AMD **Accelerated** Parallel Processing TECHNOLOGY

SAMPLE

HDRToneMapping

1 Overview

1.1 Location

`$<AMDAPPSDKSamplesInstallPath>\samples\opencl\cpp_cl\1.x`

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.

Type the following command(s).

1. `HDRToneMapping`
   This compresses the tonal range of a digital graphic from HDR to LDR.
2. `HDRToneMapping -h`
   This 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 meaning.
	`--device`	Devices on which the program 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.
	`--dump`	Dump binary image for all devices.
	`--load`	Load binary image and execute on device.
	`--flags`	Specify compiler flags to build kernel.
`-p`	`--platformId`	Select platformId to be used (0 to N-1, where N is the number of available platforms).
`-d`	`--deviceId`	Select deviceId to be used (0 to N-1, where N is the number of available devices).
`-i`	`--iterations`	Number of iterations for kernel execution.
`-c`	`--cpattanaik`	C value. (A parameter in the Pattaneik operator.)

Short Form	Long Form	Description
-l	--delta	Delta value. (A parameter in the Pattaneik operator.)
-g	--gamma	Gamma value. (A parameter in the Pattaneik operator.)
-v	--version	AMD APP SDK version string.

2 Introduction

2.1 High Dynamic Range (HDR) Rendering

In 3D computer graphics, high dynamic range rendering (HDRR or HDR rendering), also known as high dynamic range lighting, is the rendering of computer graphics scenes by using lighting calculations done in a larger dynamic range. This allows preservation of details that might be lost due to limiting contrast ratios. [1]

Normally, digital images are created for display on monitors. These monitors support 16.7 million colors (24 bits per pixel); thus, it is logical to store numeric images to match the color range of the display. For example, file formats such as *bmp* or *jpeg* traditionally use 16, 24 or 32 bits for each pixel. Each pixel is composed of three primary colors: red, green, and blue (and, eventually, alpha). If a pixel is stored as 24 bits, each component value can range from 0 to 255. This is sufficient in most cases, but this image can only represent a 256:1 contrast ratio, whereas a natural scene exposed in sunlight can expose a contrast of 50,000:1. Most computer monitors have a specified contrast ratio between 500:1 and 1000:1. [3]

HDR uses a wider dynamic range than all other image formats. This means that every pixel can represent a larger contrast and a larger dynamic range. The usual range is called Low Dynamic Range (LDR).

HDR is typically employed in two applications:

- HDR imaging is used by photographers and movie makers for static images, with full control and unlimited processing time.
- HDR rendering is for real-time applications such as video games or simulations.

2.2 HDR Tone Mapping

Real-world scenes often have a very high range of luminance values. While digital imaging technology now enables capture of the full dynamic range of real-world scenes, we still are limited by the LDR displays; thus, the scene can be visualized on a display monitor only after the captured HDR is compressed to the available range of the display device.

Tone mapping is a technique used in image processing and computer graphics to map one set of colors to another in order to approximate the appearance of HDR images in a medium that has a more limited dynamic range. Print-outs, CRT or LCD monitors, and projectors all have a limited dynamic range that is inadequate to reproduce the full range of light intensities present in natural scenes. Essentially, tone mapping addresses the problem of strong contrast reduction from the scene values (radiance) to the displayable range while preserving the image details and color appearance important to appreciate the original scene content. [2]

Applications use mapping to:

- Produce aesthetically pleasing images.
- Reproduce as many image details as possible, or maximize the image contrast.
- Obtain a perceptual match between a real scene and a displayed image, even though the display device is not able to reproduce the full range of luminance values.

3 Pattanaik Tone Mapping Operator

This sample implements Pattanaik tone mapping operator. [4]

The global contrast helps us to differentiate between various regions of the HDR (high dynamic range) image, which we can loosely classify as dark, dim, lighted, bright etc. Within each region objects become distinguishable due to local contrast against the background – either the object is darker than the background or it is brighter than the background.

Displayable luminance is **YD(x, y) = Y(x, y) / [Yx, y) + GC]**

where

GC is the global contrast factor computed by **GC = c * YA** .

YA is the average luminance value of the whole image.

c is a multiplying factor. This brings the high luminance values closer to 1, while the low luminance values are a fraction closer to zero. A lower value of c helps boost the luminance values at the lower end of the captured scene. For most images, choose c to be 0.15.

The above equation does not attempt to preserve local luminance variations within a region, causing many details to be lost.

A detail in the image can result from the pixel being either brighter or darker than those surrounding it. If luminance Y at a pixel is more than YL, the local luminance in its immediate neighborhood, increase the displayable value YD at that point, so that it appears brighter against its surroundings. On the other hand, if the luminance Y at a point is less than the local luminance YL in its immediate neighborhood, reduce the displayable value YD, so that it appears darker than its neighborhood. To achieve this, we modify the s-function as follows:

$$YD(x,y) = Y(x, y) / [Y(x, y) + CL(x, y)]$$

where CL, the contrast luminance at a point (x, y), is obtained b modifying the global contrast factor GC with a term involving the logarithm of the ratio of the local low-pass filtered value YL to the original luminance values:

$$CL = YL [\log (\delta + [YL / Y])] + GC$$

where δ is a very small value and is included to avoid singularity while doing the log operation.

The luminance values can be obtained by RGB values using the following formula.

$$Y = 0.299 * R + 0.587 * G + 0.114 + B$$

The new R, G, B values were computed using:

$$RD = (R/Y)^{\gamma} YD$$
$$GD = (G/Y)^{\gamma} YD$$
$$BD = (B/Y)^{\gamma} YD$$

Where γ controls the display color on the monitor. For the image shown in Figure 1, γ = 0.4.

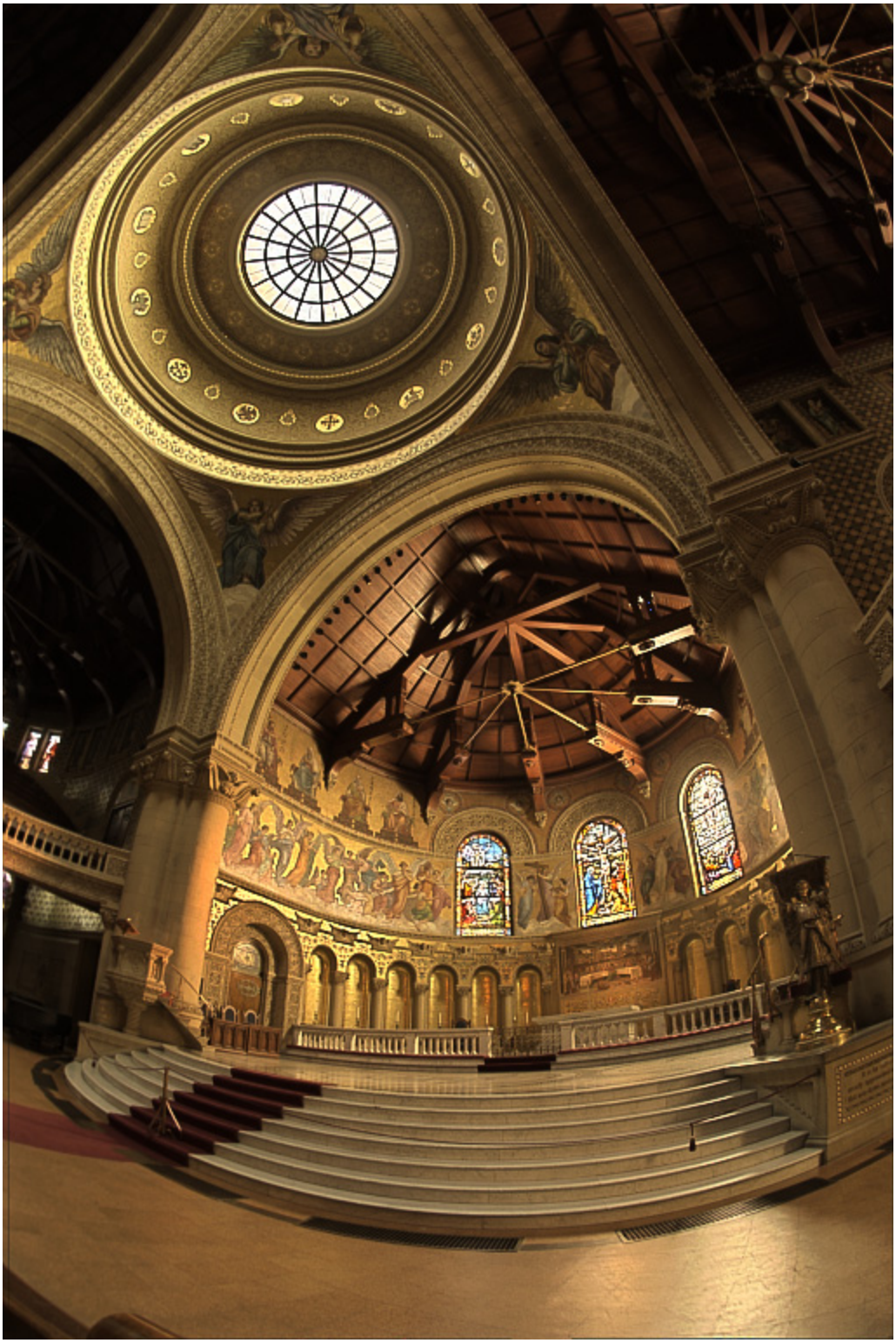

Figure 1 Sample Image

4 Implementation Details

The input to the program is a binary file that has the information of the OpenEXR image format. [5] It is possible to use OpenEXR files directly in this sample. Using a binary file reduces the complexity of image loading in the program.

Each work item calculates the luminance value by: (0.2126f * r) + (0.7152f * g) + (0.0722f * b). The average luminance in the kernel is not calculated because the input image is very small. We calculate local luminance by considering a 3X3 pixel neighborhood.

4.1 Optimizations for CPUs

1. Use the `CL_MEM_USE_HOST_PTR` flag for memory objects on CPU device. If the object is created using this flag, the CPU device is running the kernel on the buffer provided by the application. This means zero copy between the CPU device and the application buffer; the kernel updates the application buffer. In this case, a map/unmap is actually a no-op.
2. Use a work-group size of 1024 work-items.

4.2 Optimizations for Fusion Devices

Create input buffers with `CL_MEM_ALLOC_HOST_PTR` | `CL_MEM_READ_ONLY`. On Fusion systems, this type of zero copy buffer can be written to by the CPU at very high data rates, then handed over to the GPU at minimal cost for equally high GPU read-data rates over the Radeon memory bus. This path provides the highest data transfer rate for the CPU-to-GPU path. It may be necessary to use multiple CPU cores to achieve peak write performance.

5 References

1. High dynamic range rendering: http://en.wikipedia.org/wiki/High_dynamic_range_rendering
2. Tone mapping: http://en.wikipedia.org/wiki/Tone_mapping
3. High dynamic range rendering in OpenGL: transporter-game.googlecode.com/files/HDRRenderingInOpenGL.pdf
4. K.K. Biswas and Sumanta Pattanaik; School of Computer Science, University of Central Florida; Orlando, Florida, “A simple spatial tone mapping operator for high dynamic range images” in: *Proceedings of IS&T/SID's 13th Color Imaging Conference*, CIC 2005 (Nov 2005).
5. OpenEXR – A high dynamic-range (HDR) image file format developed by Industrial Light & Magic for use in computer imaging applications. http://www.openexr.com/

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