GPGPU - Project report

Nicolas Portal - Alexandre Yvart - Geoffrey Jount

?today?

?contentsname?

1 Introduction

For educational purpose in the GPGPU subject, we are asked to work on a project using a GPU parallel method. We will use the CUDA library.

2 Problem description

This project's goal is to implement an image segmentation algorithm : the max-flow/min-cut algorithm. A picture containing numerous pixels however, processing one after another in a simple `for` loop would show to be very slow. We thus need to make the process parallel, and implement it on GPU for an even faster result : the *push-relabel* algorithm, parallel version of the first mentioned, is what we seek.

3 CPU implementation

We followed the guidelines found on the NVIDIA presentation.
All in all there are height matrix: One for the heights of each node.
One for the excess flow of each node. One for the links between the source and each node. One for the links between the sink and each nodes.
Finally, there are four matrix for the links between each nodes.
The four matrix for top, bottom, right and left neighbours were labeled with the inverse of the euclidean distance in RGB space.
The links between the source and each node and the links between the sink and each node were labelled with the inverse of the average
distance to the marked pixels in RGB space. Each node were linked to the source and sink.
The height of each node was 0 except for the source which had a height equals to the number of nodes.
The stb$_l oadlibrarywasusedtoloadjpgimagesbothingrayscaleandRGBcolorspace.$
$Imageswerestoredasonedimensionalvectorswithcontiguousred, greenandbluepixels.$
$thesizeofthesevectorswerethreetimesaslargeasthenumberofpixelsinanimage.$
$Theppmfileformatwasusedtogeneratetheoutputsegmentedimage.$

4 GPU implementation

5 Bottlenecks 1

Our implementation soon ran into an infinite loop which remained one of the core issue for several weeks. As a result, the code was changed to make it possible for each node to push towards the sink or the source. This did not solved the issue. The weights of links between the source ans sink on one hand and every node on the other hand was changed using the histogram on red, green and blue channel of the marked pixels to compute a probability. This did not solved the issue as well.

5.1 Enhancements

It was found that there was actually no infinite loop but the algorithm was very long to terminate as two nodes would often exchange flow between them waiting for one of them to reach the maximum height. To solve this glitch, it was decided to manually lower the maximum height. The maximum height was no longer set to the number of nodes but rather to a low number such as five or ten. This made it possible for our algorithm to terminate.

6 Bottlenecks 2

An other issue quickly appeared. Using a depth first search to find nodes yielded bad results as every pixel would be set to white.

6.1 Enhancement

When it comes to the way to get the output image, it was decided to set every pixel with a height above zero to one. This did not generate optimal results but it was good enough to make out the segmentation.

6.1.1 Justification

A small eight by eight image was used to make the debug step simpler. This image was a ppm image with every pixels on the top part of the image set to blue and every pixel on the lower part of the imge set to red. One pixel was set to black and white in each region.

7 Bottlenecks 3

Although the maximum height was decrease to five or ten, the run time was still to high. It took 15 to 20 minutes for the algorithm to terminate.

7.1 Enhancement

Lowering the maximum height resulted in a 15, 20 mn running time. This was still too high. It was found out that calling the push and relabel function on every nodes would decrease the run time significantly. The check for active nodes was now done by the push and relabel function. It was no longer needed to write a separate function to retrieve the right active nodes. It was also no longer required to loop on every pixels to find active nodes. This decreased the run time to 10s which was a huge improvement.

8 Description

9 Performance comparisons

10 Benchmarks recapitulation

11 Tasks repartition

Nicolas
- CPU implementation and debug
- report

Alexandre
- CPU debug
- GPU implementation

Geoffrey
- CPU debug
- Benchmarking implementation