Video Motion Estimation Tutorial

Sample User's Guide

Intel® SDK for OpenCL Applications - Samples*

Document Number: 329889-001US

Contents

Legal Information

Optimization Notice
Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

About the Motion Estimation Tutorial

The Video Motion Estimation (VME) tutorial provides step-by-step guidelines on the using Intel's motion estimation extension for OpenCL* standard. The motion estimation extension includes a set of host-callable functions for frame-based Video Motion Estimation.

The motion estimation extension depends on the OpenCL 1.2 notion of the built-in kernels and on the Intel's "accelerator" extension, which provides an abstraction for the specific hardware-accelerated capabilities.

Notice that to simplify host logic, the tutorial accepts raw YUV files in YV12 only. Refer to the http://www.fourcc.org/yuv.php#YV12 for details on the YV12 format.

Introduction

Motion estimation is the process of determining the motion vectors that describe the transformation from one 2D image to another, usually from adjacent frames in a video sequence. The motion estimation functions, considered in this article, accept full-frame single-channel (luma) images as input, perform a motion search operation, and return a motion vector field as output.

The introduced VME functionality exposes part of the hardware acceleration pipeline for video acceleration. Intel's VME extension provides low-level functionality, currently restricted to the single-channel (luma) input images and block matching methods, so motion vectors are computed for rectangular pixel blocks.

For more details on the VME and associated extensions APIs, refer to the technical article on the VME at http://software.intel.com/en-us/articles/intro-to-motion-estimation-extension-for-opencl.

Motivation

Motion vectors are key element in the video compression algorithms. Motion vectors are also useful for several applications. For example, when generating "slow motion effects", motion vectors can provide the basis to generate intermediate frames for frame rate (up)conversion. Another example is increasing the original frame rate of the digitized film (24 fps) to match the TV rate.

Motion vectors are also useful for image stabilization: the motion vectors in the entire frame can be averaged to produce a "global" motion vector, which can serve as an approximation to a real video camera motion.

Accelerator and Motion Estimation Extensions

The motion estimation extension consists of the new OpenCL built-in kernel (see OpenCL 1.2 specification section 5.6.1) which performs motion estimation, as well as motion estimation accelerator object, which represent the state of the underlying acceleration engine. The kernel is queued for execution from the host using the standard ND-range mechanism.

Both `cl_intel_accelerator` and `cl_intel_motion_estimation` extensions should be listed in the `CL_DEVICE_EXTENSIONS` string (see OpenCL 1.2 specification section Table 4.3) for the Intel HD Graphics device. If not, update the Intel® Graphics Driver.

Hardware Considerations

The goal of the sample is to demonstrate hardware-accelerated VME functionality that Intel HD Graphics device offers. Thus, to run the sample, a 3rd Generation Intel® Core™ processor (and higher) or an Intel® Atom™ processor with Intel HD Graphics (formerly codenamed Bay Trail) is required.

Implementation Details

The tutorial utilizes the following APIs:

- `clGetExtensionFunctionAddressForPlatform`
- `clCreateAcceleratorINTEL`
- `clReleaseAcceleratorINTEL`
- `clEnqueueNDRangeKernel`
- `clCreateProgramWithBuiltInKernels`

Example Results

The pictures below show two frames (reference and source) and computed motion vectors overlaid on the second frame. Specifically, the vectors are rendered as the strokes of the appropriate magnitude. So they point to the new (actually best-matched) pixel block positions.

Notice the radial pattern of the motion vectors, which is due to the nature of the transformation between frames (zoom in addition to the camera movement).

Refer to the technical article for some performance considerations related to VME at http://software.intel.com/en-us/articles/intro-to-motion-estimation-extension-for-opencl.

Controlling the Sample

The sample executable is a console application. Use the following sequence of commands to control the sample:

MotionEstimation.exe --input <string> --output <string> [-width <int>] [-height <int>] [-h]

Notice that sample-specific parameters have default values, so you can run the sample binary without specifying any command-line options.

The following command-line arguments are available:

Option	Description
`-h, --help`	Show this help text and exit.
`--input  <string>`	Input video sequence filename - YUV file (YV12 format). Default input file is `1920x1080_5frames.yuv`
`--output <string>`	Output video sequence with overlaid motion vectors filename - YUV file (YV12 format). Default name for the output file is `output.yuv`.
`--nobmp`	Disable output frames to the sequence of BMP files in addition to the YUV file. By default the output is on.
`--width <int>`	Set frame width for the input file. Default value is 1920.
`--height <int>`	Set frame height for the input file. Default 1080

By default sample output sequence of bitmap files that duplicates the content of the output YUV stream.

References

1. Motion Estimation Extension for OpenCL* specification at http://www.khronos.org/registry/cl/extensions/intel/cl_intel_motion_estimation.txt
2. Accelerator Extension for OpenCL* specification at http://www.khronos.org/registry/cl/extensions/intel/cl_intel_accelerator.txt
3. Intro to Motion Estimation Extension for OpenCL* at http://software.intel.com/en-us/articles/intro-to-motion-estimation-extension-for-opencl