The Journal of the Convergence on Culture Technology (JCCT)
Vol. 4, No. 4, pp.373-377, November 30, 2018. pISSN 2384-0358, eISSN 2384-0366

http://dx.doi.org/10.17703/JCCT.2018.4.4.373
JCCT 2018-11-49

OpenCL을 이용한 랜더링 노이즈 제거를 위한 뉴럴 네트워크 가속기 구현

Implementation of Neural Network Accelerator for Rendering Noise Reduction on OpenCL

남기훈*

Kihun Nam*

요 약 본 논문에서는 OpenCL을 이용한 랜더링 노이즈 제거를 위한 가속기 구현을 제안한다. 렌더링 알고리즘 중에 고품질 그래픽스를 보장하는 레이트레이싱을 선택하였다. 레이 트레이싱은 레이를 사용하여 렌더링하는데 레이를 적게 사용하면 노이즈가 발생한다. 레이를 많이 사용하게 되면 고화질의 이미지를 생성할 수 있으나 연산 시간이 상대적으로 길어지게 된다. 레이를 적게 사용하면서 연산시간을 줄이기 위해 뉴럴 네트워크를 이용한 LBF(Learning Based Filtering) 알고리즘을 적용하였다. 뉴럴 네트워크를 사용한다고 해서 항상 최적의 결과가 나오지는 않는다. 본 논문에서는 성능향상을 위해 일반적인 행렬 곱셈을 기반으로 하는 새로운 기법의 행렬 곱셈 접근법을 제시하였다. 개발환경으로는 고속병렬 처리가 특화된 OpneCL을 사용하였다. 제안하는 구조는 Kintex UltraScale XKU690T-2FDFG1157C FPGA 보드에서 검증하였다. 하나의 픽셀에 사용되는 파라미터를 계산 시간은 Verilog-HDL 구조보다 약 1.12배 빠른 것으로 확인했다.

주요어 : 레이 트레이싱, LBF, 컨볼루션 뉴럴 네트워크, OpenCL, GEMM

Abstract In this paper, we propose an implementation of a neural network accelerator for reducing the rendering noise using OpenCL. Among the rendering algorithms, we selects a ray tracing to assure a high quality graphics. Ray tracing rendering uses ray to render, less use of the ray will result in noise. Ray used more will produce a higher quality image but will take operation time longer. To reduce operation time whiles using fewer rays, Learning Base Filtering algorithm using neural network was applied. it's not always produce optimize result. In this paper, a new approach to Matrix Multiplication that is based on General Matrix Multiplication for improved performance. The development environment, we used specialized in high speed parallel processing of OpenCL. The proposed architecture was verified using Kintex UltraScale XKU6909T-2FDFG1157C FPGA board. The time it takes to calculate the parameters is about 1.12 times fast than that of Verilog-HDL structure.

Key Words : Ray tracing, LBF, CNN, OpenCL, GEMM

Ⅰ. 서 론

현실감 넘치는 조명, 반사, 그림자 등을 구현 할 수 있는 알고리즘으로 레이트레이싱을 들을 수 있다. 레이트레이싱 기법은 자연스러운 명암 표현으로 실사와 거의 유사한 느낌이 들게 만들지만 엄청난 양의 연산

*정회원, 서경대학교 컴퓨터공학과
접수일: 20XX년 X월 X일, 수정완료일: 20XX년 X월 X일
게재확정일: 20XX년 X월 X일

Received: November 1, 2018 / Revised: November 1, 2018
Accepted: November XX, 2018
*Corresponding Author: namkh@skuniv.ac.kr
Dept. Computer Engineering, SeoKyeong Univ, Korea

을 필요로 하는 문제점을 지니고 있다. 이를 해결하고자 레이의 수를 줄이면 노이즈가 발생된다. 그림 1은 샘플 이미지를 렌더링하는데 있어서 레이 수와 연산시간에 대한 결과를 보여준다.[1][2][3]

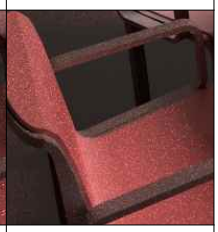
이미지		
레이 수	8	32
연산시간	157 초	710 초
이미지		
레이 수	256	2,048
연산시간	13,714 초	46,066 초

그림 1. 렌더링 결과
Fig 1. Result of Rendering

노이즈를 줄이기 위해서 다양한 알고리즘을 사용한다. 그 중 LBF(Learning Based Filtering) 알고리즘은랜더링 이미지로부터 얻은 필터 파라미터를 학습 데이터로 사용하여 노이즈를 제거하여 고품질의 이미지를 생성하는데 최적인 알고리즘이다. LBF는 뉴럴 네트워크를 이용하여 학습에 따라 이미지의 최종 색상을 결정하게 된다. 뉴럴 네트워크 구조는 다음 그림 2와 같다.[4]

뉴런은 입력 데이터와 가중치를 곱한 후 임계치 값을 연산하여 출력을 얻어내는 구조인데 이와 같은 뉴런들을 네트워크 형태로 묶은 구조를 뉴럴 네트워크라고 일컫는다.

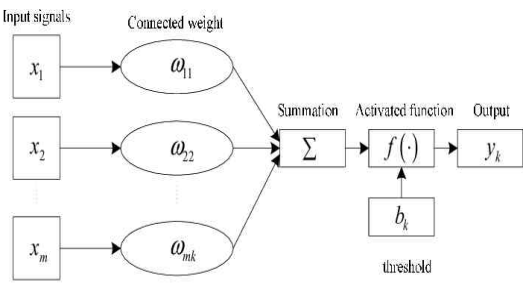


그림 2. 뉴런 모델
Fig 2. Neuron Model

뉴럴 네트워크는 연산량이 많기에 병렬처리가 가능한 환경을 구축하여야 한다. 연산결과를 메모리에 저장하는 횟수를 줄여 처리속도를 높인 CNN 구조를 채택하였다.[5]

본 논문에서는 본론 2장에서는 연산을 가속화하는 뉴럴 네트워크 가속기 구조와 그 구조에 대한 설명으로 3장에서는 개선된 행렬 곱셈 구조를 설명하고 4장에서는 실험 및 결과 그리고 5장에서 결론으로 구성된다.

II. 뉴럴 네트워크 가속기 구조

LBF에 사용되는 뉴럴 네트워크 가속기를 그림 3과 같은 구조로 Verilog HDL를 이용하여 설계했다. 가속기는 곱셈기와 덧셈기로 구성되며 곱셈기는 Modified Booth 알고리즘을 이용하여 8비트 구로 설계했다. 36개의 곱셈기로 뉴럴 네트워크 가속기를 구성했다.[6]

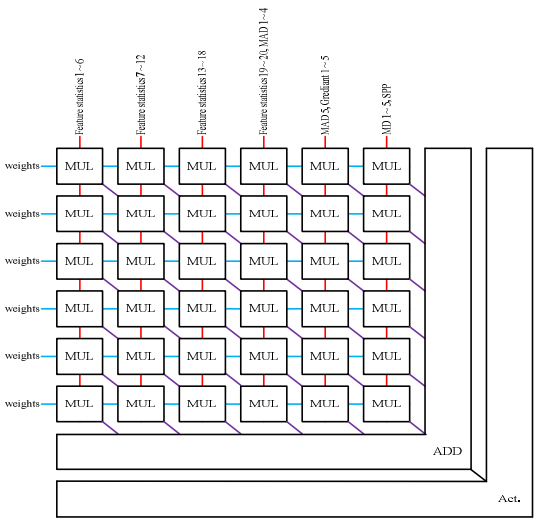


그림 3. 뉴럴 네트워크 가속기의 구조
Fig 3. Structure of Neural Network Accelerator

히든 레이어의 뉴런을 계산하기 위한 연결이 36개가 필요하기 때문에 곱셈기를 36개 구조로 선택한 것이다. 하나의 픽셀 노이즈를 제거하기 위해 필터 파라미터를 구하는데 걸리는 시간을 계산하였다.

CNN에서 Convolution은 입력신호에 특정형태의 필터(커널)를 씌워 결과를 얻는 그림 4와 같은 방법을 말한다. Pooling은 Max Pooling은 인접한 유닛들 중 가장 큰 값을 내보내고 Average Pooling은 인접한 유닛들의 평균값을 내보낸다. 이러한 특징으로 인하여

Matrix Multiplication의 연산이 90% 이상의 비중을 차지하게 되며 연산처리 속도에 많은 영향을 미친다.

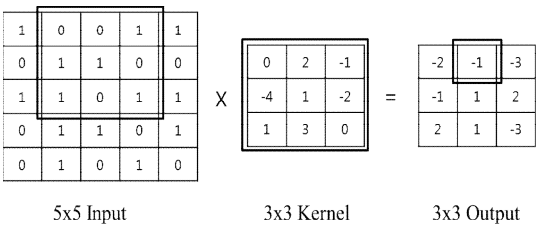


그림 4. 입력과 커널(필터)의 곱셈
Fig 4. Multiplication of Input and kernel(filter)

Verilog-HDL로 설계된 뉴럴 네트워크 가속기의 처리속도를 개선하기 위해 고속의 병렬 계산에 특화된 프로그래밍 모델인 Open Computing Language (OpenCL)을 이용했다. SW 개발자가 적응하기 쉽고 이식의 용이성과 유지보수에 이점이 있는 OpenCL로 GEMM(General Matrix Multiplication)을 수정한 행렬 곱셈(Matrix Multiplication)을 구현했다.

III. 개선된 행렬 곱셈 구조

OpenCL을 이용해서 연산속도를 높이기 위하여 행렬 곱셈 방식을 새롭게 적용하였다.

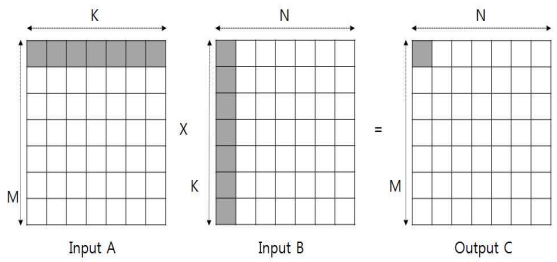


그림 5. 일반적인 행렬 곱셈 구조
Fig 5. General Matrix multiplication structure

그림 5의 행렬 구조에서 결과를 얻기 위해서는 입력 A의 첫 번째 행(column)과 입력 B의 열(row)의 값을 곱해야만 C에 결과가 저장 된다. 그로인해 많은 연산 시간이 소요되며 또한 메모리 접근 지연에 따른 성능저하까지 추가 된다. 코드의 형태는 그림 6과 같다. M과 N의

```
__kernel void GEMM (const int M, const int N,
                    const int K,
                    const __global float * A,
                    const__global float * B,
                    __global float * C) {
   const int globalRow = get_global_id(0);
   const int globalCol  = get_global_id(1);
   float sum = 0.0f;

   for(int k=0; k<K; k++) {
       sum += A[k*M+globalRow] * B[globalCol*K+k];
   }

   C[globalCol*M+globalRow] = sum;
}
```

그림 6. 일반적인 행렬 곱셈 코드
Fig 6. General matrix multiplication code

루프 인덱스 m, n은 스레드 식별자 globalRow와 globalCol로 대체되었으며 프로세서가 M*N번 실행하게 된다.

이를 개선하고자 행렬 곱셈을 병렬 처리하다도록 하였다.[7][8][9] 한 행씩이 아닌 임의의 블록 크기로 설정하여 곱셈하여 연산시간 및 메모리 접근 지연을 최대한으로 줄이도록 하였다.

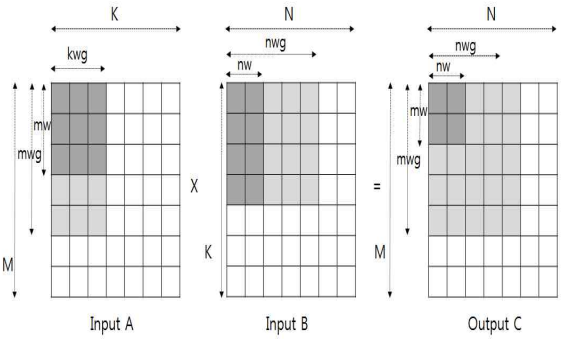


그림 6. 병렬 행렬 곱셈 구조
Fig 6. Parallel matrix multiplication structure

M에 대한 병렬 처리를 첫 번째 차원의 스레드에 할당하면 성능이 개선된다. 후속 스레드가 행렬 A와 C의 후속 데이터 요소에 액세스하여 메모리 요청을 할 수 있기 때문이다.

```
__kernel __attribute__((reqd_work_group_size(WGS)))
void Xaxpy(const int n, const dtype alpha,
          const __global dtypeV * restrict xgm,
          __global dtypeV* ygm){
      #pragma unroll
      for(int w=0; w<WPT; ++w)
      {
            int i = w*get_global_size(0) + get_global_id(0);
            ygm[i] = ygm[i] + alpha * xgm[i];
      }
}
```

그림 7. 병렬 핼렬 곱셈 코드

Fig 7. Parallel matrix multiplication code

IV. 실험 및 결과

실험은 뉴럴 네트워크 가속기를 OpenCL로 구현하여 하나의 픽셀의 노이즈를 제거할 때 사용되는 필터 파라미터를 구현하는데 걸리는 시간을 계산하였다. 그림 8에서 적은 수의 레이를 사용하여도 노이즈가 제거되는 결과를 얻었다.

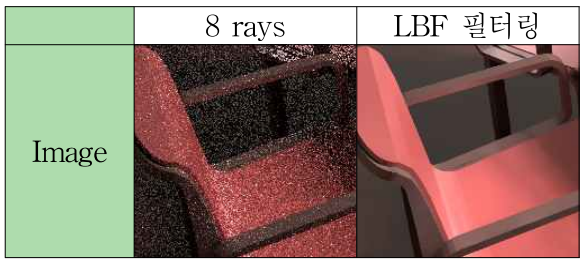


그림 8. 이미지 처리 결과

Fig 8. Image of processing result

실험은 SDAccel개발환경에서 OpenCL 코드 컴파일한 후 합성결과를 Xilin사의 Kintex UltraScale XKU690T-2FDFG1157C FPGA 보드에 적용하였다. 한 픽셀에 사용되는 필터 파라미터를 계산하는데 소요되는 시간은 10.26us로 Verilog HDL로 설계한 것 보다 약 1.12배 빠른 처리 속도를 보였다.

표 1. 픽셀당 연산 처리 시간 비교

Table 1. Comparison of processing time per pixel

	Processing Time
Existing GPU	12.41us
Verilog HDL Accelerator	11.44us
Proposed MM Accelerator	10.26us

표 1은 Verilog-HDL로 설계 되어진 가속기의 속도보다 OpenCL을 이용한 가속기의 실행시간이 단축되어 개선되었음을 확인하였다.

V. 결 론

본 논문에서 적은 수의 레이를 사용하여 레이트레이싱 렌더링 노이즈를 제거 할 때 사용되는 필터파라미터를 계산하는 뉴럴 네트워크를 OpenCL로 구현하였다. 뉴럴 네트워크 연사 속도를 빠르게 하기 위해 GEMM을 수정한 Matrix 곱셈기를 구현하여 연산 속도를 높였다. 실험은 Xilinx XKU690T-2FDFG1157C FPGA를 사용했다. 차후 연구로는 FFT(Fast Fourier transform)은 기본의 Matrix Multiplication 보다 효과적으로 곱셈 결과를 얻을 수 있다는 연구 이론들이 발표되고 있다. 아직까지 Hardware로 설계된 사례가 없기에 FTT기반 고속 가속기를 구현 할 예정이다.

References

[1] Rousselle, Fabrice, Claude Knaus, and Matthias Zwicker, "Adaptive rendering with non-local means filtering" ACM Transactions on Graphics (TOG) 2012.

[2] Li, Tzu-Mao, Yu-Ting Wu, and Yung-Yu Chuang, "SURE-based optimization for adaptive sampling and reconstruction", ACM Transactions on Graphics (TOG) 32.6 (2012): 194.

[3] Rousselle, Fabrice, Marco Manzi, and Matthisa Zwicker, "Robust denising usig feature and color inforamtion", Computer Graphics Forum, Vol. 32, No. 7, 2013.

[4] Sen, Pradeep, and Soheil Darabi, "On filtering the noise from the random parameters in Monte Carlo rendering", ACM Trans, Graph, Vol.31, No.3, 2012.

[5] Sangil Lee, Kihun Nam, Junmo Jung, "Implementation of handwritten digit recognition CNN structure using GPGPU and Combined Layer", JCCT, Vol.3, No4, pp. 165-169, Nov. 2017.

[6] Kihun Nam, "Implementation of Neural Network Accelerator for Rendering noise Reduction", ikeee, Vol.21, No4, pp. 420-425, Dec. 2017.

[7] Aravind Vasudevan, Andrew Anderson, David Gregg, "Parallel Multi Channel convolution using General Matrix Multiplication", In28th IEEE International Conference on Application-specific Systems, Architectures and Processors, ASAP 2017, pp. 19-24, July. 2017.
[8] K. Matsumo, N. Nakastio, and S.G. Sedukhin, "Performance Tuninf of Matrix Multiplication in OpenCL on Different GPUs and CPUs", In SC Companion;High Performance Computing, Netwoking Storage and Analysis. IEEE, pp. 396-405, 2012.
[9] Kwang Nin Nam, Yong Jin Jeong, "Cascade CNN with CPU-FPGA Architecture for Real time Face Detection", JIKEE, Vol.21 No.4, pp. 388-396, Dec. 2017.

※ This work was supported by Seokyeong University in 2018.