Getting Started with GPU-accelerated Computer Vision using OpenCV and CUDA
Anatoly Baksheev, OpenCV GPU Module Team Leader, Itseez
July 30, 2013, 10:00 AM PDT

Q&A Session

Q: Is NVIDIA planning on an Android tablet for Tegra 5?
A: Yes.

Q: Is there any support for cuda on android?
A: We don't support CUDA on mobile yet, but Tegra 5 will have full hardware-accelerated support of CUDA 5 in the near future!

Q: During many operation memory dump error throws while using CUDA programming..but by rebooting the system the error recovers and program successfully executes..wt is d error ?
A: It's hard to say with this information. Unless Shervin has an idea, I recommend posting a detailed question to answers.opencv.com.

Q: Details below: Using OpenCV_for_GPU i.e. with CUDA got following results. Single sample, so it may not be very conclusive, but gives a feeling that GPU should be used for a large set of or iterative set of ops.
   *Time for CPU Bilateral Filter is 1.74428 milliseconds.
   *Time for CPU Canny is 0.0686374 milliseconds.
   #Time for overall CPU ops is 1.86274 milliseconds.
   Time for GPU Upload of image is 34.5247 milliseconds.
   *Time for GPU Bilateral Filter is 2.71138 milliseconds.
   *Time for GPU Canny is 0.214972 milliseconds.
   Time for GPU Download of image is 0.00596806 milliseconds.
   #Time for overall GPU ops is 37.5986 milliseconds.
   ^Combined time for both ops on CPU: 1.8129174
   ^Combined time for both ops on GPU: 2.926352

Or probably, the GeForce 210 -GPU I'm using is poorer than the Quadcore i5!

A: This could have multiple causes. I don't think we'll be able to diagnose this one here. I recommend you post this to answers.opencv.org.

Q: Is there some mesh based image warping support in OpenCV on GPU? Like can we have merger of OpenGL and OpenCV in future to include some basic functions of OpenGL in OpenCV?
A: There are no plans to support.

Q: Could you also share about how can we contribute to OpenCV development? Like you do?
A: http://opencv.org/contribute.html

Q: CMake Error: The following variables are used in this project, but they are set to NOTFOUND. Please set them or make sure they are set and tested correctly in the CMake files:
   CUDA_CUDA_LIBRARY (ADVANCED)
linked by target "opencv_gpu" in directory /home/anand/Documents/OpenCV246PreBuilt/opencv-2.4.6/modules/gpu
linked by target "opencv_world" in directory /home/anand/Documents/OpenCV246PreBuilt/opencv-2.4.6/modules/world
CUDA_nvcuvid_LIBRARY (ADVANCED)
linked by target "opencv_gpu" in directory /home/anand/Documents/OpenCV246PreBuilt/opencv-2.4.6/modules/gpu
linked by target "opencv_world" in directory /home/anand/Documents/OpenCV246PreBuilt/opencv-2.4.6/modules/world

A: CMake can’t detect all libs from CUDA SDK for some reasons (your PC specific). We suggest setting path to the library manually in CMake. Or reinstall CUDA SDK to default place. Or try adding –DWITH_NVCUVID=NO to your CMake flags when building OpenCV, to remove the dependency on CUDA video codec such as when building for CARMA devkit.

Q: Can any build of OpenCV built with one particular Nvidia GPU also work on another version of Nvidia GPU? Or is it built with specific to that GPU? Like can the GPU dll of OpenCV built with GEForce 210 work on GEForce 435M?
A: It depends on the GPU flags you compile OpenCV with. Read details of CUDA compilation (CUDA_Compiler_Driver_NVCC.pdf) to see which entries are most suitable for you, such as PTX/ISA or CUBIN versions.

Q: Details below: Using OpenCV_for_GPU i.e. with CUDA got following results. Single sample, so it may not be very conclusive, but gives a feeling that GPU should be used for a large set of or iterative set of ops.
*Time for CPU Bilateral Filter is 1.74428 milliseconds.
*Time for CPU Canny is 0.0686374 milliseconds.
#Time for overall CPU ops is 1.86274 milliseconds.
Time for GPU Upload of image is 34.5247 milliseconds.
*Time for GPU Bilateral Filter is 2.71138 milliseconds.
*Time for GPU Canny is 0.214972 milliseconds.
Time for GPU Download of image is 0.00596806 milliseconds.
#Time for overall GPU ops is 37.5986 milliseconds.
^Combined time for both ops on CPU: 1.8129174
^Combined time for both ops on GPU: 2.926352
Or probably, the GEForce 210 GPU I'm using is poorer than the Quadcore i5! ;) :P

A: Yes OpenCV’s GPU module is better suited to large images and heavy operations such as Bilateral Filter or people detection or any stereo vision algorithms. For simple image filters on small images, a multi-core CPU might be faster than a GPU.

Q: I got a pretty worse time performance running the simple edge detection program from last webinar using GEForce 210 than using than on my i5 Quad core CPU.
A: Old GPU. Agree no speed-up at all. Especially for such difficult for GPU algorithm as Canny.

Q: is there any support for cuda on android?
A: A beta version of CUDA 5.5 is available that supports ARM-based platforms. An updated beta that has been tested on Andriod will be available shortly.
Q: As I want to implement OpenCV algs on GPUs using OpenCL on Android
A: Some Android devices from other vendors are being released with OpenCL API support, but it is unclear whether any of them actually run OpenCL any faster than CPU code, since it is still very early for OpenCL on ARM.

Q: Means support to opencl available?
A: CUDA, not OpenCL

Q: plz tell me the devices that suppots opencl
A: I am not aware of any mobile platform that supports OpenCL.

Q: how to implement opencl on android?
A: We do not currently have OpenCL support on Android.

Q: Does cv_8UC3 really store only 3 bytes per pixel or it uses additional byte for processing performance purpose?
A: —3 bytes. There is tradeoff between speed and convenience. We selected convenience to minimize number of format conversions. Users can try to use CV_8UC4 for performance.

Q: I see that OpenCV also has a lot of GPU support through OpenCL. Are OpenCL and CUDA enhancements to OpenCV being coordinated, or are they done independently?
A: They were independent, but now OpenCL is starting to becoming officially supported by OpenCV, so it will be more coordinated.

Q: Is there any plan/support of cuda for FPGA like opencl..
A: No plans as of yet.

Q: Time for gpu upload for an image was high if i measure the time across the api for a single run.i have noticed if running twice (.upload) and measure the time across the upload, time will be comparable to that of opencl api.
A: When measuring performance of GPU code, always do a GPU test run first so that it initializes the GPU and GPU executable, since it will always be very slow. After the first run, it should be full speed from then onwards.

Q: I have noticed upload API in cvgpu for uploading of images to be processed in GPU. What does the upload wrapper contain exactly (like opencl using enqueread/write buffer) for h2d and d2h(download) operations
A: Yes it basically does an enqueue read or a cudaMemcpy. Since OpenCV is open-source, you are free to look at the exact code in the folder "opencv\modules\gpu\src".

Q: Can you brief about how the task parallelism is achieved here in cuda?
A: Task parallelism here means overlapping CPU and GPU operations using the asynchronous calls Anatoly described earlier.
Q: Answering to shalini question, run the exe twice and measure the time across upload function you will definitely get upload time lesser. i have observed the similar reading when i did it first
A: No you shouldn't run the EXE twice, you should do a call to the GPU module twice in your code (or an iteration of your loop twice), because the very first call to OpenCV GPU module will take long. If you run your ""EXE"" again, it will still take long again on the first call to GPU, because the startup delay is for each program (it basically has to compile your CUDA code and then run it on the GPU).

Q: Is GPU supported with Python bindings?
A: No.

Q: From a library point of view, do you have scheduled increasing functionality into the GPU, like Jacket ?
A: Do you mean does OpenCV have plans of integration with Jacket or other libraries? Not that I know of. It will make use of some highly-optimized CUDA libraries like CUB (http://nvlabs.github.io/cub/) that is currently being developed.

Q: I know how to use OpenCV GPU functions. The CUDA GPU Computing SDK is very specific to OpenGL (which I don’t use). Is there a better way of tapping into the functions within the GPU Computing library?
A: I'm not sure I understand the question. You don't need to use OpenGL to use the CUDA Computing SDK--you would program in CUDA and C++ instead.

Q: Can I use CUDA from Matlab?
A: Yes there are ways

Q: I would like to know if there are any dual CUDA-enabled GPU laptops that can run OpenCV, processing USB HD webcam streams?
A: There are laptop GPUs that support CUDA and OpenCV. OpenCV supports a number of camera input sources including USB webcams. Not sure what you mean by ""dual"" (CPU?).

Q: I would like to know if there are any dual CUDA-enabled GPU laptops that can run OpenCV, processing USB HD webcam streams?
A: I'm not sure, but I don't see why not. Any laptop with ""Optimus"" technology has a CUDA GPU, but whether it can process HD video always depends on how much processing you want to do!

Q: ""dual"" as in a laptop that has two GPUs, e.g. one of several that can host two CUDA-enabled 780M
A: No, the dual-GPU laptops have an integrated Intel GPU and a discrete NVIDIA GPU, so that you can use Intel GPU for normal low power use and the NVIDIA GPU will only activate when you need it for things like CUDA or 3D games

Q: Thanks. Indeed, dual NVIDIA 780M in a single laptop (Alienware m18x, Falcon, Sager), which is very tempting, but it's not clear to me how OpenCV will handle them, so I asked. I know they can be in SLI configuration, for example, but my understanding is that OpenCV will still treat them as distinct GPU units. I'm strongly interested in knowing if anyone has done it.
A: It's true that OpenCV doesn't yet have automatic handling of multiple GPUs, if the device appears as 2 CUDA GPUs and you want to run something on both, you need to manually split your data or tasks across both GPUs yourself, typically by using 2 CPU threads to handle 2 GPUs. Eventually it is expected that OpenCV will have easy multi-GPU support like it has multi-CPU support.
Q: Or, for example, the new EUROCOM with dual 780M (total > 3000 CUDA cores) and Haswell board/chipset. Again, very interested in making it work, but only reservation is I've yet to see someone do so.
A: There is typically a certain amount of trial and error when trailblazing, but I don't think we know of any reason why it shouldn't work.

Q: Okay, thank you very much for the clarification re: multi-GPU handling, and "trailblazing". The note about dedicating one CPU to each GPU may be particularly helpful, too.
A: You basically have to develop your multi-GPU code the same way you traditionally had to develop multi-threaded CPU code before there was OpenMP & TBB & CUDA: you had to manually split up your work into threads and deal with synchronization issues yourself. You currently have to do that to use multi GPU in OpenCV.

Q: hmm ... there are a few laptops (e.g. alienware m18x, falcon) that have two distinct NVIDIA GPUs (e.g. 780M), in SLI configuration or otherwise. i.e. /not/ an integrated Intel GPU + one NVIDIA GPU. are you only referring to the systems with one NVIDIA GPU?
A: Oh sorry I wasn’t aware of that, I have only used laptops with one NVIDIA GPU with an Intel GPU. Apparently you can get dual-NVIDIA GPU laptops!

Q: Can i use cuda mixed with arm platform
A: Yes you can currently buy a CARMA or Kayla development board. Eventually you will be able to run CUDA directly on Tegra 5!

Q: what are some of the future plans for OpenCV-CUDA?
A: All complex algorithms will be converted to abstract interfaces with implementors; extended device layer (expression templates, transform iterators, probably lambdas); memory pools for temporary buffers

Q: any good instruction for installing the cuda compatible opencv using cmake from command line?
A: Cmake command line can be used, but there are differences between platforms and it's a bit more complicated. We recommend the GUI.

Q: Does PtrStepSz support 3D array?
A: Indirectly, yes you could do it, but it is expected to be used for 2D images.

Q: Are any point-cloud data abstraction libraries supported or supplemented by CUDA?
A: PCL is a sister-project with OpenCV, it is commonly accelerated with CUDA

Q: Is there a specific way to flush the GPU cache between subsequent calls or does loading the image again suffice?
A: Each call to a OpenCV GPU function actually goes from CPU, and therefore you can't expect to re-use GPU cache between OpenCV calls, or even CUDA "'shared memory'".

Q: As I know - different versions of nvidia architecture provide different compute capabilities so algorithms can get benefits from apropriate hardware architecture. Does OpenCV's gpu module contain solution which take into account benefits of appropriate gpu?
A: There are some optimizations for specific architectures, but most of the code is designed to work with older GPUs as well as new GPUs. Anatoly can give more details.
Q: Can the GPU handle an array of GpuMat? Moreover, if the data is become large-scaled, say MRI data (4D), how large it can handle? How about the MultiGpu approach?
A: Yes some of OpenCV functions take an array of GpuMat, but only when it makes sense. For the max size of GpuMat, it depends on your GPU hardware / available GDDR memory.

Q: Does the OpenCV work with MultiGPU Cluster (either Shared or Distributed memory system)
A: --No

Q: could you repeat the link to GeonInt accelartor?
A: Try ""http://www.nvidia.com/object/tesla-geoInt-accelerator.html"

Q: make no sense for GPU access to global mem to go thru cpu. They are physically different and on different addres lines
A: yes, thats why it doesn't normally happen (unless if you have an integrated GPU such as an Optimus dual-GPU laptop, or for mobile devices). The probably is that a normal ""cudaMemcpy"" is typically a CPU->CPU copy in the driver followed by a CPU->GPU upload, but page-locked memory allows skipping the CPU->GPU memcpy.

Q: So you trade GPU DMA to DDR3 vs direct access to DDR5 RAM
A: No page-locked memory does not mean each CUDA access to ""global memory"" actually goes through the CPU, page-locked mem just means that it will be more efficient to transfer the data to the GDDR from the CPU, since the driver doesn't have to make its own copy of the buffer before sending it to the GPU's GDDR.

Q: what is the basis for saying page lock mem is 2x? Does match the physics of a system
A: Normally when you do a ""cudaMemcpy"", it does a mempy of the data into the GPU driver's memory area first, before sending it to the GPU. If you use page locked memory, it means the data can go straight to the GPU, so basically just a CPU->GPU copy instead of a CPU->CPU->GPU copy.

Q: does 2D DWT operation can be implement with OpenCV/GPU?
A: No OpenCV GPU module does not come with Discrete Wavelet Transform, but you can use other CUDA libraries for this.

Q: Can we tile/chain OpenCV calls to keep intermediate images in cache?
A: You can create a sequence of calls to keep the images in the GPU's global memory between calls, but not in cache. Additional chaining functionality is planned for OpenCV3.0.

Q: Does the GPU model support OpenCV data structures like CvSeq
A: No, it is only really for images. Some of the GPU functions output other data structures but it is stored in CPU host memory instead of on GPU.