AMD Accelerated Parallel Processing TECHNOLOGY

SAMPLE

Region Growing Segmentation

1 Overview

1.1 Location

$<*AMDAPPSDKSamplesInstallPath*>\samples\opencl\cl\2.0

1.2 How to Run

See the *Getting Started* guide for how to build samples. You first must compile the sample.

Use the command line to change to the directory where the executable is located. The pre-compiled sample executable is at `$<AMDAPPSDKSamplesInstallPath>\samples\opencl\bin\x86\` for 32-bit builds, and `$<AMDAPPSDKSamplesInstallPath>\samples\opencl\bin\x86_64\` for 64-bit builds. Ensure that the OpenCL 2.0 environment is installed.

Type the following command(s).

1. `RegionGrowingSegmentation`
   This command runs the program with the default options.
2. `RegionGrowingSegmentation -h`
   This command prints the help file.

1.3 Command Line Options

Table 1 lists, and briefly describes, the command line options.

Table 1 Command Line Options

Short Form	Long Form	Description
-h	--help	Shows all command options and their respective meanings.
	--device [cpu|gpu]	Devices on which the OpenCL kernel is to be run. Acceptable values are cpu or gpu.
-q	--quiet	Quiet mode. Suppresses all text output.
-e	--verify	Verify results against reference implementation.
-t	--timing	Print timing-related statistics.
-v	--version	AMD APP SDK version string.
	--dump [filename]	Dump the binary image for all devices.
	--load [filename]	Load the binary image and execute on the device.
	--flags [filename]	Specify the filename containing the compiler flags for building the kernel.
-i	--iterations	Number of iterations.
-p	--platformId	Select the platformId to be used[0 to N-1 where N is number platform s available].
-d	--deviceId	Select deviceId to be used[0 to N-1 where N is number devices available].

Short Form	Long Form	Description
`-o`	`--hostq`	Host side enqueue of kernels.

2 Introduction

This sample shows the use of the device-side enqueue feature in OpenCL 2.0, in scheduling dynamically generated work by a device kernel on the device itself. This dynamic generation of work and re-scheduling it on the device is shown by an example of seeded region growing image segmentation. Given an image, the user provides seed pixels, each inside one region to be segmented out. The seed pixels are put in a queue. Starting with the seed pixels, the algorithm classifies already queued pixels into one of the regions, which is chosen based on minimum difference between pixel luma and average luma of the region. It then queues the unclassified neighbors of the queued pixels to be classified in the next iteration. With each iteration average luma of all the regions are also updated.

The algorithm is adapted from seeded region growing algorithm of Adams and Bischof[1].

This sample must be run in the OpenCL 2.0 environment.

3 Implementation

The host reads an input image "input.bmp", and a seed file called "seeds.txt". The top row of the seed file contains the number of regions into which the image is to be segmented, and rest of the rows contains seed pixels co-ordinates, each for one region. The images should be in 24 bit RGB bitmap format.

The sample uses device side enqueue feature to do image segmentation. This feature is used to repeatedly enqueue dynamically generated work on device is follows.

1. In the beginning, all seeded pixels are classified to their respective regions and put in a queue by an initialization kernel.
2. A top level kernel named `grow_region` then evaluates amount of work to be done based on the queue size and enqueues another kernel `queue_neigh` having work items equal to size of the queue.
3. `queue_neigh` finds unclassified pixels in the neighborhood of each queued pixel and forms another queue. It then enqueues a kernel `classify` to classify pixels in the newly formed queue.
4. `classify` classifies pixels in the new queue to regions, and again enqueues `grow_region`. `grow_region` dynamically evaluates the work to be done based on the size of the new queue and enqueues `queue_neigh`.

The iteration continues till no pixels remains to be queued.

The output of device side enqueue is written in "ocl_output.bmp". a host side verification code writes output in "ver_output.bmp".

4 References

1. Adams R. and Bischof L., “Seeded Region Growing”, IEEE Trans on PAMI, V.16, N0.6, pp 641-647.
2. The OpenCL Specification (ver 2.0, rev 22) document.
3. The OpenCL C Programming Language (ver 2.0, rev 22) document.

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000

For AMD Accelerated Parallel Processing:
URL: developer.amd.com/appsdk
Developing: developer.amd.com/
Support: developer.amd.com/appsdksupport