GPU and GPGPU Programming

Shuai Lu 170742

Assignment 5

Task
1. Matrix, Vector Operations and Reduction
You have to program the matrix and vector operations (matrix-vector multiplication, vector-vector operations, and reduction) on the CPU. 1.b) Implement the same operations on the GPU.
Find the comments "TASK" in the code and implement them (one CPU version and one GPU version). For testing a small hard-coded 4x4 matrix, and .txt files containing sparse matrices are provided.

2. Improve the performance of the GPU implementation You must use shared memory for the vector in the matrix-vector-multiplication! For the reduction you must use shared memory as well (as discussed in the lecture). Measure the performance of CPU, unoptimized (without shared memory) and optimized (with shared memory) GPU versions.

3. Test using Image Deblurring Method Once the Conjugate Gradient method is working for the matrices it can be used to solve practical applications. A simple image de-blurring application is implemented that works for a known blurring filter as described in the following: An image is loaded and blurred with a filter kernel. The image can be de-blurred again by solving the equation Ax = b were x is the unknown input image, that was filtered with filter operation A such that the result is the known blurred image b. In order to formulate the blurring operation (convolution) as a matrix multiplication, b and x are represented as vectors and each row of matrix A is one filter kernel for the whole image x. This naive method is very memory inefficient ($O(N^2)$ where N is the number of pixels). Your task is to test if your GPU implementations (of tasks 1. and 2.) work and to measure the performance. Hint: Because of numerical inaccuracies for the very large matrices that occur you will have to increase the error tolerance for the CG-method to terminate.

BONUS:
1. Implement a sparse matrix format. 2. Modify the code to divide the image into small patches and solve for them separately. This will allow to run the de-blurring on larger input images.

Outcome:
The matrix-vector multiplication, vector-vector operations, and vector reduction are implemented both in CPU and GPU. The shared memory is used in the GPU implementation (the matrix-vector-multiplication and vector reduction) to improve the performance. The result of CG method when matrix size equal to 16 is shown in Fig.1.
The measurement of the run time of CPU, GPU and optimized GPU implementations are shown in Fig.2.
The GPU implementations also work for image deblurring. The outcome is shown in Fig.3. And the performance is shown in Fig.4.

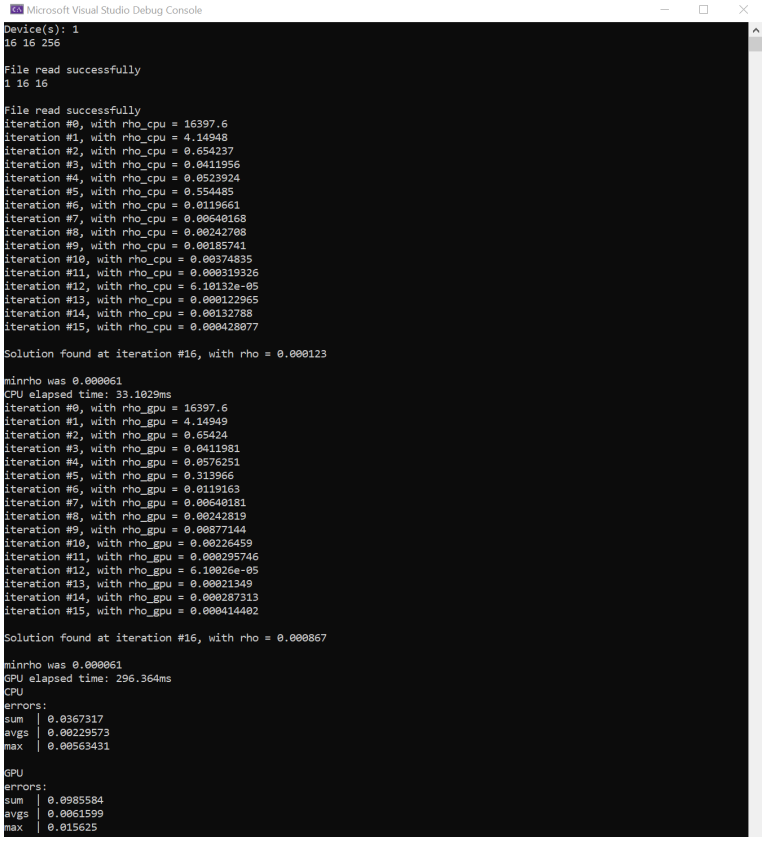

Figure 1: MatrixSet=16

MatrixSet=16	Time	Error_sum	Error_avgs	Error_max
CPU	34	0.0367317	0.0022957	0.00563431
GPU	309	0.0985584	0.0061599	0.015625
OptimizedGPU	258	0.103142	0.0064464	0.0177574
MatrixSet=64	Time	Error_sum	Error_avgs	Error_max
CPU	189	0.021759	0.00034	0.00128174
GPU	497	0.0204773	0.00032	0.00115967
OptimizedGPU	401	0.0295715	0.0004621	0.00195312
MatrixSet=200	Time	Error_sum	Error_avgs	Error_max
CPU	647	0.244141	0.0012207	0.00537109
GPU	1074	0.241699	0.0012085	0.00390625
OptimizedGPU	964	0.241699	0.0012085	0.00390625

Figure 2: Measurement of different implementations on different matrix sizes

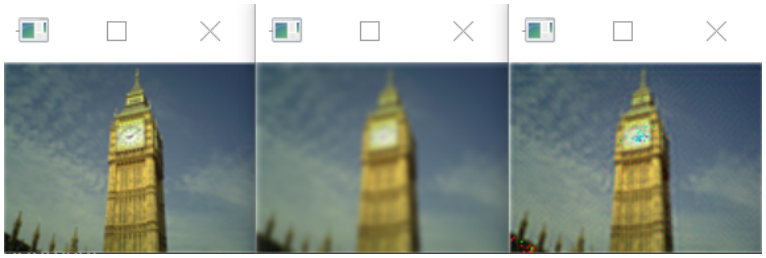

Figure 3: DeBlurring

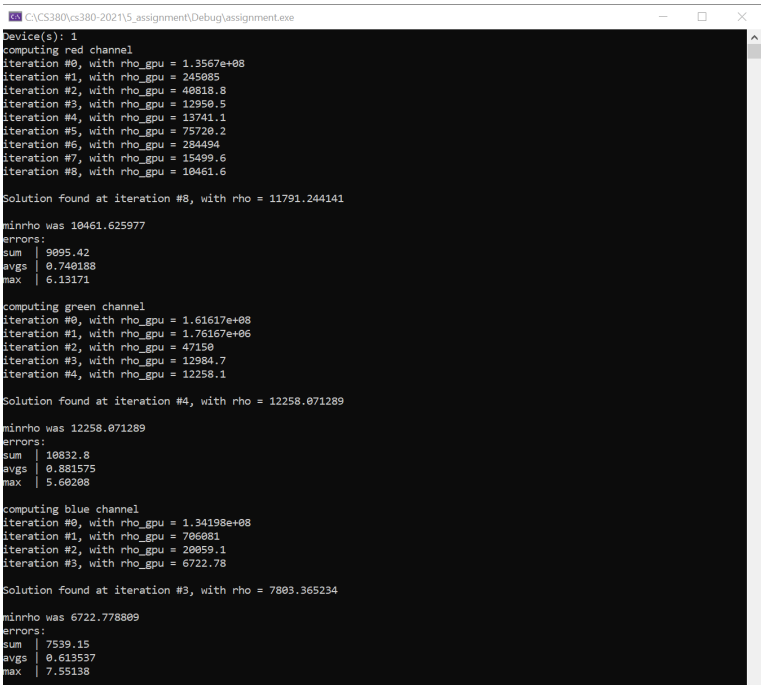

Figure 4: Performance