Computer Graphics IV Tutorial

– Assignment 3 –

Computer Graphics and Multimedia Systems Group
Martin Lambers

Assignment 1 Simple Image Processing: Separable Filter Using Shared Memory

Extent the image processing program from the last two assignments by adding a third variant that uses shared memory to speed up the computation (see below for details). Use CUDA kernel timing functions to verify that this new variant of the filter performs better than the previous two variants.

In the separable filter variant implemented in the last assignment, concurrent threads access overlapping image data from global memory. Let us look at the horizontal step of the separable filter with a filter size of 5×5. The thread for pixel $(4, 7)$ will access the following pixels in global memory: $(2, 7), (3, 7), (4, 7), (5, 7), (6, 7)$. The thread for pixel $(5, 7)$ will access the following: $(3, 7), (4, 7), (5, 7), (6, 7), (7, 7)$. Four pixel accesses of both threads are identical.

The idea is to reduce these concurrent accesses to global memory by first transferring image data to shared memory. Since shared memory is accessible per block, each thread block should cache the data that is relevant for its operation.

We divide the horizontal step kernel into two parts: in a first part, a shared memory array is filled with the relevant image data, and in the second part the actual computation on this data takes place.

In the first part, with a block size of 32×32 threads and a filter size of 5×5, we need to cache 32×32 values, and the core image region to be processed is $32 - \lfloor 5/2 \rfloor - \lfloor 5/2 \rfloor = 28$ pixels wide: 2 pixels at the left and right border are excess pixels that overlap between neighboring tiles. A sketch helps to identify the necessary index computations.

In the second part, we can now run our horizontal filter loop on the core image region from $x - 2$ to $x + 2$ to access all pixel values from shared memory without checking for boundaries.

The vertical step is handled in a similar way.

Do not forget to use `__syncthreads()` at the appropriate place!