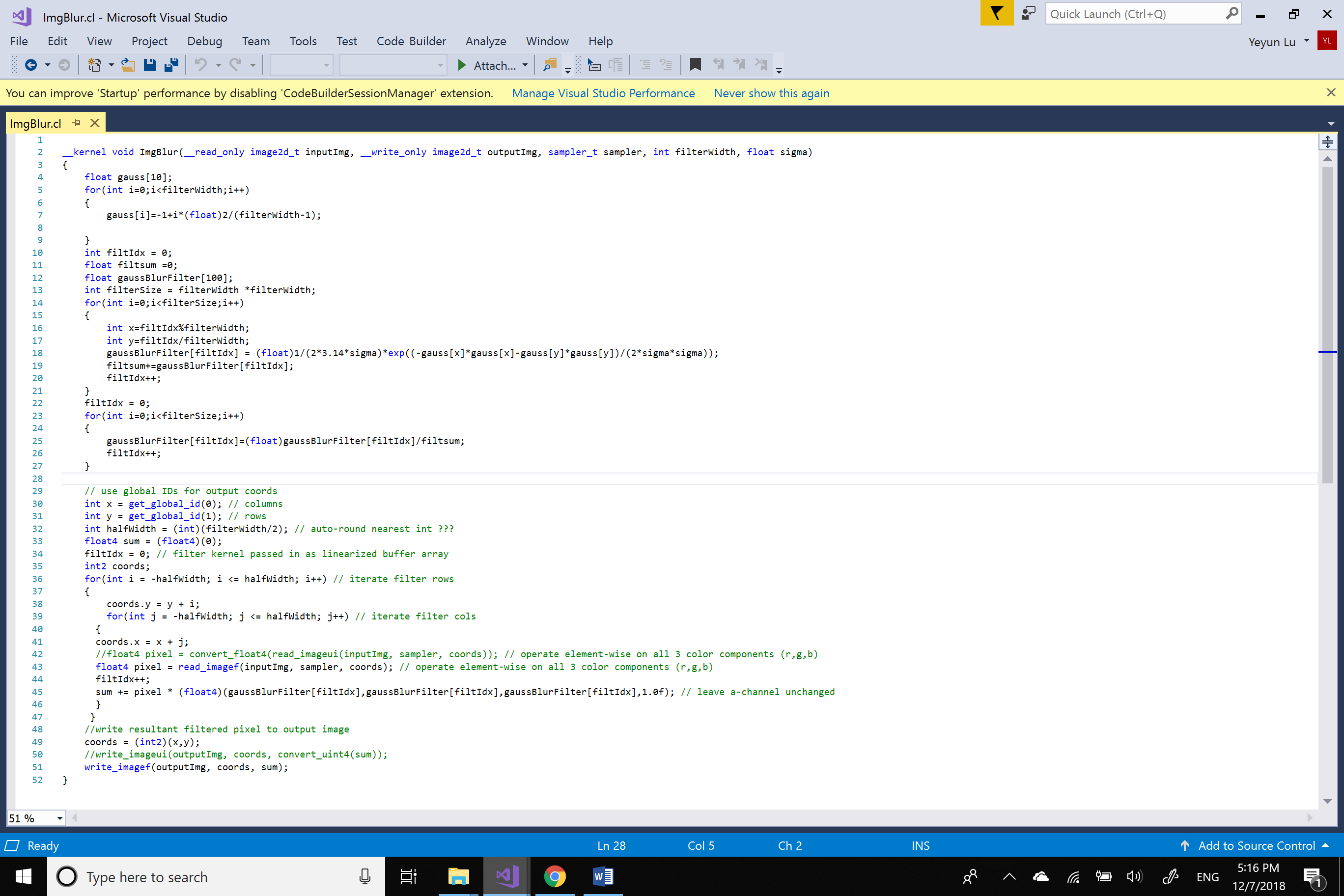
100 iterations

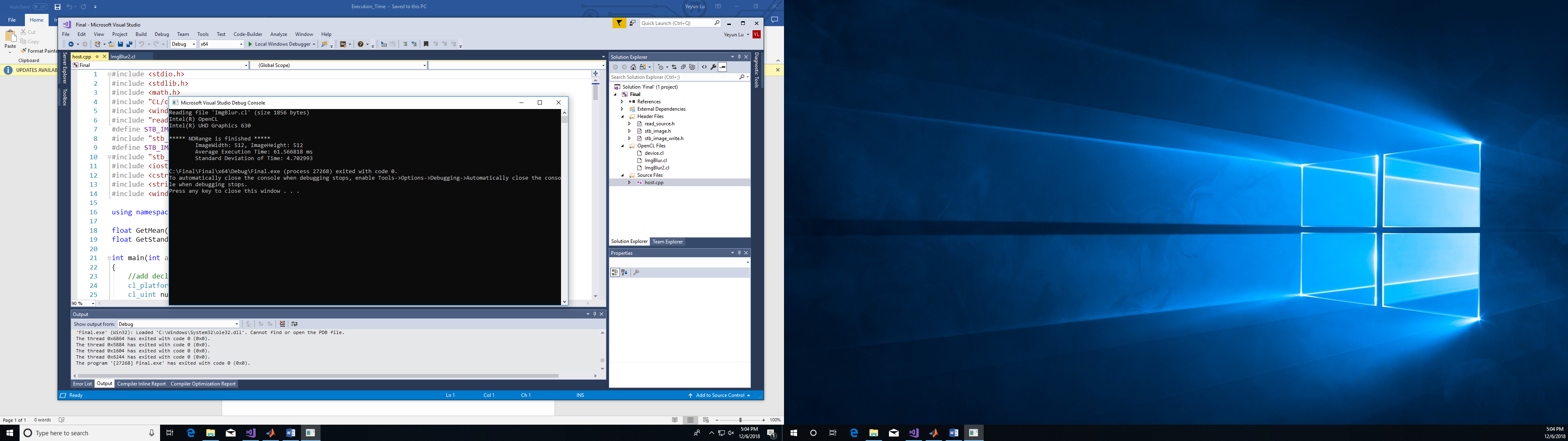
Produce gaussian matrix and process gaussian blur in one single kernel



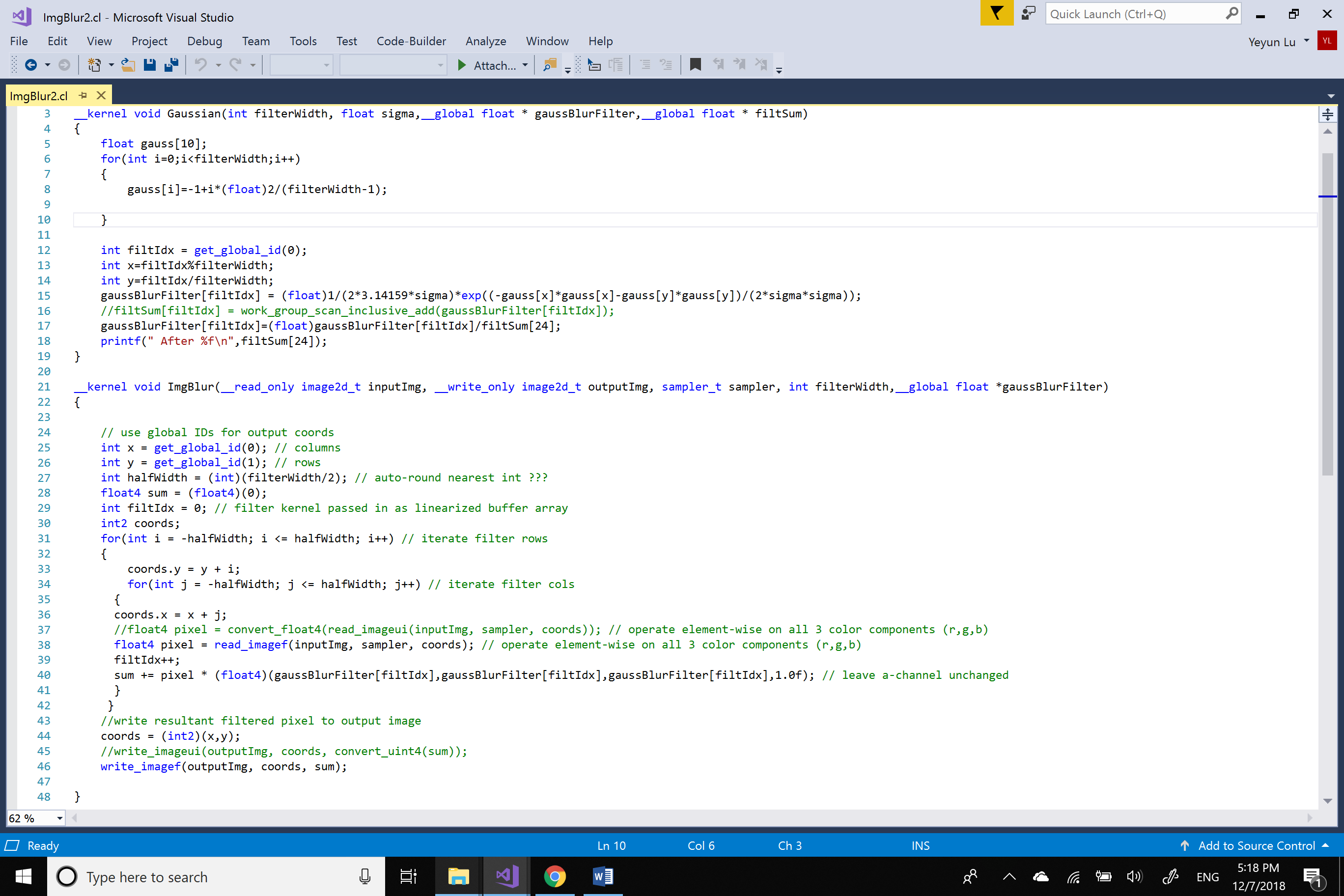
Gaussian Filter

+

Gaussian Blur

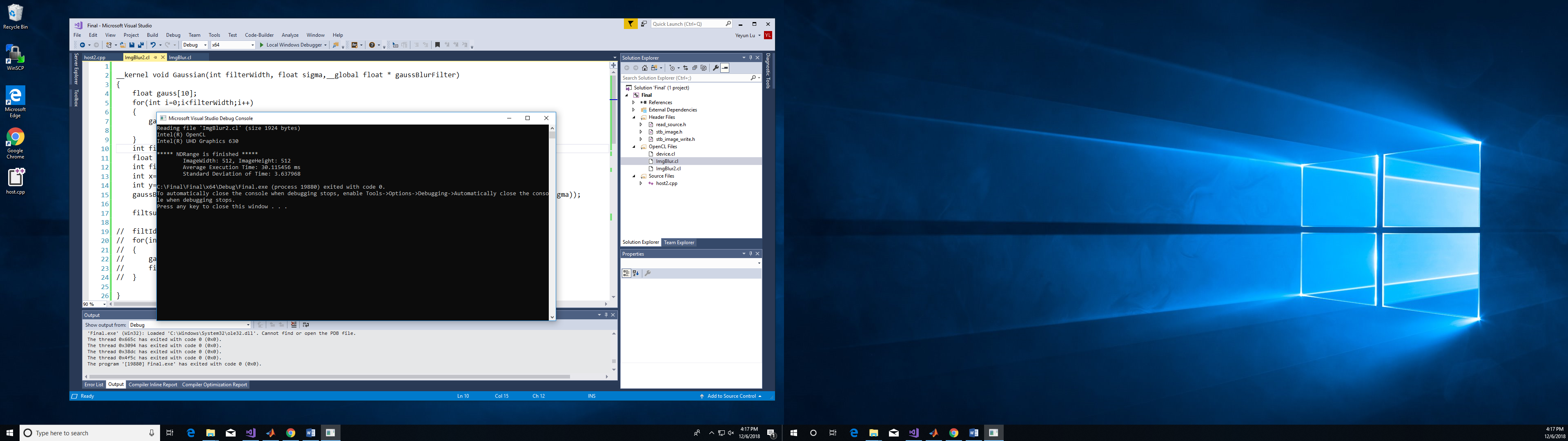


Parallelize gaussian matrix producing kernel as well as gaussian bluring kernel



Gaussian Blur

Gaussian Filter



At first, we test the efficiency of each kernel and then we consider how to combine them together.

Down sample kernel results: (using mean method)

Input Image:

 (512\*512)

Output Image:

(256\*256) (128\*128) (64\*64) (32\*32)

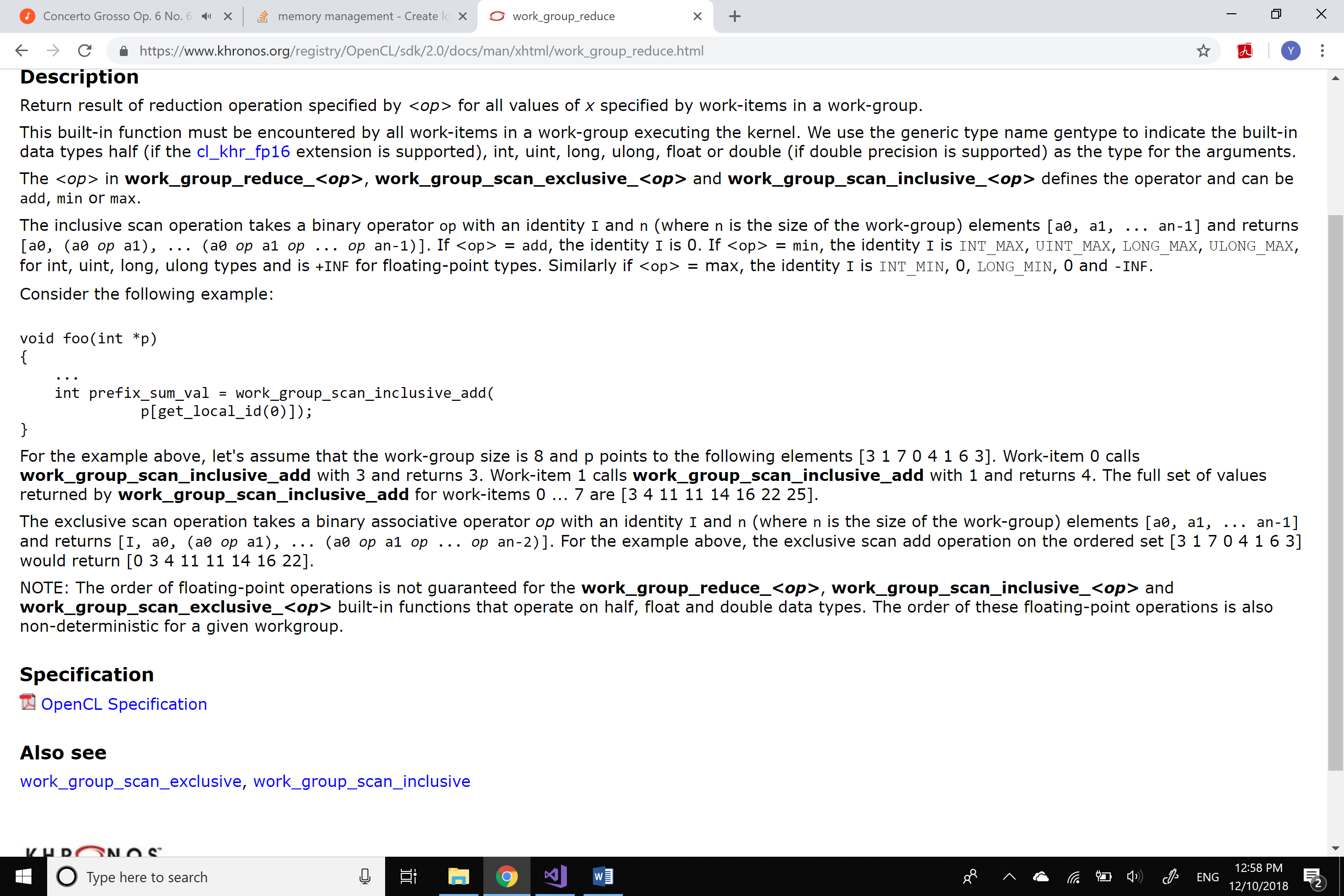
Guassian Blur kernel results:



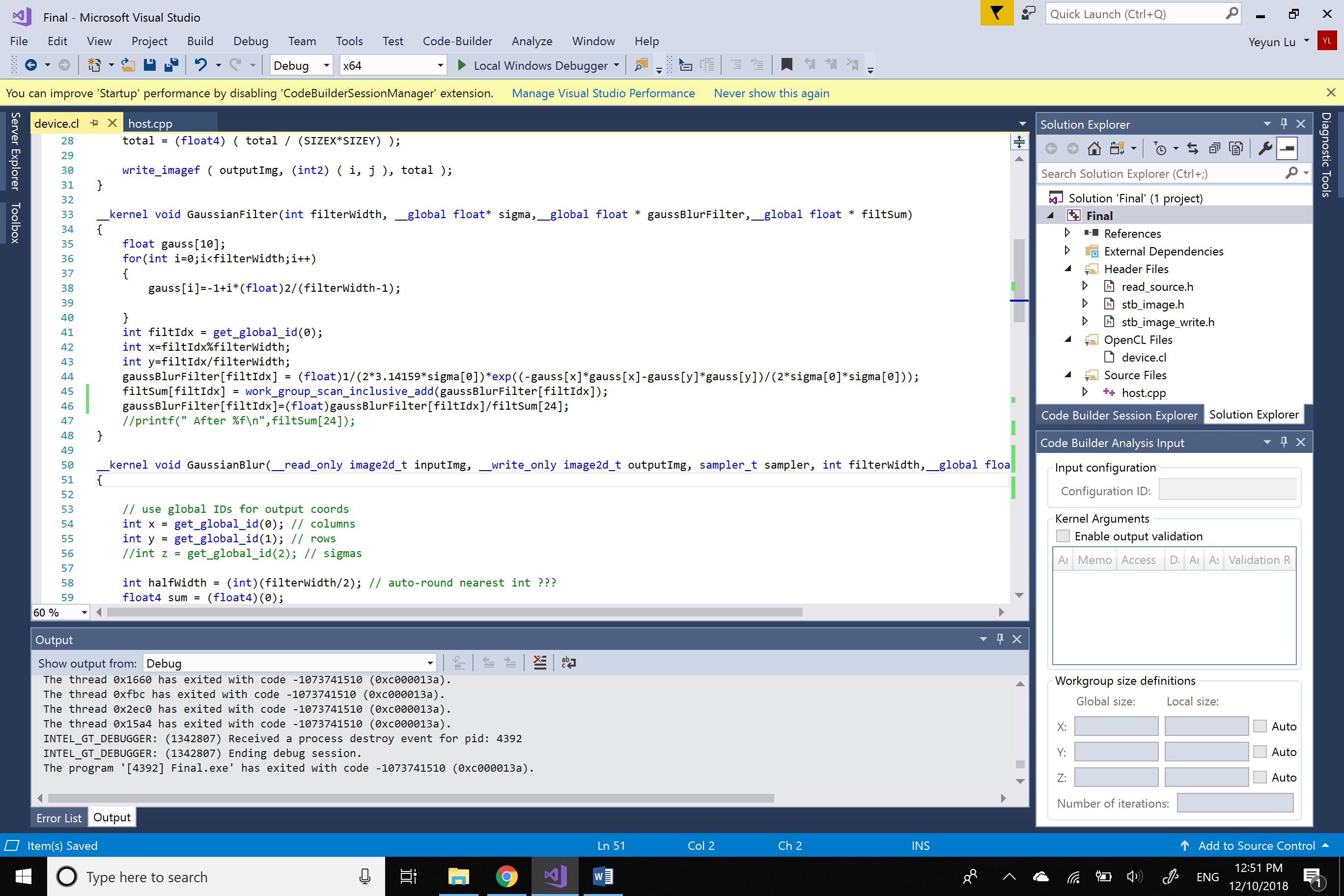
Sigma=0.7 sigma=1.4 sigma=2.8

Issue 1: how to normalize the gaussian filter?

The barrier function ( barrier(CLK\_GLOBAL\_MEM\_FEN) or barrier(CLK\_LOCAL\_MEM\_FEN) ) didn’t work because it only set barrier for different work items within one single work group. However, there are several work groups in our program.



Finally, we used work\_group\_scan\_inclusive\_add function to get the sum of all the elements in the Gaussian Filter and share this data between different work groups.



Issue 2: how to parallelize the Gaussian octave of Different of Gaussian octave producing?

Strategy a: use different commands and queues to parallelized kernels with different sigma.

Strategy b: use image2d\_array\_t to deal with several images with different sigma and use the third dimension (i.e. index = get\_global\_id(2); s=sigma[index];) to parallelized it.