FAST IMPLEMENTATION OF ANISOTROPIC DIFFUSION USING THE CUDA TECHNOLOGY ON THE GPU GEFORCE 8800 GTX

by

Neha Sarjerao Patil

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the degree of

MASTER OF SCIENCE

Major Subject: Electrical Engineering

Approved:

_____

Dr. Badrinath Roysam, Thesis Adviser

Rensselaer Polytechnic Institute

Troy, New York

November, 2007

(For Graduation December 2007)

ABSTRACT

CUDA ("Compute Unified Device Architecture") is high level language for GPU programming. GPU Computing with CUDA on the GeForce 8 series is a new approach to computing where hundreds of on-chip processors simultaneously communicate and cooperate to solve complex computing problems up to 100 times faster than traditional approaches. A CUDA-enabled GPU operates as either a flexible thread processor, where thousands of computing programs called threads work together to solve complex problems, or as a streaming processor in specific applications such as imaging where threads do not communicate. CUDA-enabled applications use the GPU for fine grained data-intensive processing, and the multi-core CPUs for complicated coarse grained tasks such as control and data management. We use it here for Image processing algorithms called anisotropic diffusion to achieve a faster implementation of it. It is well suited to address problems like anisotropic diffusion that can be expressed as data- parallel computations – the same program is executed on many data elements in parallel.

The anisotropic diffusion algorithm is implemented using the AOS (additive spilt operator) approach as describe by Weickert. This semi-implicit scheme is more robust than the explicit schemes and has some parallelism in it. The algorithm has been developed on the GPU GeForce 8800 GTX, spanning all the available memory spaces. This includes the global, shared memory, constant and registers. GeForce 8800 makes use of CUDA technology which is unified that is a single floating point shader core with multiple independent processors is capable of handling any type of shading operation depending upon the workload of the application. This gives unprecedented performance and efficiency.

The algorithm is broken into small modules which are then executed on the GPU processors in the form of kernels. Each kernel forms a grid space which consists of number of blocks which perform the same function. The blocks consist of group of threads which run in parallel as long as there is no thread communication required. Incase there is some communication the instruction would then be executed in parallel and this will then affect the speed and hence the performance. Hence efforts have to be taken by programmer to make efficient use of the available resources within the processor.