Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder

Chakravarty R.alla Chaitanya et al.  
SIGGRAPH 2017

Jee-Hyeok Park  
2018.04.05  
Computer Graphics @ Korea University
Index

• Abstract
• Introduction
• Related Work
• Path Tracing
• Image Sequence Reconstruction with Recurrent Denoising Autoencoder
• Implementation and Result
• Conclusion and Future Work
• References
Abstract

• Path Tracing
  ▪ 빛의 경로를 시뮬레이션해서 Rendering 하는 기술

• Denoising Autoencoder
  ▪ AI가 Denoising 해주는 기술

빠르고 정교한 영상
Abstract

- 결과

Learning-based filter  Recurrent autoencoder  real sample/pixel input
Introduction
Title of Paper

- Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder
  - Interactive
  - Reconstruction
  - Monte Carlo Image (Path Tracing)
  - Sequence
  - Recurrent
  - Denoising
  - Autoencoder
Background

• Ray Tracing과 Path Tracing으로 Rendering하는 방식이 많이 나오

• 게임 등의 어플리케이션에서는 Realistic한 Rendering을 많이 요구

• Realistic한 Rendering을 하려면 Real Time을 구현하기 어려움

• Monte Carlo Path Tracing을 적게 Sampling하면 Noise가 있는 이미지가 나오

• 이 이미지를 만들고 Reconstruct하는 연구들이 나오

• CNN 등 Deep Learning 통해 Noise가 있는 하나의 Image를 Reconstruct하는 연구가 나오
Contribution

• Image Sequence에 대해 Reconstruct하고 있기 때문에 (Recurrent Autoencoder이기 때문에) Temporal Stability가 상승함(즉, 시간에 따라 변해도 안정적임)
  ▪ Path Tracing 영상을 denoising하는 선행 연구는 없었음

• End-to-End Training이 보조 Feature들을 잘 이용할 수 있도록 함, 따라서 User Guidance가 필요 없음
Related Work
Related Work
Path Tracing in the Industry

- The Path Tracing Revolution in the Movie Industry, Keller, A. et al. (SIGGRAPH Courses 2015)
  - Path Tracing이 Movie Industry에서 많이 사용됨
  - Rendering Algorithm과 Material Description을 분리시켜 아티스트들이 직관적으로 이해하기 쉬움
  - 모듈화가 용이하여 재사용 가능성을 높음

![Path Tracing Examples]
Related Work

Offline Denoising for Monte Carlo Rendering (1/2)

- Optimizing Path Tracing using Noise Reduction Filters, Jensen. H. W. et al. (WSCG 1995)
  - Path Tracing은 Sample 개수가 적을 수록 Noise가 많이 생기기 마련, Indirect Illumination을 잘 계산하지 못하기 때문
  - Sample 개수가 많으면 느리기 때문에 Sample을 적게 하고 Filter를 통해 이미지의 Noise를 줄이는 연구를 함

No Filter

With Filter
Related Work

Offline Denoising for Monte Carlo Rendering (2/2)

- Robust Denoising using Feature and Color Information, Fabrice Rousselle et al. (Pacific Graphics 2013)
  - Pixel의 Normal, Depth, Texture 등 Auxiliary Data를 이용해 Filter를 만들어서 Denoising함
  - Noise를 효과적으로 제거하기는 하지만 많은 시간이 걸림
Related Work

Interactive Denoising for Monte Carlo Rendering

- Edge-avoiding A-Trous wavelet transform for fast global illumination, Holger Dammertz et al. (High Performance Graphics 2010)
  - 빠르고 성능이 좋음
  - Error Estimate가 부족하기 때문에 Local Detail이 사라지는 경우가 있음
  - Filter를 만들기 위해 User Parameter가 필요함
Related Work

Convolutional Neural Network

- Gradient-Based Learning Applied to Document Recognition, Yann LeCun et al. (In Proc. IEEE 1998)
  - 컬러라이제이션 등 이미지 프로세싱에서 많이 쓰임
  - 이미지 작은 공간의 Feature를 뽑아낼 수 있기 때문
Related Work

Recurrent Neural Network

- Bidirectional Recurrent Convolutional Networks for Multi-frame Super-Resolution, Yan Huang *et al.* (NIPS 2015)
  - RNN을 사용하여 multi-frame에 대해 Super-Resolution하는 논문

![Diagram of Recurrent Neural Network](image)
Related Work

Image Restoration using Deep Learning (1/3)

  - CNN을 이용해 해상도를 자동으로 키워주는 논문
Related Work

Image Restoration using Deep Learning (2/3)

- Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections, Xiao-Jiao Mao et al. (NIPS 2016)
  - 박일러닝 모델을 제안하는 논문
  - Encode 스테이지와 Decode Symmetric하게 연결함으로써 Image Denoising, Super Resolution, Inpainting 등을 더 잘할 수 있게 함
Related Work

Image Restoration using Deep Learning (3/3)

- Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings, Steve Bako et al. (ACM Transactions on Graphics 2017)
  - Monte Carlo Rendering된 이미지를 Denoising 하는 논문
Path Tracing
Rendering

- 3D 세계를 2D 평면에 그리는 것
Ray Tracing

- 사물을 본다는 것은 사물이 반사(혹은 방사)한 빛을 시세포가 감지하는 것
  - 컴퓨터로 시뮬레이션 해서 Rendering

![Diagram of ray tracing](image-url)
Path Tracing

- Rendering Equation (Path Tracing)

\[
L_o(x, \omega_o, \lambda, t) = L_e(x, \omega_o, \lambda, t) + \int_{\Omega} f_r(x, \omega_i, \omega_o, \lambda, t) L_i(x, \omega_i, \lambda, t) (\omega_i \cdot n) \, d\omega_i
\]
Monte Carlo Integration

\[ \langle F^N \rangle = (b - a) \frac{1}{N} \sum_{i=0}^{N-1} f(X_i) \]
1spp Unidirectional Path Tracer

- Rasterize primary hits into a G-Buffer
- Path tracing from the primary hits
  - 1 ray for direct shadows
  - 2 rays for indirect (sample + connect)
- 1 direct + 1 indirect path (1spp)
G-Buffer

- GPU를 사용해서 Rasterize
- G-buffer 사용
  - RGB: 16bit
  - depth(linearized): 16bit
  - view space normal(projected): 8bit
  - material roughness: 8bit

7개
1024spp Path Tracing

Reference (1024spp) (~240s)

1spp (~70ms)
Image Sequence Reconstruction with Recurrent Denoising Autoencoder


**Architecture**

- **Convolutional Neural Network**
  - Extract Feature

- **Denoising Autoencoder with Skip Connections**
  - Denoising

- **Recurrent Convolutional Block**
  - Avoid Flickering
Machine Learning & Deep Learning

• 기계가 일일이 코드로 명시하지 않은 동작을 데이터로부터 학습하여 실행할 수 있도록 하는 알고리즘을 개발하는 연구 분야

![Diagram showing classification]

Artificial Intelligence
Any technique which enables computers to mimic human behavior.

Machine Learning
Subset of AI techniques which use statistical methods to enable machines to improve with experiences.

Deep Learning
Subset of ML which make the computation of multi-layer neural networks feasible.
Artificial Neural Network (1/2)

Input: 구름, 온도, 밝기, 습도

Output: 비 올 확률

구름*\( w_1 \) + 온도*\( w_2 \) + 밝기*\( w_3 \) + 습도*\( w_4 \) + \( b \) = 비 올 확률

수 많은 데이터가 있다면? 학습이 가능함
Artificial Neural Network (2/2)

- **Cost Function**
  - Loss Function은 실제 결과 값과 예측 값의 차이
  - Cost Function은 Loss Function의 합
  - $C(w_1, w_2, w_3...)$, 각 weight에 관한 함수

- **Gradient Descent**는 Feedforward의 일종
Convolutional Neural Network

\[ a_i^l = \sigma \left( \sum_j a_{j}^{l-1} \ast k_{ij}^l + b_i^l \right) \]
Denoising Autoencoder

- Autoencoder
  - 자동으로 인코딩해주는 모델

- Denoising Autoencoder
  - 자동으로 Denoising 해주는 모델
  - Input에 일부러 Noise를 넣고 Autoencoder를 사용, Decode된 결과가 Noise를 넣기 전의 결과와 같도록 학습
Skip Connections

• Encode 레이어와 Decode 레이어를 연결
  ▪ 바로 전 Step의 정보만 이용하는 것이 아니라 대응되는 Feature Map의 정보도 이용

![Diagram of Skip Connections with Convolution and Deconvolution layers](image-url)
• RNN을 추가한 Denoising Autoencoder
  - CNN만 이용하면 각 이미지에 대해서는 잘 Denoise됨, 하지만 Image Sequence(영상)에 대해서 잘 되지 않음(Flickering 현상)
  - Recurrent Network를 사용함
Recurrent Denoising Autoencoder (2/2)

\[
h_i = O_i = C_{3\times3} (C_{3\times3} (C_{3\times3}(I_i)^\wedge h_{i-1}))
\]
Training

- 각 씬마다 1000개의 frame
  - 각 frame당 1개의 target image
  - 각 frame당 10개의 noisy image

- 1024x1024 크기의 이미지
  - 128x128 크기의 이미지를 추출
  - 7개의 연속적인 noisy image를 추출해 training
Loss Function

\[ L_s = \frac{1}{N} \sum_{i}^{N} |p_i - t_i| \]
\[ L_g = \frac{1}{N} \sum_{i}^{N} |\nabla p_i - \nabla t_i| \]
\[ L_t = \frac{1}{N} \sum_{i}^{N} \left( \left| \frac{\partial p_i}{\partial t} - \frac{\partial t_i}{\partial t} \right| \right) \]

- color
- edge
- time

\[ L = w_s L_s + w_g L_g + w_t L_t \]
\[ w_{s/g/t} = 0.8/0.1/0.1 \]
Implementation and Result
Network Design Analysis

Auxiliary Features

![Graph showing training loss over epochs for different auxiliary features. The graph compares training loss on a logarithmic scale for various feature combinations, including color only, untextured color, untextured + depth, untextured + normal, untextured + normal + depth, and untextured + normal + depth + roughness. The x-axis represents epochs, and the y-axis represents training loss. Each line on the graph represents a different feature combination, with distinct colors for easy differentiation.]
## Network Design Analysis

### Network Size (1/2)

<table>
<thead>
<tr>
<th>AE smallest</th>
<th>AE small</th>
<th>AE</th>
<th>AE large</th>
<th>AE feat small</th>
<th>AE feat large</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv-43</td>
<td>conv-43</td>
<td>conv-43</td>
<td>conv-43</td>
<td>conv-36</td>
<td>conv-43</td>
</tr>
<tr>
<td>conv-57</td>
<td>conv-57</td>
<td>conv-57</td>
<td>conv-57</td>
<td>conv-54</td>
<td>conv-57</td>
</tr>
<tr>
<td>conv-57</td>
<td>conv-76</td>
<td>conv-76</td>
<td>conv-76</td>
<td>conv-81</td>
<td>conv-76</td>
</tr>
<tr>
<td>conv-76</td>
<td>conv-101</td>
<td>conv-101</td>
<td>conv-122</td>
<td>conv-101</td>
<td>conv-100</td>
</tr>
<tr>
<td>conv-101</td>
<td>conv-135</td>
<td>conv-122</td>
<td>conv-101</td>
<td>conv-100</td>
<td></td>
</tr>
<tr>
<td>conv-135</td>
<td>conv-135</td>
<td>conv-81*2</td>
<td>conv-76*2</td>
<td>conv-83*2</td>
<td></td>
</tr>
<tr>
<td>conv-101*2</td>
<td>conv-76*2</td>
<td>conv-76*2</td>
<td>conv-54*2</td>
<td>conv-57*2</td>
<td>conv-69*2</td>
</tr>
<tr>
<td>conv-76*2</td>
<td>conv-57*2</td>
<td>conv-57*2</td>
<td>conv-36*2</td>
<td>conv-43*2</td>
<td>conv-58*2</td>
</tr>
<tr>
<td>conv-57*2</td>
<td>conv-57*2</td>
<td>conv-24*2</td>
<td>conv-32*2</td>
<td>conv-48*2</td>
<td></td>
</tr>
<tr>
<td>conv-43*2</td>
<td>conv-43*2</td>
<td>conv-43*2</td>
<td>conv-128,64</td>
<td>conv-128,64</td>
<td>conv-128,64</td>
</tr>
<tr>
<td>conv-32*2</td>
<td>conv-32*2</td>
<td>conv-32*2</td>
<td>conv-32*2</td>
<td>conv-128,64</td>
<td>conv-128,64</td>
</tr>
<tr>
<td>conv-64</td>
<td>conv-128,64</td>
<td>conv-128,64</td>
<td>conv-128,64</td>
<td>conv-128,64</td>
<td>conv-128,64</td>
</tr>
</tbody>
</table>

**Number of Layers**

**Number of Features**
Network Design Analysis

Network Size (2/2)

Number of Layers

Number of Features
Network Design Analysis

Recursive Filtering

Input Network Output

(a) iteration #1 (b) iteration #2 (c) iteration #5 (d) iteration #10

Auxiliary Features
Training Data

• Fly-throughs of 3D Scene
  ▪ Different Scene Geometries
  ▪ Lighting Setups
  ▪ Camera Motions

SponzaDiffuse
Diffuse BRDF
Area Light
2000 spp

SponzaGlossy
GGX BRDF
Area Light
4000 spp

Classroom
Directional Light
4000 spp
Reconstruction Quality

- Other Methods
  - Learning Based Filter
  - Axis Aligned Filter (AAF)
  - A-Trous Wavelet Filter (EAW)
  - SURE-based Filter (SBF)

Structural Similarity

\[
SSIM(x, y) = \frac{(2\mu_x \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}
\]

\[
c_1 = (k_1 L)^2, \quad c_2 = (k_2 L)^2
\]

\[
k_1 = 0.01, \quad k_2 = 0.01
\]

\[
L = 2^{\text{bits per pixel}} - 1
\]

Root Mean Square Error

\[
RMSE = \sqrt{E\left(\left(\hat{\theta} - \theta\right)^2\right)}
\]
Reconstruction Quality

1 Sample Per Pixel / 1-Bounce

![Image of reconstruction results and quality metrics]

<table>
<thead>
<tr>
<th>Filter</th>
<th>CornellBox</th>
<th>Sponza</th>
<th>GlossySponza</th>
<th>SanMiguel</th>
<th>Classroom</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>SSIM</td>
<td>RMSE</td>
<td>SSIM</td>
<td>RMSE</td>
</tr>
<tr>
<td>AAF</td>
<td>0.018</td>
<td>0.978</td>
<td>0.029</td>
<td>0.869</td>
<td>0.069</td>
</tr>
<tr>
<td>EAW</td>
<td>0.019</td>
<td>0.978</td>
<td>0.043</td>
<td>0.902</td>
<td>0.089</td>
</tr>
<tr>
<td>SBF</td>
<td>0.044</td>
<td>0.818</td>
<td>0.060</td>
<td>0.502</td>
<td>0.057</td>
</tr>
<tr>
<td>Our</td>
<td>0.014</td>
<td>0.984</td>
<td>0.016</td>
<td>0.953</td>
<td>0.034</td>
</tr>
</tbody>
</table>
Reconstruction Quality

256 Sample Per Pixel / 2-Bounce

<table>
<thead>
<tr>
<th>Our reconstruction result</th>
<th>Reference</th>
<th>MC input</th>
<th>EAW</th>
<th>SBF</th>
<th>NFOR</th>
<th>Our</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bathroom</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Horse Room</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Living Room</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Filter</th>
<th>Bathroom</th>
<th>Horse Room</th>
<th>Living Room</th>
<th>Frame time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>SSIM</td>
<td>RMSE</td>
<td>SSIM</td>
</tr>
<tr>
<td>EAW</td>
<td>0.127</td>
<td>0.840</td>
<td>0.094</td>
<td>0.831</td>
</tr>
<tr>
<td>SBF</td>
<td>0.044</td>
<td>0.902</td>
<td>0.040</td>
<td>0.939</td>
</tr>
<tr>
<td>NFOR</td>
<td>0.009</td>
<td>0.990</td>
<td>0.018</td>
<td>0.984</td>
</tr>
<tr>
<td>Our</td>
<td>0.041</td>
<td>0.944</td>
<td>0.034</td>
<td>0.945</td>
</tr>
</tbody>
</table>
Performance

NVIDIA (Pascal) Titan X
CUDA kernel
cuDNN 5.1
1280 x 720 px
70ms – 1 sample/pixel
54.9ms – Reconstruction
Conclusion and Future Work
Conclusion and Future Work

- 처음으로 Global Illumination이 되고 Noise가 제거된 Animation Sequences를 Recurrent Denoising Autoencoder를 이용해서 만든 것

- 더 속도를 향상시켜 Real Time을 구현할 수 있도록 Network를 Optimize해볼 것

- Deep Learning Feature에 Time이나 Lens 정보를 넣어서 Motion Blur와 Depth of Field를 해볼 것
References
References

• [https://www.youtube.com/watch?v=xrOQGYuoluo](https://www.youtube.com/watch?v=xrOQGYuoluo)
  ▪ 논문 저자가 직접 한 프레젠테이션