Parallelising Image transformations using OpenCL

Utkarsh Gautam, Yogesh Kumar

140050009, 140050004

May 6, 2018

Abstract

Many image transformations such edge detection, image sharpening etc. are essentially image convolution with different masks. Images can be partitioned into multiple patches and combinations of the convolutions of the patches is same as convolution of the entire image. We aim to use this property to speed up the convolution transformation for an image. We plan to use OpenCL and OpenCV in achieving the same. At the end we make performance comparison of serial and parallel exec

1 Introduction

Image transformation algorithms form the building blocks of image processing and computer vision. Convolution is a widely used method in image processing. Speeding up the convolution of an image with a mask, can lead to improvement in computational times of many image image processing algorithms like, edge detection, blurring etc.

Convolution operation can be very computationally expensive for large size images or real time image transformation. Since convolution applied on each pixel is independent of other pixel operations, this leaves huge scope for independent computations being done in parallel to improve the overall time. We have explored with multiple image processing algorithms.

We have used OpenCL for parallelization to make the code compatible on all GPU devices (unlike CUDA)

2 Experiment Details

We have experimented with three popular algorithms. In each case, we have a serial code and parallel code to do the same operation. We have used images of multiple sizes and computed the execution time.

2.1 rgb2gray

One of the most fundamental algorithms of image processing of converting a RGB image to grayscale image. We have used this particular formula, Y'=0.299R'+0.587G'+0.114B' for conversion though we know many other formulas exist too.

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

Figure 1: Gaussian Blurring Mask

2.2 Gaussian Blurring

An algorithm of blurring an image by a Gaussian function (named after mathematician and scientist Carl Friedrich Gauss). It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The operation involved convolving image with gaussian blurr mask. Thought there are various variations of mask depending upon the variance of the gaussian, we have particularly used the one one the left in figure 2.

2.3 Gaussian Edge Detection

An algorithm to find edges in an image. The operation is convolving image with gaussian edge detection mask. The operation involved convolving image with gaussian blurr mask. Thought there are various variations of mask depending upon the variance of the gaussian, we have particularly used the one in figure 3

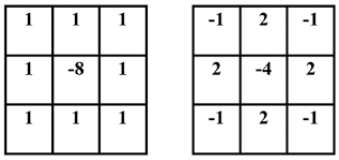

Figure 2: Gaussian Edge Detection mask

3 Results

Following are the computation time for each algorithm.

Image Size (in mb) / Algorithm	0.5	1	10	40
Serial rgb2gray	0.06238	0.1123	2.0568	3.4157
Serial gaussian blur	1.8490	3.8093	69.0476	111.4697
Serial gaussian edge detection	0.1947	0.39941	7.45562	11.9095
Parallel rgb2gray	0.0011	0.0014	0.01736	0.02435
Parallel gaussian blur	0.00099	0.00119	0.01237	0.01705
Parallel gaussian edge detection	0.00109	0.00142	0.01883	0.02835

Table 1: Time taken for each algorithm in seconds

4 Conclusion

We observe that with increasing image size (and increasing computational task) parallel algorithms perform way better than their serial parts. This has aligned with our intuition.

5 Future Work

Convolution operation presented here is restricted to 2D filters. It can very easily be extended to 3D filters which can convolve over multiple channel images. This can be further extended to faster neural network implementations in machine learning domain.