Denoising Algorithms: Path to Neural Networks II

Image courtesy Vogel et al. [2018]

Philipp Slusallek    Karol Myszkowski    Gurprit Singh
Recap
Image-space Adaptive Sampling

Hachisuka et al. [2008]
Image-space Adaptive Sampling

Multidimensional Adaptive Sampling

Hachisuka et al. [2008]
Depth of field
The trajectories of samples originating from a single apparent surface never intersect.
input Monte Carlo (8 samples/pixel)

after RPF (8 samples/pixel)
(a) Input MC (8 spp)  
(b) Dependency on \((u, v)\)  
(c) Our approach (RPF)
Bilateral Filtering

\[ BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) I_q \]

\[ W_p = \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) \]

Spatial weight

Range weight

Multiplication of range and spatial weights

Bilateral filter weights at the central pixel

Input

Result
Bilateral Filtering of Features

\[ w_{ij} = \exp\left[-\frac{1}{2\sigma^2_p} \sum_{1 \leq k \leq 2} (\bar{p}_{i,k} - \bar{p}_{j,k})^2 \right] \times \]

\[ \exp\left[-\frac{1}{2\sigma^2_c} \sum_{1 \leq k \leq 3} \alpha_k (\bar{c}_{i,k} - \bar{c}_{j,k})^2 \right] \times \]

\[ \exp\left[-\frac{1}{2\sigma^2_f} \sum_{1 \leq k \leq m} \beta_k (\bar{f}_{i,k} - \bar{f}_{j,k})^2 \right], \]
Multi-layer Perceptron

\[ y_1 = f(x_1 w_{11} + w_{10}) \]
\[ y_2 = f(x_1 w_{21} + w_{20}) \]
\[ y_3 = f(x_1 w_{31} + w_{30}) \]
Multi-layer Perceptron

\[ y_1 = f(x_1w_{11} + w_{10} ) \]
\[ y_2 = f(x_1w_{21} + w_{20} ) \]
\[ y_3 = f(x_1w_{31} + w_{30} ) \]
Filter weights

For cross Bilateral filters:

\[ d_{i,j} = \exp \left[ - \frac{\| \bar{p}_i - \bar{p}_j \|^2}{2\alpha_i^2} \right] \times \exp \left[ - \frac{D(\bar{c}_i, \bar{c}_j)}{2\beta_i^2} \right] \times \prod_{k=1}^{K} \exp \left[ - \frac{D_k(\bar{f}_{i,k}, \bar{f}_{j,k})}{2\gamma_{i,k,i}^2} \right] \]

Pixel screen coordinates
Mean sample color value
Scene features
Our result with a cross-bilateral filter (4 spp)
Basics of Neural Networks

Each network has a forward pass and a backward (back-propagation) pass.

All components of the network must be differentiable.

Differentiability is essential for back-propagation of error.
Convolution

No zero padding
Stride-1 Convolution

No zero padding
Stride-1 Convolution

No zero padding
Stride-1 Convolution

No zero padding
Stride-2 Convolution
Zero Padding and Strides

1D image to illustrate the strides and zero padding

![Diagram of stride and zero padding]
Strides

1D image to illustrate the strides and zero padding
Max Pooling / Down Sampling

![Max Pooling Example](image.png)
Overview on Convolutional Neural Networks

Image Courtesy: Mathworks (online tutorial)
Multi-layer Perceptron vs. CNNs
Multi-layer Perceptron vs. CNNs

Multi-layer perceptron

- All nodes are fully connected in all layers
- In theory, should be able to achieve good quality results in small number of layers.
- Number of weights to be learnt are very high

CNNs

- Weights are shared across layers
- Requires significant number of layers to capture all the features (e.g. Deep CNNs)
- Relatively small number of weights required
Introduction to CNNs

Kernel-Predicting Denoising
Kernel-Predicting Networks for Denoising Monte-Carlo Renderings

Bako et al. [2017]
Limitations of MLP based Denoiser

Kernel was pre-selected to be joint bilateral filter

- Unable to explicitly capture all details
- lacked flexibility to handle wide range of MC noise in production scenes

Fixed

- can cause unstable weights causing bright ringing and color artifacts

Too many parameters to optimize
Requirements

The function must be flexible to capture complex relationship between input data and reference colors over wide range of scenarios.

Choice of loss function is crucial. Should capture perceptual aspects of the scene.

To avoid overfitting, large dataset required
Using a Vanilla CNN

Denoising a raw, noisy color buffer causes overblurring

- difficulty in distinguishing scene details and MC noise

High dynamic range

- can cause unstable weights causing bright ringing and color artifacts
Vanilla CNN
Denoising Model

\[ \hat{\theta}_p = \arg\min_{\theta} \ell(\bar{c}_p, g(X_p; \theta)) \]

Denoised function with parameters

Reference image

\[ \hat{c}_p = g(X_p; \hat{\theta}_p) \]

Denoised value

\[ \ell(\bar{c}, \hat{c}) \]

Loss function
Computational Model

\[ \hat{\theta}_p = \arg\min_{\theta} \sum_{q \in N(p)} (c_q - \theta^T \phi(x_q))^2 \omega(x_p, x_q) \]

Neighborhood

\[ \hat{c}_p = g(x_p; \hat{\theta}_p) \]
Denoised value

\[ \hat{c}_p = \hat{\theta}_p^T \phi(x_p) \]
Final denoised value

\[ \phi : \mathbb{R}^{3+\bar{D}} \rightarrow \mathbb{R}^M \]

\[ \omega(x_p, x_q) \]
Kernel weights
Direct Prediction Network

Direct prediction convolution network: outputs denoised image

\[ \hat{c}_p = g_{\text{direct}}(X_p; \theta) = z_p^L \]
Direct Prediction Network

Direct prediction convolution network: outputs denoised image

\[ \hat{c}_p = g_{\text{direct}}(X_p; \theta) = z_p^L \]

Issues:

The constrained nature and complexity of the problem makes optimization difficult.

The magnitude and variance of stochastic gradients computed during training can be large, which slows convergence of training loss.
Kernel Prediction Network

Kernel prediction convolution network: outputs learned kernel weights

\[ w_{pq} = \frac{\exp([z^L_q]^T)}{\sum_{q' \in \mathcal{N}(p)} \exp([z^L_{q'}]^T)} \]

Softmax activation to enforce weights within range

\[ 0 \leq w_{pq} \leq 1 \]

Denoised color values:

\[ \hat{c}_p = g_{\text{weighted}}(X_p; \theta) = \sum_{q \in \mathcal{N}(p)} c_q w_{pq} \]
Kernel Prediction Network

\[ w_{pq} = \frac{\exp([z^L_p]_q)}{\sum_{q' \in N(p)} \exp([z^L_p]_{q'})} \]

\[ \tilde{c}_p = g_{\text{weighted}}(X_p; \theta) = \sum_{q \in N(p)} c_q w_{pq} \]

0 ≤ \( w_{pq} \) ≤ 1

Final color estimate always lies within the convex hull of the respective neighborhood (avoid color shifts).

Ensures well-behaved gradients of the error w.r.t the kernel weights
Proposed Architecture
Diffuse/Specular components

Each component is denoised separately.

Diffuse components are well-behaved and typically has small ranges

- albedo is factored out to allow large range kernels

\[ \tilde{c}_{\text{diffuse}} = c_{\text{diffuse}} \odot (f_{\text{albedo}} + \epsilon) \]

Specular components are challenging due to high dynamic ranges: uses logarithmic transform

\[ \tilde{c}_{\text{specular}} = \log(1 + c_{\text{specular}}) \]
Training Dataset: 600 frames
Training

8-hidden layers used with 100 kernels of 5x5 in each layer for each network

For KPCN (kernel-predicting network), output kernel size used = 21

Weights for 128 app and 32 spp networks were initialized using Xavier method

Diffuse and specular components were independently trained with L1 loss metric
Learning rate of DPCN vs. KPCN

On Cars 3 dataset, KPCN converges 5-6x faster
Results

Ours

Input (128 spp)

NFOR (log)

Ours

Ref. (32K spp)

relative $\ell_2$

1 – SSIM

29.15e-3

0.562

0.90e-3

0.019

0.69e-3

0.017

38.57e-3

0.552

1.12e-3

0.025

0.92e-3

0.024
Results

Input (32 spp)
Results

w/o Decomposition, w/o Albedo divide
w/ Decomposition, w/o Albedo divide
w/o Decomposition, w/ Albedo divide
w/ Decomposition, w/ Albedo divide
Ref. (2K spp)
Results

Also works on Piper short movie frames
Interactive Reconstruction of Monte Carlo Sequences

Chaitanya et al. [2017]
Motivation: Interactive Reconstruction

Limited to a few rays per pixel @ 1080p @ 30Hz

Never enough to reconstruct an image

Deep learning approach for interactive graphics
Motivation: Interactive Reconstruction

Limited to a few rays per pixel @ 1080p @ 30Hz

Never enough to reconstruct an image

Deep learning approach for interactive graphics
Problem Statement

Handle generic effects:

- Soft shadows
- Diffuse and specular reflections
- Global illumination (one-bounce)
- No Motion blur or depth of field
System setup: Path tracing
System setup: Path tracing
System setup: Path tracing

Rasterize primary hits in G-buffers

Path-tracing from the primary paths
- 1 ray for direct shadows
- 2 rays for indirect (sample + connect)

1 direct + 1 indirect path (spp)
Denoising Autoencoder (DAE)

Train auto encoders to reconstruct image from 1spp
Fig. 2. Architecture of our recurrent autoencoder. The input is 7 scalar values per pixel (noisy RGB, normal vector, depth, roughness). Each encoder stage has a convolution and $2 \times 2$ max pooling. A decoder stage applies a $2 \times 2$ nearest neighbor upsampling, concatenates the per-pixel feature maps from a skip connection (the spatial resolutions agree), and applies two sets of convolution and pooling. All convolutions have a $3 \times 3$-pixel spatial support. On the right we visualize the internal structure of the recurrent RCNN connections. $I$ is the new input and $h$ refers to the hidden, recurrent state that persists between animation frames.
Recurrent Neural Networks

Encoder and decoder stages for dimensionality reduction

Encoder

Decoder
Recurrent Neural Networks

Encoder and decoder stages for dimensionality reduction

Encoder

Decoder

Skip connections to reintroduce lost information
Auxillary Features

Untextured color

View space normals

Linearize depth
Training sequences

SponzaDiffuse

SponzaGlossy

Classroom
Realistic Image Synthesis SS2018

1spp approx. 70 ms
DAE 1spp
approx. 70 ms + approx. 60 ms
Recurrent Denoising Autoencoder

Feedback loops to retain important information after every encoding stage

Encoder

Decoder

RCNN
Recurrent Neural Networks vs. Simple Feed-Forward NN

Recurrent Neural Network

Feed-Forward Neural Network

Source link
Recurrent Neural Networks

An unrolled recurrent neural network.
Recurrent Neural Networks

Fully convolutional blocks to support arbitrary image resolution

6 RNN blocks, one per pool layer in the encoder

Design:
- 1 conv layer (3x3) for current features
- 2 conv layers (3x3) for previous features
Recurrent Neural Networks
Recurrent Neural Networks

CNNs, fixed input, fixed output

one to one
Recurrent Neural Networks

CNNs, fixed input, fixed output

one to one

\[ \text{Seq} \]

e.g., image captioning takes an image as input and outputs a sentence of words
Recurrent Neural Networks

CNNs, fixed input, fixed output

Sequence input

one to one

Sequence output

one to many

many to one

e.g., to know the sentiments of a sentence
Recurrent Neural Networks

CNNs, fixed input, fixed output

Sequence input

Sequence output

e.g. Machine translation
Recurrent Neural Networks

CNNs, fixed input, fixed output

Sequence input

Sequence output

Synced sequence input & output

E.g., video classification where we want to label each frame

E.g., Machine translation

Sequence input, Sequence output.
Training

Input is a sequence of 7 frames

128x128 random image crop per sequence

Play the sequence forward/backward

Each frame advance the camera or random seed
Loss Functions

Spatial Loss to emphasize more the dark regions

\[ L_s = \frac{1}{N} \sum_{i}^{N} |P_i - T_i| \]
Loss Functions

Spatial Loss to emphasize more the dark regions

\[ L_s = \frac{1}{N} \sum_{i}^{N} |P_i - T_i| \]

Temporal loss

\[ L_t = \frac{1}{N} \sum_{i}^{N} \left( \left| \frac{\partial P_i}{\partial t} - \frac{\partial T_i}{\partial t} \right| \right) \]
## Loss Functions

### Spatial Loss to emphasize more the dark regions

\[ L_s = \frac{1}{N} \sum_{i}^{N} |P_i - T_i| \]

### Temporal loss

\[ L_t = \frac{1}{N} \sum_{i}^{N} \left( |\frac{\partial P_i}{\partial t} - \frac{\partial T_i}{\partial t}| \right) \]

### High frequency error norm loss for stable edges

\[ L_g = \frac{1}{N} \sum_{i}^{N} |\nabla P_i - \nabla T_i| \]
Loss Functions

Spatial Loss to emphasize more the dark regions

\[ L_s = \frac{1}{N} \sum_{i}^{N} |P_i - T_i| \]

Temporal loss

\[ L_t = \frac{1}{N} \sum_{i}^{N} \left( \left| \frac{\partial P_i}{\partial t} - \frac{\partial T_i}{\partial t} \right| \right) \]

High frequency error norm loss for stable edges

\[ L_g = \frac{1}{N} \sum_{i}^{N} |\nabla P_i - \nabla T_i| \]

Final Loss is a weighted averaged of above losses

\[ L = w_sL_s + w_gL_g + w_tL_t \]
Training Loss depends on Auxiliary Features

Auxiliary Features

Training loss vs. Epochs

- Color only
- Untextured color
- Untextured + depth
- Untextured + normal
- Untextured + normal + depth
- Untextured + normal + depth + roughness
Temporal Stability
Introduction to CNNs

Kernel Predicting Denoising

Sample-based MC Denoising (next lecture)
Thanks to Chaitanya and colleagues for making their slides publicly available.