High Performance Scientific Computing
Yue Zhang
Dec,21 2012

Final Project Report: Seam Carving

The topic of project is Horizontal Seam Carving. This is a program can resize the width of image without losing important data or causing distortion. Seam Carving is invented by Shai Avidan and Ariel Shamir from Mitsubishi Electric Research Laboratories (MERL), and developed by many programmers around the world. The coder's main source is Avidan and Shamir's paper: Seam carving for content-aware image resizing[1] and the Seam Carving page on [2]. The main purpose of this program is to resize wide image to a proper narrower image. After reducing the width of picture, the generated picture should delete the unimportant part of the image and keep the shape and color of the most important component, like human, building, animal in the graph. The interval of image width is [128, 8192]. The interval of image height is [4,1024]. Since the program is used for wide image, wider image is preferred. The lower bound of width is based on the work group size, which will effect the efficiency of the program. The upper bound of the width is decide by the time consuming: if it is larger than the upper bound, then the program can took a long time to finish the work. The lower bound is 4 because less than that there will be no dynamic seam calculation and the program becomes meaningless. Since the program works with wide image, height greater than 1024 can also cause a long time to process. So the upper bound of height is decided based on the efficiency of the program.

There are some OpenCV, Java, and OpenGL codes doing Seam Carving on Internet. There are also programs available to do similar image-resizing. This program does not consult those codes or programs because it is based on C language and OpenCl language, using paralleling computing. The expectation for this program is to use GPU doing parallel computing to do seam carving as quickly as possible. The step to calculate seams is hard to parallel since each row of the image matrix depends on the former row. So it is hard to parallel this dynamic programming part on the whole matrix. Thus, this part is also the one cost most time.

Here are some results of efficiency. It is calculated by x = width*height*delete_seam/processing time.

```
31452460.168107 of pixel/s, generate size 2000x1078 to 1000x1078
28490629.925246 of pixel/s, generate size 4288x1429 to 3000x1429
32824069.328321 of pixel/s, generate size 4288x1429 to 2000x1429
38585203.248761 of pixel/s, generate size 4288x1429 to 1000x1429
38102112.171293 of pixel/s, generate size 5992x624 to 2000x624
43222954.303436 of pixel/s, generate size 5992x624 to 1000x624
```

As the program shrinks of the width more and more, the efficiency increases since the matrix becomes smaller as deleting the seams. As the width, and the height increase, it

[1] http://dl.acm.org/citation.cfm?id=1276390

[2] http://en.wikipedia.org/wiki/Seam_carving

should cost more time to delete one seam and efficiency should decrease. However, in fact, to delete same number of seam, as image increase the efficiency increases. Since applying parallel computing on GPU, the increase of size does not cause a lot change in efficiency.

In CPU part, the program first imports image data and new width. Then it calculates grayscale image by using the formula gray=0.299r+0.587g+0.114b, where r,g,b are red, green and blue pixel data. The CPU then sets context on GPU, builds queue, load kernels, allocates and transfers those data to GPU. In GPU, the kernels are used to calculate the gradient of each pixel in the image, calculate the seam energy, find the seam with the lowest energy, find the index of this seam in each row, and delete the seam. When calculating the gradient of each pixel, the program uses the grayscale image, and calculates gradient along x and y axis. The program parallels the computing on the whole matrix with group size of 128. In the next step, each time the kernel imports one row from top and one row from bottom in the gradient matrix. Then it calculates seam energy by applying dynamic programming, finding the shortest path from top to middle and from bottom to middle at the same time. In this kernel, the work group size is 128. In the informal version of the program, the coder tried to use local memory, imported the data and processes them. However, the coder found it did not improve the speed and slowed the program. So the coder edited it only used private and global data and it worked better. After the dynamic programming part, there are other kernels to deal with the middle row (or 2 middle rows if height is even), and generate the final seam energy for each seam. Then the next kernel finds the lowest seam, traces it back to find the index of this seam in each row, and saves the index. Then, the next kernel works on deleting seam will delete the seam by setting each row as one work group, deleting the seam by moving forward each pixel after the seam's row index by one. All those kernels loop to work until all seams needed to deleted are removed from the image. Then the last kernel will transfer the image data to the new image. Finally in the CPU, the program will import the new image data and produce this new image named “sc.ppm” with the new width. Then program will free all the memory used and end.

Instructions:

Please go to Forge[3] to download the data.
First, convert your image to .ppm file and put it in the same folder as the code.
In terminal, type “make seamcarving” and click return.
Then type “./seamcarving image-name.ppm new_width” and click return.
If the new width is longer than the original one the program will abort.
Then choose GPU for the program and it will started to work. The screen will display how many seams left to deleted. Finally it will display the efficiency of resizing the image.

[3] http://forge.tiker.net/p/hpc12-fp-yz1386/source/tree/master/

Use “display sc.ppm” and the new image will be displayed. Or find “sc.ppm” in the folder and open it directly.

Example:

In the terminal, first type “make seamcarving”. Then click return.
Type “./seamcarving fatcat.ppm 600” and click return.
Choose GPU like AMD and click return.
Type “display sc.ppm” and click return (or open the folder and click to open the image sc.ppm).