CUDA SAMPLES GUIDE TO NEW FEATURES

DA-05689-001_v5.0 | October 2012

Application Note

TABLE OF CONTENTS

Chapter 1. CODE SAMPLES IN CUDA 5.0

NVIDIA® CUDA™ Toolkit version 5.0 introduces some exciting new features and capabilities. To illustrate the capabilities and advantages of the new features, the CUDA Toolkit includes many new and improved code samples. In addition, existing code samples have been upgraded to take advantage of the new features. This document serves as a guide to the new code samples as they relate to the new CUDA Toolkit Version 5.0 and Version 5.0 feature list.

1.1 CUDA Version 5.0 Highlights

- Native support for Kepler GPUs (SM 3.5), with CUDA Dynamic Parallelism as a new CUDA 5.0 feature.
- Overall improvements in driver and toolkit for Kepler GPUs (SM 3.0) performance.
- All projects and Makefiles have been updated accordingly.
- New directory structure for CUDA samples. Samples are classified accordingly to categories: `0_Simple`, `1_Utilities`, `2_Graphics`, `3_Imaging`, `4_Finance`, `5_Simulations`, `6_Advanced`, and `7_CUDALibraries`

1.2 New CUDA Dynamic Parallelism Samples in CUDA 5.0

cdpSimplePrint

This sample demonstrates simple `printf` implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

cdpSimpleQuickSort

This sample demonstrates a simple quicksort implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

`cdpAdvancedQuickSort`

This sample demonstrates an advanced quicksort implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

`cdpLUDecomposition`

This sample demonstrates LU Decomposition implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

`cdpQuadTree`

This sample demonstrates Quad Trees implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

`simpleDevLibCUBLAS`

This sample implements a simple CUBLAS function calls that call GPU device API library running CUBLAS functions. CUBLAS device code functions take advantage of CUDA Dynamic Parallelism and requires compute capability of 3.5 or higher.

1.3 New CUDA Code Samples in CUDA 5.0

`simpleIPC`

This CUDA Runtime API sample is a very basic sample that demonstrates Inter Process Communication with one process per GPU for computation. Requires Compute Capability 2.0 or higher and a Linux Operating System.

`simpleSeparateCompilation`

This sample demonstrates a CUDA 5.0 feature, the ability to create a GPU device static library and use it within another CUDA kernel. This example demonstrates how to pass in a GPU device function (from the GPU device static library) as a function pointer to be called. Requires Compute Capability 2.0 or higher.

`bindlessTexture`

This example demonstrates use of `cudaSurfaceObject`, `cudaTextureObject`, and MipMap support in CUDA. Requires Compute Capability 3.0 or higher.

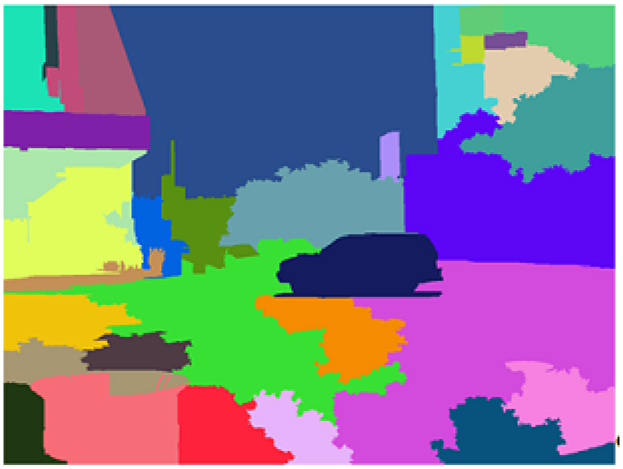
`stereoDisparity`

A CUDA program that demonstrates how to compute a stereo disparity map using SIMD SAD (Sum of Absolute Difference) intrinsics. Requires Compute Capability 2.0 or higher.

Chapter 2.
CODE SAMPLES IN CUDA 4.2

`segmentationTreeThrust` (New!)

This example demonstrates a method to build image segmentation trees using Thrust. This algorithm is based on Boruvka's MST algorithm.



Chapter 3. CODE SAMPLES IN CUDA 4.1

MersenneTwisterGP11213

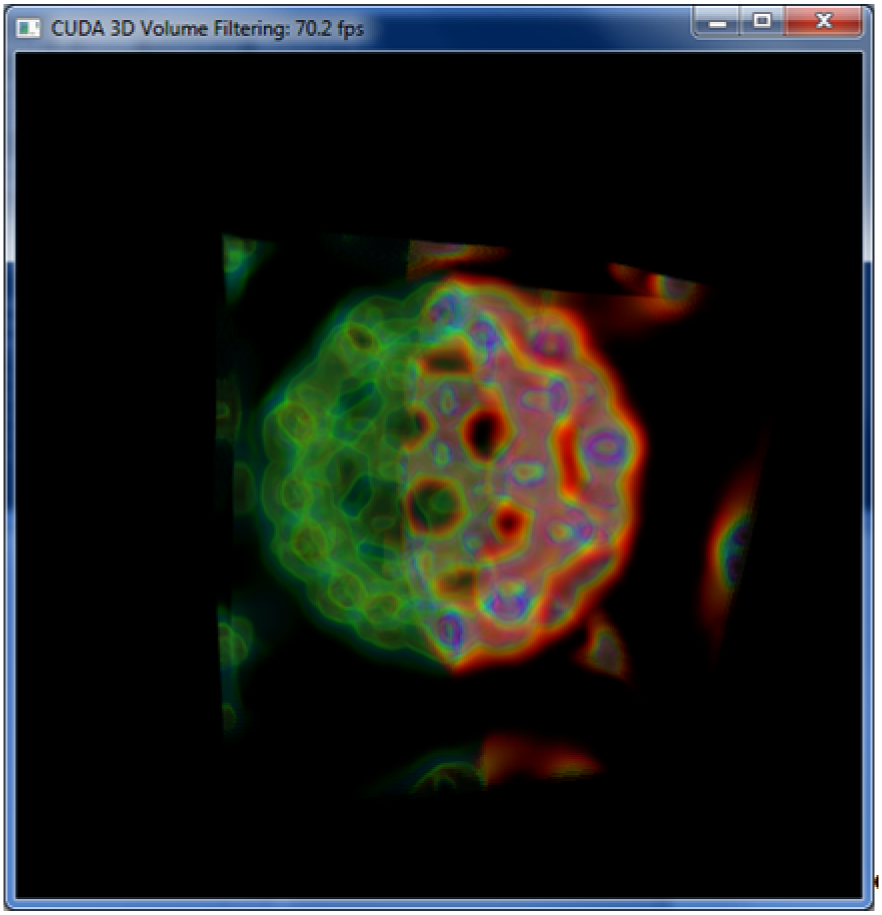
This sample implements Mersenne Twister GP11213, a pseudorandom number generator using the CURAND library.

HSOpticalFlow

When working with image sequences or video it's often useful to have information about objects movement. Optical flow describes apparent motion of objects in image sequence. This sample is a Horn-Schunck method for optical flow written using CUDA.

volumeFiltering

This sample demonstrates basic volume rendering and filtering using 3D textures.



simpleCubeMapTexture

This sample demonstrating how to use `texcubemap` fetch instruction in a CUDA C program.

simpleAssert

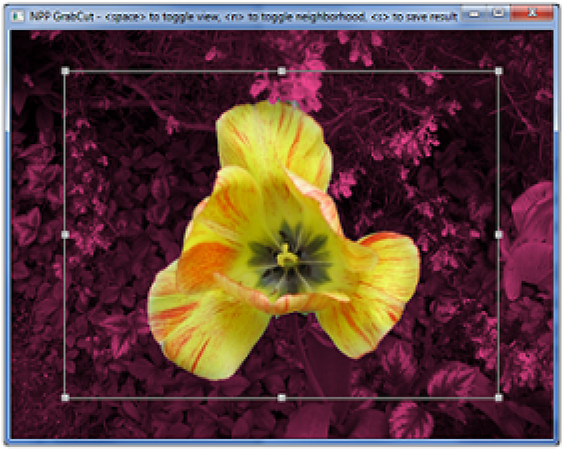
This sample demonstres how to use GPU assert in a CUDA C program.

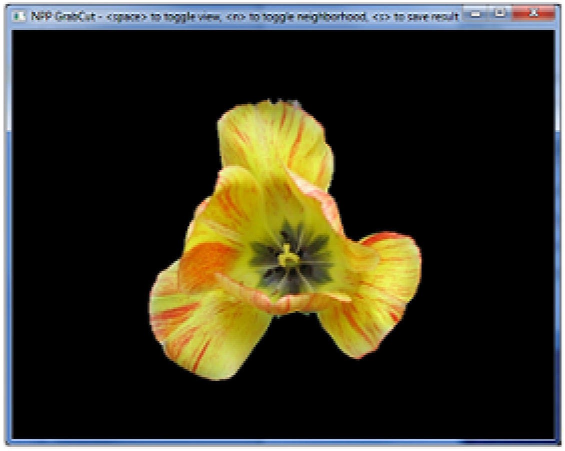
NPP

For additional information about `NPP`, please refer to the document *NPP_Library.pdf* included with the CUDA toolkit.

grabcutNPP

CUDA implementation of Rother et al. GrabCut approach using the 8 neighborhood `NPP` Graphcut primitive introduced in CUDA 4.1. (C. Rother, V. Kolmogorov, A. Blake. *GrabCut: Interactive Foreground Extraction Using Iterated Graph Cuts. ACM Transactions on Graphics (SIGGRAPH'04)*, 2004).


NPP GrabCut - <space> to toggle view, <n> to toggle neighborhood, <s> to save result


NPP GrabCut - <space> to toggle view, <n> to toggle neighborhood, <s> to save result

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

Copyright