Hardware and Software Platforms for Computer Vision

CS: 534 Computer Vision

Chetan Tonde

Outline

- Hardware Platforms
  - CPU
  - GPUs
  - Heterogeneous platform (CPU+GPU+others)
  - High Performance Computing (HPC- clusters)
  - FPGA's , Embedded systems (ARM's) etc.
- Software Platforms
  - CPU
    - MATLAB, OpenCV
    - Cimg++,gil (generic image library - boost),ITK Toolkit etc.
    - Java - JImage.
  - GPU
    - CUDA
    - OpenGL ,Cg scripting, HLSL (High Level Shader Language)
  - CPU+GPU's
    - OpenCL
  - Others platforms
    - JImage, OpenCV

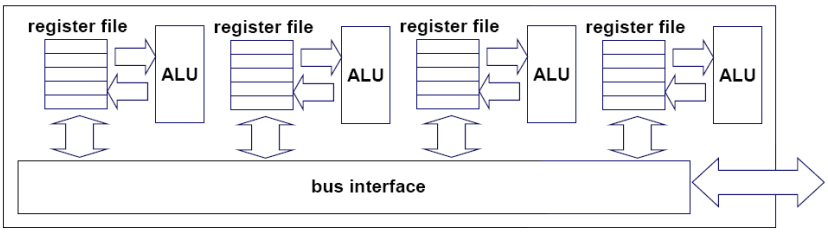
Hardware

- Requirements
  - Fast processing required in many image processing applications
  - Image processing algorithms are computationally expensive
  - Data needs to processed efficiently in serialized or parallel threads
  - Low cost and portable
  - Optimized memory architecture faster memory access

CPU - Architecture

- Intel® Core™2 Quad Processor Q9650
  - CPU cores – 4
  - Number of parallel Threads – 4
  - Clock Speed - 3 GHz
  - L2 Cache - 12 MB
  - FSB Speed - 1333 MHz
  - RAM DDR2 – 8 GB
  - freq - 1066Hz

CPU - Architecture



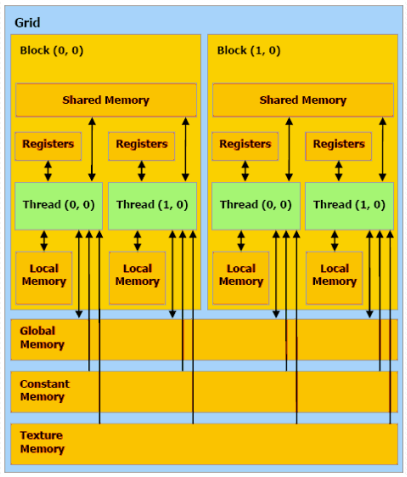
GPU Architecture -Specs

- Tesla C1060
  - 240 GPU cores
  - 1.296 GHz
  - Internal RAM: 4 GB
  - Single precision floating point performance: 933 GFLOPs (3 single precision flops per clock per core)
  - Double precision floating point performance: 78 GFLOPs (0.25 double precision flops per clock per core)
  - Internal RAM speed: 102 GB/sec (compared 21-25 GB/sec for regular RAM)
  - Has to be plugged into a PCIe slot (at most 8 GB/sec)

Ref: Parallel Programming & Cluster Computing GPGPU: Number Crunching in Your Graphics Card – SC 2009 - Portland

GPU Architecture

- Host memory (CPU)
- Device memory (GPU)
  - *Global*: visible to all threads in all blocks – largest, slowest
  - *Shared*: visible to all threads in a particular block – medium size, medium speed (share pixels)
  - *Local*: visible only to a particular thread – smallest, fastest
- Texture Hardware
  - Cached
  - Bilinear interpolation
  - Data-type conversion
  - Boundary clamping
  - Multi-channel aware 1, 2, or 3D
- Visualization
  - Graphics interpolation without data transfer overhead



Ref: Parallel Programming & Cluster Computing GPGPU: Number Crunching in Your Graphics Card – SC 2009 - Portland

Example Multi-GPU's

- Multi-GPU support (nVidia-SLI, ATI-Crossfire)
- Workload sharing by SFR/AFR/SLI anti-aliasing
- Physics calculation (eg. nVidia-PhysX)
- Hybrid SLI – (Onboard + GPU's)
- Easily extend to multiple GPU's via Context

Notes:
SLI – (Scan Line Interleave)/Scalable Link Interface .
SLI is a Master-slave configuration – requires 2 PCI-e slots.
SFR – Split frame rendering – Splits rendered image 50/50 by using geometry of scene (sky+bottom half).
AFR- Alternate Frame Rendering – GPU's share even-odd frames.
SLI- Anti-Aliasing – Use of both GPU's for higher image quality.

Source: www.nvidia.com

Others

- HPC clusters
  - fast, large memory, very generic but costly
- DSP,FPGA, …
  - application specific
- Embedded Systems
  - Smart camera's
  - small form factor Mini-ATX and micro-ATX boards
  - microcontrollers boards– (ARM's)
  - ASIC's -(Application Specific IC's)

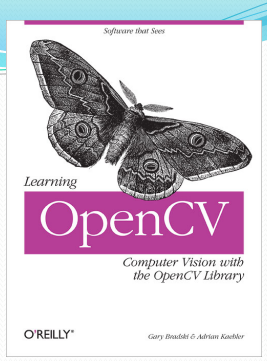
Overall Comparison

	CPU	GPU	CPU+GPU	Others
Performance	Good	Better	Best	Slowest
Memory diversity	Good	Better	Best	Restricted
Ease of programming	Easy	Hard	Very Hard	Easy
Scalability	Good	Restricted	Best	Not Scalable
Cost	Moderate cost	Moderate Cost	Costly	Very Cheap
Parallelization	Good	Best	Best	Worst
Form factor	Large	Small	Large	Smallest

Software Platforms

- Performance
- Efficient
- Ease of use
- Platform independence
- Multi-core support
- Memory Management
- Language Support
- Hardware Interface support

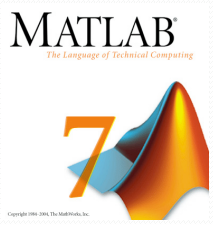
OpenCV



- C/C++ library – Open source BSD license
- Most widely used
- Provides more than 500 vision functions
  - Low level vision
  - highgui – GUI library
  - Machine learning library
- Support for Intel IPP[1] enhancements
- Support for USB\firewire camera's
- Cross platform (Windows\Linux\Mac\embedded platforms)
- Sequential architecture (No data\task parallelism)
- C#, python wrappers available
- Wide Support available- Yahoo! OpenCV Group

[1] IPP – Integrated Performance Primitives

MATLAB



- Used mainly for research purposes
- Ease of use
- Matlab Image processing toolbox
- Simulink Image and Video processing blockset
- Image Acquisition Toolbox support for camera's
- Many low level vision functions available
- Other toolboxes for data-mining, Statistics, Signal Processing etc. also available
- Wrappers for C++ and Java through mex functionality
- Parallel Computing toolbox
- Supports Embedded Systems programming
- Available for Windows\Linux\Mac

CUDA, OpenCL, Cg, HLSL, …

- Recently explored approach for vision applications
- C\C++ extensions
- Primary purpose for 3D graphics rendering for games
- Can be appropriated for vision applications
- Requires explicit parallelism done by the developer
- Requires explicit memory management
- "OpenVIDIA "library for Parallel GPU Computer Vision.
- nVidia "CUDA" extensions for C\C++ for nVidia GPU's
- "ATI Stream" Computing for ATI GPU's

CUDA - (Compute Unified Device Architecture)

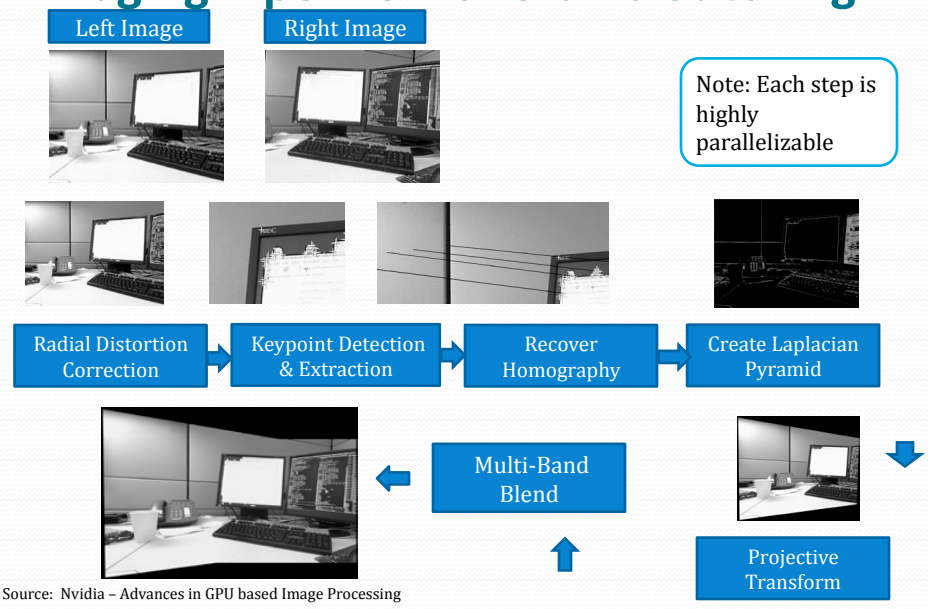
- C/C++ library – (Proprietary, Freeware)
- Currently most wide support
- Support for Intel NVPP[2] enhancements
  - Data exchange and initialization –(Set, Convert, CopyConstBorder, Copy, Transpose, SwapChannels)
  - Arithmetic and logical operations - (Add, Sub, Mul, Div, AbsDiff)
  - Threshold and compare
  - Color Conversion – (RGB To YCbCr (& vice versa), ColorTwist, LUT_Linear)
  - Filter functions – (FilterBox, Row, Column, Max, Min, Median, Dilate, Erode, SumWindowColumn/Row)
  - Geometry transformations – (Mirror, WarpAffine, WarpPerspective, Resize)
  - Statistics – (Mean, StdDev, NormDiff, MinMax, Histogram, SqrIntegral, RectStdDev)
  - Computer Vision - (ApplyHaarClassifier , Canny)
- Many vision libraries present for low level vision and Machine learning library
- Available for Windows, Unix and Mac.
- OpenVidia - Parallel GPU Computer Vision
  - http://openvidia.sourceforge.net/index.php/OpenVIDIA

[2] NVPP - nVidia Image Processing Primitives Source: Nvidia – Advances in GPU based Image Processing

Research & Libraries

- Level-Set segmentation with CUDA
  - http://code.google.com/p/cudaseg/
- Video segmentation with CUDA
  - http://code.google.com/p/gpuwire/
- Multiclass SVM implementation in CUDA
  - http://code.google.com/p/multisvm/
- CUDA implementation of pedestrian detection algorithm –MIT Project
  - http://code.google.com/p/cudapeddet/
- GPU4Vision
  - http://gpu4vision.icg.tugraz.at/
- MinGPU: A minimum GPU library for Computer Vision
  - http://server.cs.ucf.edu/~vision/MinGPU/
- CUDA SIFT
  - http://www.csc.kth.se/~celle/
  - http://www.cs.unc.edu/~ccwu/siftgpu/
- Imaging Papers/Resources at http://www.nvidia.com/cuda
- and many more….

Imaging Pipeline: Panorama Stitching



Source: Nvidia – Advances in GPU based Image Processing

GrabCuts: CUDA Graph Cuts

- Comparison between Boykov (CPU), CudaCuts and GrabCuts implementation (by Timo Stich, NVIDIA)
  - Intel Core2 Duo E6850 @ 3.00 GHz
  - NVIDIA Tesla C1060

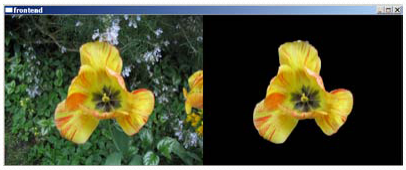


Image	Boykov (CPU)	CudaCuts (GPU)	GrabCuts	Speedup (GrabCuts vs CPU)
Flower (600x450)	**191 ms**	**92 ms**	**20 ms**	**9.5x**
Sponge (640x480)	**268 ms**	**59 ms**	**14 ms**	**19x**
Person (600x450)	**210 ms**	**78 ms**	**35 ms**	**6x**

Source: Nvidia – Adavances in GPU based Image Processing

Programming for CUDA $y_n = ax_n + y_n$

```
void saxpy_serial(int n, float a, float *x, float *y)
{
        for(int i = 0; i<n; ++i)
        y[i] = a*x[i] + y[i];
}
// Invoke serialSAXPY kernel
saxpy_serial(n, 2.0, x, y);
```

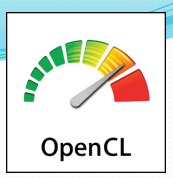
C

```
__global__ void saxpy_parallel(int n, float a, float *x, float *y)
{
        int i = blockIdx.x*blockDim.x + threadIdx.x;
        if(i<n) y[i] = a*x[i] + y[i];
}
// Invoke parallelSAXPY kernelwith 256 threads/block
int nblocks = (n + 255) / 256;
saxpy_parallel<<<nblocks, 256>>>(n, 2.0, x, y);
```

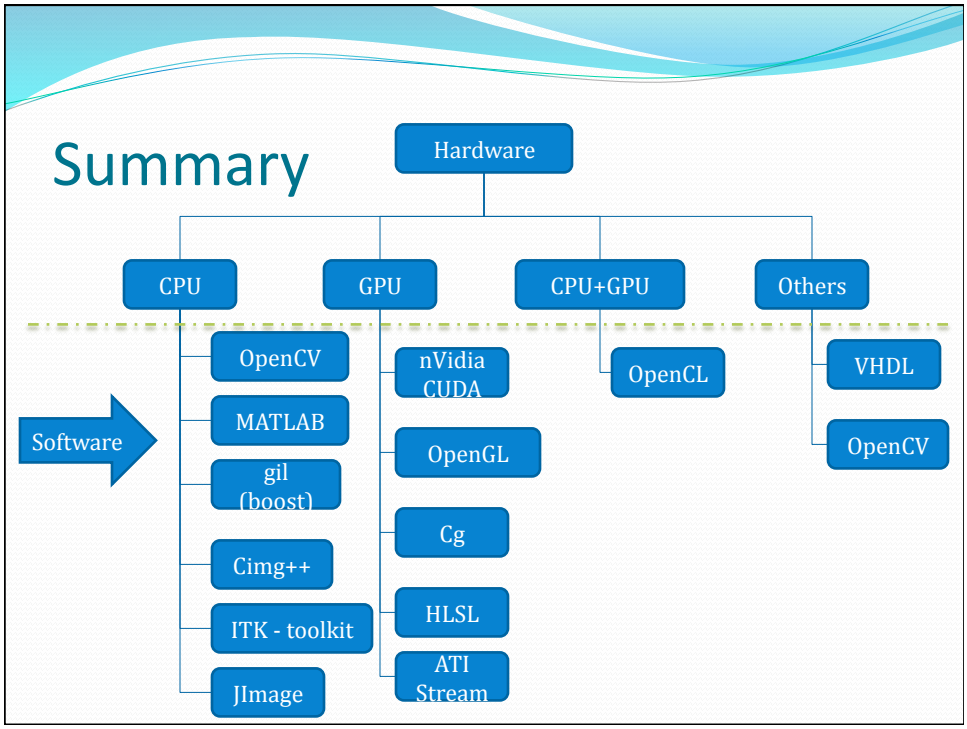
CUDA

Source: Nvidia – Adavances in GPU based Image Processing

OpenCL



- Combined use of CPU+GPU's for Vision computing.
- Requires explicit parallelism by the developer
- Explicit memory management for each device
- Recently explored approach for vision applications
- Support from many hardware vendors to create an open standard for parallel computing, graphics and media



Demos

- Dense Stereo on GPU
- CPU-GPU Performance Comparison
  (Canny edge Detection)