

1 Overview

1.1 Location \$(AMDAPPSDKSAMPLESROOT)\samples\bolt\examples

1.2 How to Run See the *Getting Started* guide for how to build samples. You must first compile the sample. Use the command line to change to the directory where the executable is located. The pre-compiled sample executable is at \$(AMDAPPSDKSAMPLESROOT)\samples\bolt\bin\x86_64\ for 64-bit builds. Bolt currently does not support 32-bit libraries.

Type the following command(s).

1. `RgbToYuv`
This runs the program with the default options; $s = (1024 * 768)$.
2. `RgbToYuv -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.
-q	--quiet	Quiet mode. Suppresses most text output.
-e	--verify	Verify results against reference implementation.
-t	--timing	Print timing related statistics.
-v	--version	Bolt library and runtime version string.
-x	--samples	Number of sample input values to be calculated.
-i	--iterations	Number of iterations.

2 Introduction

RGB to YUV conversion is a common preprocessing module used in video and image processing pipelines. The conversion modifies the color space of the image pixels, taking the human perception into account. This allows for transmission errors or compression artifacts to be concealed from human perception, as compared to the RGB color space.

3 RGB to YUV Conversion

RGB to YUV conversion is typically implemented as a 3x3 matrix multiplication of the input RGB data. The matrix that is applied for this sample is:

Equation 1

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.257 & 0.504 & 0.098 \\ 0.439 & -0.368 & -0.071 \\ -0.148 & -0.291 & -0.439 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix}$$

4 Implementation Details

This sample shows the usability of Bolt library's `transform()` as an alternative to `std::transform()` to perform RGB to YUV 4:4:4 color space conversion. Each iteration of the functor calculates the Y, U, and V values corresponding to the input R, G, and B values by applying the 3x3 conversion matrix, as shown above. The resultant YUV values then are clipped to a maximum of 255. This sample does not use `device_vectors` to allocate buffers on an available device. It is left as an exercise to the user to look at other Bolt samples and use `device_vectors` in this example and gain significant performance boost while running the sample using Bolt library.

The advantage of defining a Functor as a `BOLT_FUNCTOR` is that this code is usable by both `std::transform()` and `bolt::cl::transform()` without modification to user code elsewhere.

5 Recommended Input Option Settings

For the best performance, enter the following on the command line: `-x 786432 -i 100 -q -t`

6 References

1. <http://en.wikipedia.org/wiki/YUV>.

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/
Forum: developer.amd.com/opencclforum



The contents of this document are provided in connection with Advanced Micro Devices, Inc. ("AMD") products. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice. The information contained herein may be of a preliminary or advance nature and is subject to change without notice. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property rights is granted by this publication. Except as set forth in AMD's Standard Terms and Conditions of Sale, AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

AMD's products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD's product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

Copyright and Trademarks

© 2012 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, ATI, the ATI logo, Radeon, FireStream, and combinations thereof are trademarks of Advanced Micro Devices, Inc. OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos. Other names are for informational purposes only and may be trademarks of their respective owners.