Assignment 4

In this assignment the Basic Algorithm is already given.

The Histogram Calculation has to be done on an image of size of the order Mb. Now in cases of >30Mb , computing this in python takes up too much time. So we do need an GPU implementation to speed up such a process. We use atomic operations to prevent writing to the same memory and thus preventing race conditions or contradictory results.

Kernel 1:So the algorithm implemented in order to speed up the basic algorithm performs computation using a temporary variable in the local memory. Total number of work groups is Size/Work items which in this case is 256.

To prevent random values in the execution we first initialize all the work items to zero, which Is followed by barrier synchronization to make sure that the values are copied first.

So, We have a local_id which the id of one work item In a work_group. The intention is that as this local_id gets incremented , it will move over the entire image , in sets of 256 which is 'i'. Thus eventually accounting for each pixel in the image. We use get_num_group to get the number of work groups present.

We then store this in the local memory in the temp variable. Once all the iterations are done, we add this finally into the hist in the global memory, thus memory is coalesced. This further improves computation speed.

Kernel 2: This is similar to Kernel, with the exception that Atomic_Inc is used instead of atomic_add. This results in slightly better speed.

The histo1 algorithm is faster than that of the Basic Algorithm at a matrix size of about 102 MB.

The histo2 algorithm is faster than that of the Basic Algorithm at a matrix size of about 102 MB. This is also marginally faster than the histo1 algorithm.

Reference: 1)Lecture slides and Videos on Histogram and Atomic operations

2)Worked in Collaboration with Maanit and Zoltan.