Direct capture to GPUs

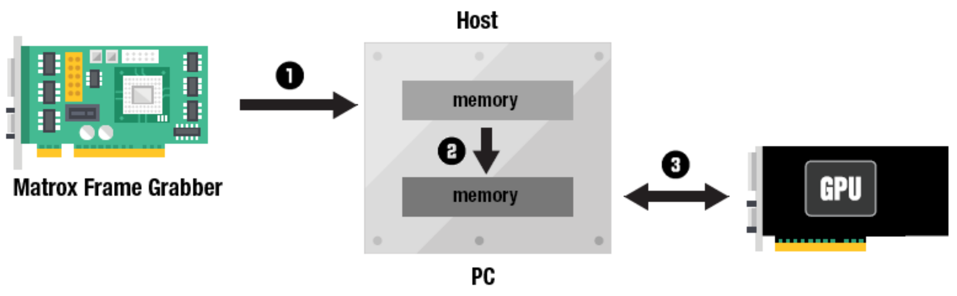
GPUs for vision

Graphics Processor Units (GPUs) are used increasingly in high-performance vision systems because of the tremendous computing power they provide through their massive parallel processing architectures. They're able to greatly accelerate a significant number of image processing operations.

However, the performance gains they deliver (in terms of execution speed) are hampered by the traditional way of getting image data onto their memory, which relies on the host CPU and memory acting as intermediaries between the image acquisition device and the GPU itself.

With camera resolutions and image acquisition rates rapidly increasing, users want a way to directly capture images to GPU memory for processing and/or display to obtain a better performing overall system.

Traditional data flow



❶ A frame grabber's Direct Memory Access (DMA) engine transfers the captured image data from the board to a host memory buffer.

❷ A host memory to host memory copy is then required to create a GPU-compatible buffer. That is a CUDA/DirectX/OpenGL/other (depending on the GPU) surface on pinned host memory.

❸ The GPU-compatible buffer then needs to be transferred onto the GPU. This transfer is either initiated by the GPU (fetching from host memory) or by the CPU (copying to GPU memory). Once the processing is completed by the GPU, the resulting pixel data can be used for display or transferred back to the host, by the GPU. A host memory to host memory copy is often required again to bring the GPU-compatible buffer to a host memory buffer formatted for further CPU-based processing.

Host CPU intervention, and thus utilization, is often required in many of the steps once the initial image data is transferred to the host memory buffer by the frame grabber. The multiple memory copy operations (i.e., Host to Host, Host to GPU, GPU back to Host and often Host to Host again) required to complete the process adds latency and can drop throughput.

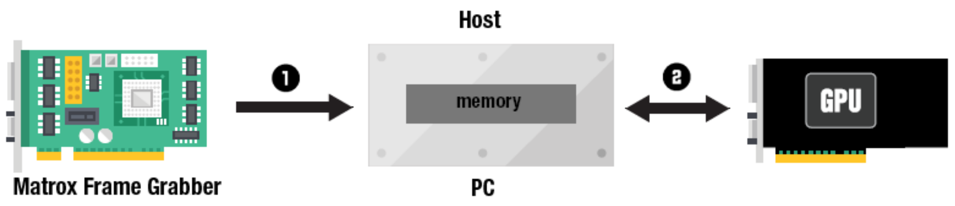
Direct capture to GPUs

Seeking to improve the effectiveness of GPU processing, and subsequently, the image transfers between the GPU and Host, GPU manufacturers have come up with dedicated SDKs and functions to reduce Host involvement for getting data from an acquisition device directly to their GPUs (i.e.,NVIDIA's GPUDirect for Video or CUDA and AMD's DirectGMA).

Working with Matrox Imaging Library

Matrox Imaging Library (MIL) can be used in conjunction with these specialized SDKs to achieve highly efficient data transfers with very low host CPU usage, between the frame grabber and the GPU or from the GPU to the Host. The method for doing so with a NVIDIA and AMD GPU are described below. The program source code for each method is available upon request.

Nvidia GPUs



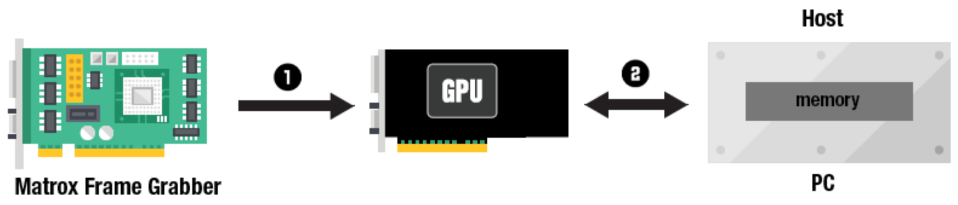
The method for a NVIDIA GPU is similar to GPUDirect™ for Video (https://developer.nvidia.com/gpudirectforvideo). It is based on a standard MIL example (MdigProcess)reworked to use standard CUDA 7.0 functions. The approach works with all NVIDIA GPUs supported by CUDA and not just those supported by GPUDirect for Video.

❶ The image acquisition and DMA transfer to Host are handled by the frame grabber using the MdigProcess( ) function.

❷ Using CUDA, the GPU then copies the image buffer from host memory to the GPU without CPU involvement (i.e., 0% CPU usage) at up to 12GB/s over a PCIe 3.0 x16link. Once the GPU-based processing is completed (ex. Sobel filter), the image is copied from the GPU back to host memory using the GPU, again without CPU involvement.

Direct capture to GPUs

AMD GPUs



The method for an AMD GPU uses DirectGMA (http://developer.amd.com/tools-and-sdks/graphics-development/firepro-sdk/firepro-directgma-sdk/) through OpenGL. Note that DirectGMA is officially supported only with the FirePro™ W5x00 and above and all FirePro™ S series GPUs.

The image data path is slightly different than the CUDA example previously shown as an image acquisition buffer is allocated directly in GPU memory, which is in turn accessible to MIL using OpenGL. The location of this buffer is then mapped to MIL.

❶ The frame grabber performs a DMA transfer to the GPU memory.

❷ The GPU can then display, process or copy back the result to a host memory buffer without CPU intervention (i.e. 0% CPU usage).