GPU Computing Project

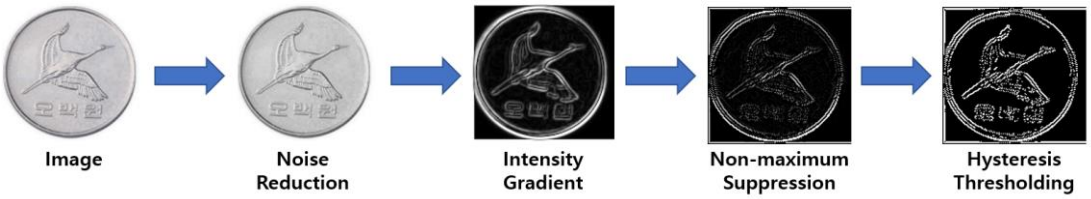
-Canny Edge Detection

1. 프로젝트 개요

Canny Edge Detection은 Image의 Edge를 검출하는 대중적인 알고리즘이다. 위 프로젝트는 CPU 기반의 Canny Edge Detection을 GPU를 활용해 가속하는 프로젝트로, 제공되는 C언어 기반으로 구현된 Canny Edge Detection의 5개의 함수를 CUDA로 구현해 기존 CPU 코드보다 수행시간을 단축시키는 것을 목표로 한다.

2. 배경지식

Canny Edge Detection은 Image의 Edge를 검출하는 대중적인 알고리즘으로 Noise Reduction, Intensity Gradient, Non-maximum Suppression, Hysteresis Thresholding 과정을 거쳐 Image의 Edge를 검출한다(Figure 1).



<Figure 1>

A. Noise Reduction

Noise가 있을 경우 Edge 검출이 어려워 Blurring 작업이 필요하다. Canny Edge Detection은 Blurring을 하기 위해 Gaussian blur를 사용한다. Gaussian blur는 Equation 1과 같은 식을 통해 5x5 Fillter를 구성하고, 구성한 Gaussian Fillter(F)와 Image의 2D Convolution 연산을 통해 Blurring을 수행한다.

$$F(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

<Equation 1>

B. Intensity Gradient

Image의 Pixel이 Edge인지, 아닌지 검출하기 위하여 인접 Pixel과의 Gradient를 구하고, 어떤 방향의 Gradient인지 구해야 한다. 이를 위하여 Sobel Filter(Figure 2)와 Image를 2D Convolution 연산해 각 픽셀의 x,y축 방향의 Gradient를 연산하고, Equation 2를 통해 Gradient 값과 방향을 구한다.

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

<Figure 2>

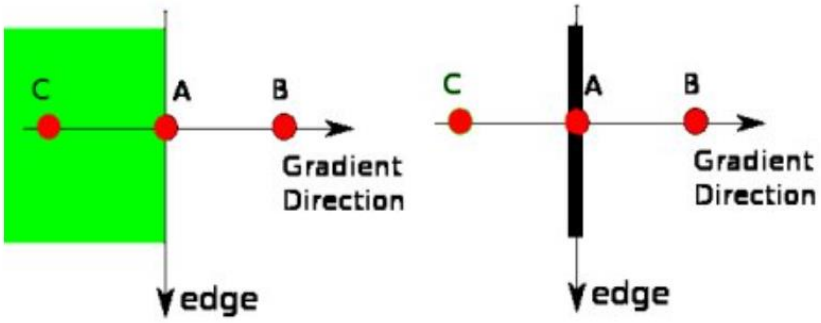
$$Edge\_Gradient(G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle(\theta) = tan^{-1}(\frac{G_x}{G_y})$$

<Equation 2>

C. Non-maximum Suppression

Gradient를 구한 후, Edge가 아닌 Pixel들을 제거하기 위해 Gradient 방향의 주변 Pixel들의 값을 비교하여, 그 중 가장 큰 Pixel값을 제외하고 모두 0으로 바꾸는 연산을 수행한다. 이를 통해 Gradient 방향의 Pixel 중 가장 큰 Pixel을 구할 수 있다. 예를 들어, Figure 3에서 A와 C, B를 비교해 A가 클 경우 그 값을 유지하고, 그렇지 않을 경우 0으로 값을 바꾼다.

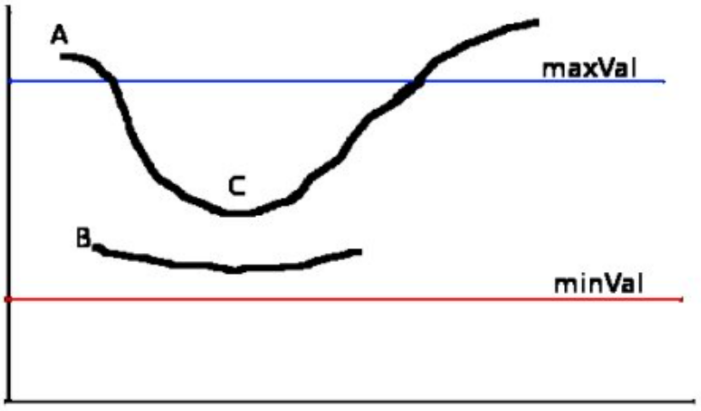


<Figure 3>

D. Hysteresis Thresholding

Hysteresis Thresholding은 2가지 단계인 Double Thresholding과 Hysteresis로 나뉜다. Double Thresholding은 2가지 Threshold 값인 maxVal와 minVal을 통해 검출되어 있는 Edge를 강한 Edge와 약한 Edge로 나누는 연산을 수행한다(Figure 4). 이 때 maxVal보다 큰 값을 가지는 Edge를 강한 Edge, maxVal과 minVal 사이의 값을 가지는 Edge를 약한 Edge로 판단한다.

Hysteresis은 Double Thresholding을 통해 연산한 약한 Edge가 강한 Edge와 연결되어 있을 경우 강한 Edge로 바꾸고, 그렇지 않을 경우 Edge를 제거하는 연산을 수행해 최종 Edge를 검출한다.



<Figure 4>

3. 프로젝트 설명

위 프로젝트는 제공된 C언어 기반 Canny Edge Detection 코드의 5가지 함수(Gray Scale, Noise Reduction, Intensity Gradient, Non-maximum Suppression, Hysteresis Thresholding)를 CUDA로 구현해 수행시간을 감소시키는 프로젝트이다.

각 함수의 수행시간 측정은 제공되는 소스코드 내 함수 및 출력을 통해 이루어지며, 채점 환경은 Colab을 이용해 채점한다. 또한 Input Image는 제공되는 이미지를 사용하고 주어지는 소스코드 중 이름이 GPU_로 시작하는 함수 외 함수들은 수정하지 않는다.

점수는 각 함수별로 채점되고, CPU 함수의 결과와 GPU 함수의 결과가 다르거나 GPU 함수가 CPU 함수에 비해 속도가 느린 경우 틀린 것으로 간주한다. 또한 5개 함수를 모두 구현한 사람 중 GPU 함수의 수행시간이 가장 짧은 10명에게 점수를 부여한다.

4. 보고서

A. Introduction

프로젝트에 대한 전반적인 설명(5~10줄 이내)

B. Background

GPU의 구조 및 CPU와 대비 GPU에서의 연산 시 장점 및 단점에 대해 서술

C. Code explanation & Optimization

구현한 CUDA 사용 함수에 대한 설명

Kernel의 연산을 최적화하기 위해서 어떤 방법을 사용했는지, 코드에서 Block 및 Thread의 수를 정한 이유와 Memory를 어떻게 사용했는지 등 **자세히** 서술

D. Result

결과화면을 삽입하고, 수행속도가 빨라지거나 느려진 이유를 분석해 서술

E. Conclusion

프로젝트를 구현하며 어려웠던 것, 배운 것, 문제가 되었던 부분과 어떻게 해결했는지

등을 서술

5. 배점

A. 보고서(40)

i. Introduction(5)

ii. Background(5)

iii. Code explanation & Optimization(20)

iv. Result(5)

v. Conclusion(5)

B. 코드(60)

i. Gray Scale(10)

ii. Noise Reduction(10)

iii. Intensity Gradient(10)

iv. Non-maximum Suppression(10)

v. Hysteresis Thresholding(10)

vi. Execution Time이 가장 짧은 10명(10)

C. 제출기한 및 양식

i. 제출기한 : 11월 23일(수) 오전 9시

ii. 양식 : 소스코드 파일과 이미지 파일 및 보고서(PDF파일)을 압축해 **Project_학번_이름**으로 제출
예시) Project_2015722030_박우혁