gradient is unstable, or it might be hard to converge when closing to the optimal solution. Adam [4] utilizes momentum to stabilize the descending process.

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Fig. 6. Persuade Code of Adam [4]

From Fig. 7, we can observe the unstable behavior of SGD when the scene is hard to optimize. By using Adam, the optimization process is more stable.

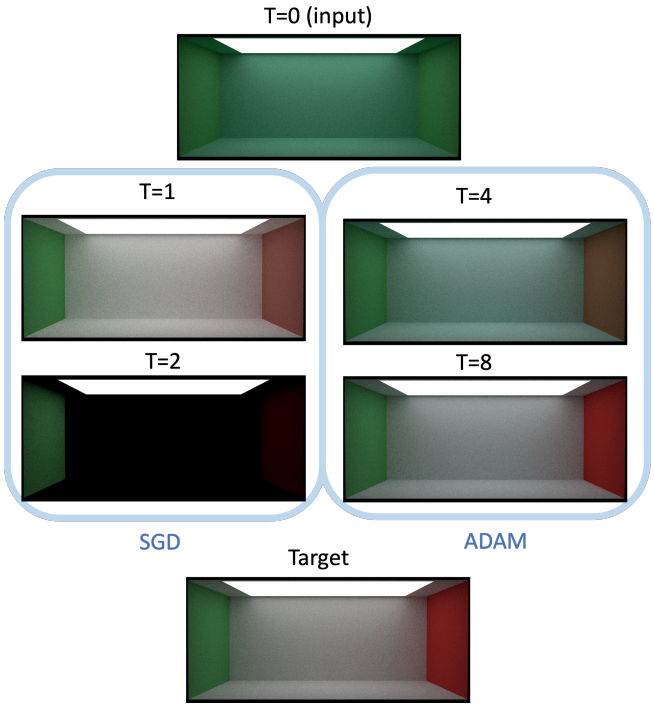

Fig. 7. Optimizing multiple Lambertian materials. The gradient of this one is less stable as the largest surface requires optimization, whose variation changes the scene largely.

6 DISTANCE-DEPENDENT LIGHT EMISSION

Typically light location is considered as a global parameter. However, it is possible to make a "movable" light source that does not modify

geometry using:

$$emission(x) = \frac{c}{distance(x - x_0)}$$

, where c is a constant and x is the hit position, x_0 is the point with highest emission.

Fig. 8 shows the successful optimization of the light center.

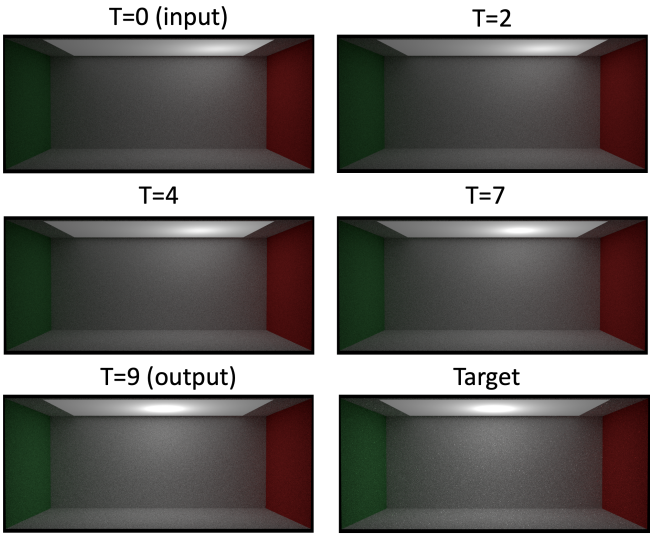

Fig. 8. Optimize the center of the light source.

7 IMPLEMENTATION DETAILS

7.1 JSON-based Configuration

Setting a material to require gradient easily can be achieved with one line of change to the DIRT JSON configure file. An example is shown in Fig. 9.

```
{
    "type": "lambertian",
    "name": "green",
    "albedo": [0.12, 0.45, 0.15]
},
```

No optimize (DIRT)

```
{
    "type": "gaussian light",
    "name": "light",
    "grad" : "true",
    "emit": 1,
    "center": [300, 150, 0]
}
```

Require Grad (ours)

Fig. 9. Configure Grad Using JSON

7.2 Differentiable Material Type

Compared with the original DIRT material, our framework requires a small number of functions to add to each differentiable material type, as shown in Fig. 10.

The *update param* function will update the value given gradient (after multiplying by the learning rate). The *get score* functions will return the local score at the hit position.

```
Color3f score; //
string name;
virtual void update_param(const Vec3f &grad) {};
virtual Vec3f get_score(const Vec3f &dirIn,
    const Vec3f &scattered, const HitInfo &hit) const {return Color3f(0.0f);}
```

Fig. 10. New functions needed to support Differentiable Material.

7.3 Differentiable Renderer

Compared with the original path tracer, our differentiable path tracer needs to accumulate the score at each hit (if the material is differentiable), shown in Fig. 11.

```
Color3f eval = hit.mat->eval(ray_1.d, srec.scattered, hit);
result += throughput * emitted;
throughput *= eval/pdf;
if (hit.mat->requries_grad){
    grad_info.at(hit.mat->name + "_score") += hit.mat->get_score(ray_1.d, srec.scattered, hit);
```

Fig. 11. Differentiable Tracer.

REFERENCES

[1] Ioannis Gkioulekas, Anat Levin, and Todd Zickler. 2016. An Evaluation of Computational Imaging Techniques for Heterogeneous Inverse Scattering, Vol. 9907. 685–701. https://doi.org/10.1007/978-3-319-46487-9_42

[2] James T. Kajiya. 1986. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. Association for Computing Machinery, New York, NY, USA, 143–150. https://doi.org/10.1145/15922.15902

[3] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. 2020. Differentiable Rendering: A Survey. arXiv:2006.12057 [cs.CV]

[4] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[5] Stephen Robert Marschner. 1998. *Inverse rendering for computer graphics*. Ph. D. Dissertation. USA. Advisor(s) Greenberg, Donald P. AAI9839924.