OpenVX Overview

June 2015
Vision Pipeline Challenges and Opportunities

Growing Camera Diversity

Flexible sensor and camera control to GENERATE an image stream

Diverse Vision Processors

Use efficient acceleration to PROCESS the image stream

Sensor Proliferation

Combine vision output with other sensor data on device
OpenVX - Vision Acceleration

- Royalty-free open standard API
  - Reliably accelerated by hardware vendors
  - Tightly defined conformance tests

- Targeted at low-power, real-time applications
  - Mobile and embedded platforms

- Portability across diverse heterogeneous processors
  - ISPs, Dedicated hardware, DSPs and DSP arrays, GPUs, Multi-core CPUs ...

- Doesn’t require high-power CPU/GPU Complex
  - Low-power host can setup and manage frame-rate vision processing pipeline

OpenVX extends easily re-usable vision acceleration to very low power domains
OpenVX Graphs - The Key to Efficiency

- OpenVX developers express a graph of image operations (‘Nodes’)
  - Nodes can be on any hardware or processor coded in any language
- Graph enables implementations to optimize for power and performance
  - E.g. Nodes may be fused by the implementation to eliminate memory transfers
  - E.g. Processing can be tiled to keep data entirely in local memory/cache
- Minimizes host interaction during frame-rate graph execution
  - Host processor can setup graph which can then execute almost autonomously
OpenVX 1.0 Functional Overview

- **Core data structures**
  - Images and Image Pyramids
  - Processing Graphs, Kernels, Parameters

- **Image Processing**
  - Arithmetic, Logical, and statistical operations
  - Multichannel Color and BitDepth Extraction and Conversion
  - 2D Filtering and Morphological operations
  - Image Resizing and Warping

- **Core Computer Vision**
  - Pyramid computation
  - Integral Image computation

- **Feature Extraction and Tracking**
  - Histogram Computation and Equalization
  - Canny Edge Detection
  - Harris and FAST Corner detection
  - Sparse Optical Flow

OpenVX 1.0 defines a framework for creating, managing and executing graphs. It focuses on a set of widely used functions (Nodes) to be accelerated. Widely used extensions are adopted into future versions of the core, and implementers can add functions as extensions. Khronos maintains an extension registry.
Example Graph - Stereo Machine Vision

OpenVX Graph

- Camera 1→Stereo Rectify with Remap→Compute Depth Map (User Node)→Detect and track objects (User Node)
- Camera 2→Stereo Rectify with Remap→Image Pyramid→Compute Optical Flow
- Object coordinates

Tiling extension enables user nodes (extensions) to also optimally run in local memory
Layered Vision Processing Ecosystem

- Lower-level compute APIs can be used to implement OpenVX nodes
  - Depending on the available processors
- Coding in OpenCL can provide portability across heterogeneous processors
  - ISPs, Dedicated hardware, DSPs and DSP arrays, GPUs, Multi-core CPUs ...

Implementers may choose to use OpenCL or Compute Shaders to implement OpenVX nodes on programmable processors ...

And then use OpenVX to enable a developer to easily connect those nodes into a graph

<table>
<thead>
<tr>
<th>Application Software</th>
<th>Engine/frameworks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Powerful, flexible</td>
</tr>
<tr>
<td></td>
<td>low-level APIs /</td>
</tr>
<tr>
<td></td>
<td>languages</td>
</tr>
<tr>
<td>Processor Hardware</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application</th>
<th>Programmable Vision Processors</th>
<th>Dedicated Vision Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenCL</td>
<td>C/C++</td>
<td></td>
</tr>
<tr>
<td>OpenGL ES</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Copyright Khronos Group 2015 - Page 7
OpenVX Status

- Finalized OpenVX 1.0 specification released October 2014
  - OpenVX 1.0.1 spec maintenance update released June 2015
  - www.khronos.org/openvx

- Khronos open source sample implementation of OpenVX 1.0 released
  - https://www.khronos.org/registry/vx/sample/openvx_sample_20141217.tar.gz

- Full conformance test suite and Adopters Program available
  - Test suite exercises graph framework and functionality of each OpenVX 1.0 nod
## OpenVX and OpenCV are Complementary

<table>
<thead>
<tr>
<th></th>
<th>OpenCV</th>
<th>OpenVX</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Implementation</strong></td>
<td>Community driven open source library</td>
<td>Open standard API designed to be implemented by hardware vendors</td>
</tr>
<tr>
<td><strong>Conformance</strong></td>
<td>Extensive OpenCV Test Suite but no formal Adopters program</td>
<td>Implementations must pass defined conformance test suite to use trademark</td>
</tr>
<tr>
<td><strong>Consistency</strong></td>
<td>Available functions can vary depending on implementation / platform</td>
<td>All core functions must be available in all conformant implementations</td>
</tr>
<tr>
<td><strong>Scope</strong></td>
<td>Very wide 1000s of imaging and vision functions Multiple camera APIs/interfaces</td>
<td>Tight focus on core hardware accelerated functions for mobile vision - but extensible Uses external/native camera API</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>Memory-based architecture Each operation reads and writes to memory</td>
<td>Graph-based execution Optimizable computation and data transfer</td>
</tr>
<tr>
<td><strong>Typical Use Case</strong></td>
<td>Rapid experimentation and prototyping - especially on desktop</td>
<td>Production development &amp; deployment on mobile and embedded devices</td>
</tr>
<tr>
<td><strong>Embedded Deployment</strong></td>
<td>Re-usable code</td>
<td>Callable library</td>
</tr>
</tbody>
</table>
Vision Processing Power Efficiency

- Wearables will need ‘always-on’ vision
  - With smaller thermal limit / battery than phones!
- GPUs have x10 imaging power efficiency over CPU
  - GPUs architected for efficient pixel handling
- Dedicated Hardware/DSPs can be even more efficient
  - With some loss of generality
- Mobile SOCs have space for more transistors
  - But can’t turn on at same time = Dark Silicon
  - Can integrate more gates ‘for free’ if careful how and when they are used

Potential for dedicated sensor/vision silicon to be integrated into Mobile Processors
But how will they be programmed for PORTABILITY and POWER EFFICIENCY?