ISSN:1226-7244 (Print)
ISSN:2288-243X (Online)
j.inst.Korean.electr.electron.eng.Vol.21,No.4,420~423,December 2017
논문번호 17-04-15
http://dx.doi.org/10.7471/ikeee.2017.21.4.420
90

렌더링 노이즈 제거를 위한 뉴럴 네트워크 가속기 구현
Implementation of Neural Network Accelerator for Rendering Noise Reduction

남기훈*

Kihun Nam*

Abstract

In this paper, we propose an implementation of a neural network accelerator for reducing the rendering noise. Among the rendering algorithms, we selects a ray tracing to assure a high-definition graphics. Ray tracing rendering uses ray to render. Less use of the ray will result in noise, and if used too much, it will produce a higher quality image, but will take longer. In order to quickly process such lace rendering, an algorithm is used that uses less rays and removes the noise generated. Among such algorithms, there is an algorithm using a neural network, and a neural network accelerator which obtains a filter parameter used in an operation is implemented in order to speed up the operation speed. The time it takes to calculate the parameters used for a pixel is 11.44us.

요 약

본 논문에서는 렌더링 노이즈 제거를 위한 뉴럴 네트워크 가속기 구현을 제안한다. 렌더링 알고리즘 중에 고품질 그래픽스를 보장하는 레이트레이싱을 선택하였다. 레이트레이싱 렌더링은 레이를 사용해 렌더링을 한다. 레이를 적게 사용하게 되면 노이즈가 발생하게 되며, 많이 사용하게 되면 고화질의 이미지를 생성할 수 있으나 연산 시간이 길어진다. 이러한 레이트레싱 렌더링을 빠르게 처리하기 위해서 적게 레이를 사용하고 발생한 노이즈를 제거하는 알고리즘을 사용하게 된다. 그러한 알고리즘 중에 뉴럴 네트워크를 사용한 알고리즘이 있으며, 연산 속도를 빠르게 하기 위해서 연산에 사용되는 필터 파라미터를 구하는 뉴럴 네트워크 가속기를 구현했다. 하나의 픽셀에 사용되는 파라미터를 계산하기 위해 걸리는 시간은 11.44us 이다.

Key words : Rendering, Neural network, MLP, noise, rays

* Dept. of Computer Engineering, Seokyeong University
★ Corresponding author
E-mail: namkh@skuniv.ac.k Tel: +82-2-940-7667
※ Acknowledgment
This research was supported by Seokyeong University in 2017
Manuscript received, Nov, 30, 2017; revised, Dec, 5, 2017; accepted, Dec, 22, 2017

Ⅰ. 서론

레이트레이싱은 렌더링을 하는 알고리즘 중의 많이 사용되는 알고리즘 중의 하나이다. 레이트레이싱 알고리즘은 레이를 광원으로 부터 발사해, 물체에 빛이 닿았을 때, 투과되거나 반사되는 것을 감안하여 색상을 결정한다. 레이트레이싱 렌더링의 경우 레이를 많이 사용하게 되면 고화질의 이미지가 생성되지만, 연산시간이 오래 걸린다. 그러나 레이를 적게 사용하게 되면 이미지의 노이즈가 발생하게 된다. 그림 1은 렌더링된 이미지의 일부분이다. 그림 1을 보면 레이 수에 늘어남에

따라 노이즈가 점차 사라지는 것을 볼 수 있으나, 연산 시간이 늘어나는 것을 확인 할 수 있다.

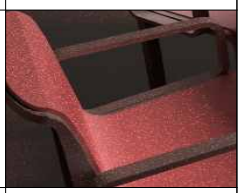
Image		
Rays	8	32
Operating time	157 sec	710 sec
Image		
Rays	256	2,048
Operating time	13,714 sec	46,066 sec

Fig. 1. Rendering images
그림 1. 렌더링 이미지

이러한 노이즈들을 제거하기 위해서 여러 알고리즘을 사용하게 된다. 기존의 연구는 NLM(Non-Local Means filtering), RPF(Random Parameter Filtering), SBF(SURE-Based Filtering), RD(Robust Denoising), LBF(Learning Based Filtering) 등이 있다[1][2][3][4][5]. 그 중 LBF 알고리즘은 고품질 이미지를 생성하는데 좋은 알고리즘이다. LBF의 경우 뉴럴 네트워크를 활용해 최종 색상을 결정하는데 사용한다. 그에 따라 학습을 얼마나 했느냐에 따라서 더욱 고화질의 이미지를 생성할 수 있다.

뉴럴 네트워크에서의 뉴런은 다음 그림 2와 같은 구조를 가진다[6].

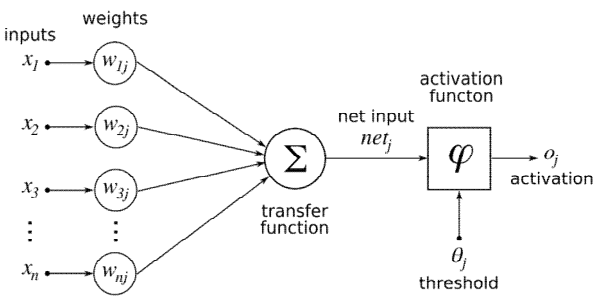


Fig. 2. Neuron
그림 2. 뉴런

뉴런은 입력과 가중치를 곱해 일정한 임계치를 넘으면 출력을 내보내는 구조를 갖는다. 이러한 뉴런들을 네트워크 형태로 묶어서 뉴럴 네트워크가 부르며, 뉴럴 네트워크는 Back Propagation을 사용해 학습한다[7].

이러한 뉴럴 네트워크는 연산양이 많아 Open MP와 같은 멀티 코어 프로그래밍이나, GPU를 범용목적에 사용하는 GP-GPU를 사용한다. 그러나 그 이상의 연산속도를 필요로 하는 경우도 있다. 그럴 경우엔 FPGA 및 ASIC으로 설계해 연산속도를 높인다.

II. 본론

1. Learning Based Filtering

LBF(Learning Based Filtering)은 많은 수의 레이로 렌더링한 이미지를 통해 얻은 필터 파라미터를 학습데이터로 사용해 noise를 제거하는 필터의 파라미터를 학습 시켜 noise를 제거하는 알고리즘이다. 그림 3을 보면 왼쪽의 이미지는 2,048개의 레이를 사용한 이미지 이며, 오른쪽의 이미지는 16개의 레이를 사용해 이미지를 렌더링 한 후 LBF를 적용한 이미지이다.

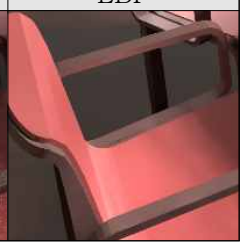
	2,048 rays	LBF
Image		

Fig. 3. LBF image
그림 3. LBF 이미지

LBF에 적용되는 뉴럴 네트워크는 렌더링할 때 사용되는 1차 피처를 이용해 2차 피처를 만든다. 2차 피처를 뉴럴 네트워크의 입력으로 사용한다. 1차 피처는 이미지의 x, y의 위치, 적은 수로 렌더링한 이미지에서의 R, G, B 값, 3D 모델에서의 3차원 좌표, x, y, z와 3D 모델에서의 노말 벡터 x, y, z와 두 개의 텍스처에 대한 x, y, z와 visibility를 가진다. 총 18개의 1차 피처를 통해 36개의 2차 피처를 만드는데, Feature statistics 20개,

MAD 5개, Gradient 5개, MD 5개, SPP 1개로 구성된다. 다음 표 2는 이러한 내용을 정리했다.

Table 1. Primary feature and Secondary feature
표 1. 1차 피처와 2차 피처

Primary Feature	
Image position X, Y	2
Color R, G, B	3
World Position X, Y, Z	3
Normals X, Y, Z	3
Texture 1 X, Y, Z	3
Texture 2 X, Y, Z	3
Visibility	1

Secondary feature	
Feature statistics	20
MAD	5
Gradient	5
MD	5
SPP	1

36개의 2차 피처를 뉴럴 네트워크의 입력으로 사용되며, 10개의 히든 뉴런, 6개의 출력 뉴런을 가지는 뉴럴 네트워크를 구성한다. 6개의 출력 뉴런은 필터에 사용되는 필터 파라미터 이다.

2. 뉴럴 네트워크 가속기

이러한 LBF에 사용되는 뉴럴 네트워크를 빠르게 사용하기 위해서 뉴럴 네트워크 가속기를 설계했다. 가속기는 크게 곱셈기와 덧셈기로 구성되어 있다. 곱셈기는 Modified Booth 알고리즘을 이용해 구현했다. 연산을 빠르게 하기위해 8비트 곱셈기를 이용했다. 이러한 곱셈기는 수식 1을 연산하며, 그림 4는 곱셈기의 블록 다이어그램을 보여준다.[7]

$$Y = \sum_{i=0,\, i=even}^{N-1} (y_{i-1} + y_i - 2y_{i+1}) \cdot 2^i = \sum_{i=0,\, i=even}^{N-1} y' \cdot 2'$$

$$N = even,\ y' = y_{i-1} + y_i - 2y_{i+1} \in \{-2, -1, 0, 1, 2\} \quad (1)$$

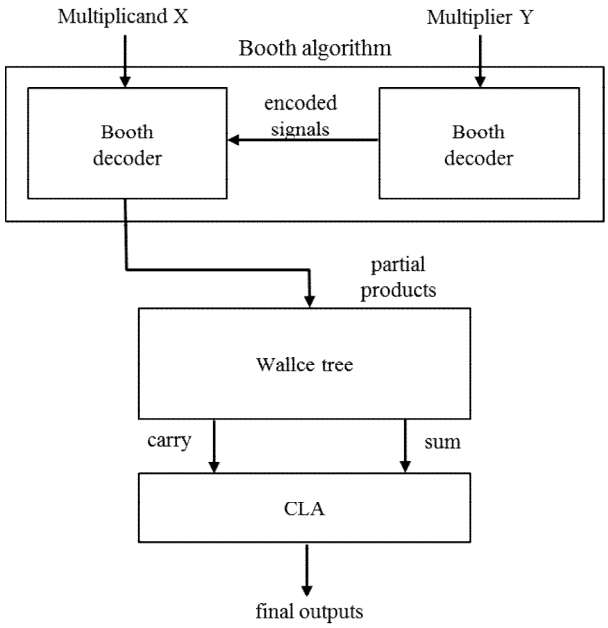


Fig. 4. Modified Booth multiplier
그림 4. Modified Booth 곱셈기

이러한 Modified Booth 곱셈기를 36개를 이용하여 뉴럴 네트워크 가속기를 구성했다. 아래의 그림 5는 가속기의 구조를 보여준다.

Fig. 5. Architecture of Accelerator
그림 5. 가속기의 구조

곱셈기를 36개를 사용한 이유는 히든 레이어의 뉴런을 계산하기 위한 연결이 36 가지이기 때문이다. 다음 그림 6은 연산의 순서를 네트워크를 보여준다.

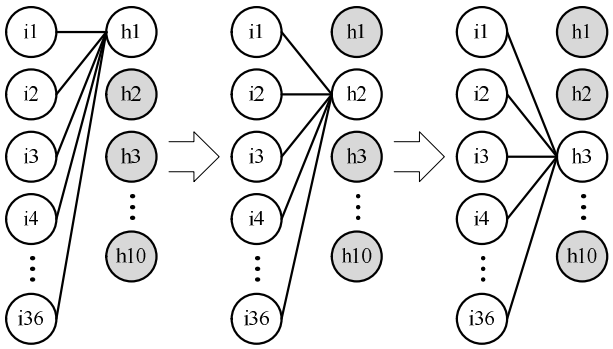


Fig. 6. Operation order of Accelerator
그림 6. 가속기의 연산 순서

3. 실험 결과

실험은 뉴럴 네트워크 가속기를 Verilog HDL로 작성하여 하나의 픽셀의 노이즈를 제거할 때 사용되는 필터 파라미터를 구하는데 걸리는 시간을 계산하였다. 실험에 사용된 FPGA 보드는 Xilinx 사의 VC-707 FPGA 보드다[8]. 하나의 픽셀의 사용되는 필터 파라미터를 계산하는데에는 11.44us의 시간이 걸렸다. 처리시간은 고성능 GPU는 12.4us가 걸리므로 약 1.08배 빠르다.

Ⅲ. 결 론

본 논문에서 적은 수의 레이를 사용하는 레이트레이싱 렌더링 알고리즘의 노이즈를 제거할 때 사용되는 필터파라미터를 계산하는 뉴럴 네트워크를 Verilog HDL로 구현했다. 36개의 입력 뉴런과 10개의 은닉 뉴런, 6개의 출력 뉴런을 가지는 뉴럴 네트워크의 연산 속도를 빠르게 하기 위해서 Modified Booth 곱셈기를 36개를 병렬로 배치해 연산 속도를 높였다. 실험은 뉴럴 네트워크를 Xilinx VC-707 FPGA를 사용했다. 하나의 픽셀에 사용되는 필터파라미터를 계산하기 위해 11.44us의 시간이 걸렸으며, 이러한 뉴럴 네트워크 가속기를 병렬로 설계하면 더 빠른 속도로 노이즈를 제거할 수 있을 것으로 예상된다.

References

[1] Rousselle, Fabrice, Claude Knaus, and Matthias Zwicker. "Adaptive rendering with non-local means filtering," *ACM Transactions on Graphics (TOG)* vol.31,no.6, 195, Nov, 2012. DOI:10.1145/2366145.2366214

[2] Sen, Pradeep, and Soheil Darabi. "On filtering the noise from the random parameters in Monte Carlo rendering." *ACM Trans. Graph.* vol.31,no.3, 18-1. 2012.

[3] Li, Tzu-Mao, Yu-Ting Wu, and Yung-Yu Chuang. "SURE-based optimization for adaptive sampling and reconstruction." *ACM Transactions on Graphics (TOG).* vol.31, no.6, 194. Nov, 2012 DOI:10.1145/2366145.2366213

[4] Rousselle, Fabrice, Marco Manzi, and Matthias Zwicker. "Robust denoising using feature and color information." *Computer Graphics Forum.* Vol. 32, No.7, pp. 121-130, Oct. 2013. DOI: 10.1111/cgf.12219

[5] Kalantari, Nima Khademi, Steve Bako, and Pradeep Sen. "A machine learning approach for filtering Monte Carlo noise." *ACM Trans. Graph.* Vol.34, No.4, 122-1, 2015

[6] https://en.wikibooks.org/wiki/Artificial_Neural_Networks/Activation_Functions

[7] Dhanya. Geethanjali. Sasidharan, Aarathy. Iyer, “Comparison of Multipliers Base on Modified Booth Algorithm,”*International Journal of Engineering Research and Applications.* (2013) Vol.3, No.1, pp.1513- 1516. Jan. 2013.

[8] http://www.xilinx.com