OpenVX Ecosystem Overview
OpenVX Workshop, Embedded Vision Summit, May 2016

Neil Trevett | Khronos President
NVIDIA Vice President Developer Ecosystem
Khronos is an Industry Consortium of over 100 companies creating royalty-free, open standard APIs to enable software to access hardware acceleration for graphics, parallel compute and vision.
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

OpenKCam™ OpenVX™ StreamInput™
Accelerated Vision API Jungle

Vision Frameworks
Neural Net Libraries

cuDNN
OpenCV
clBLAS
OpenVX

Language-based Acceleration Frameworks

Explicit Kernels

OpenGL ES
Vulkan

NVIDIA CUDA
SYCL
OpenCL

GPU FPGA DSP Dedicated Hardware

© Copyright Khronos Group 2016 - Page 4
OpenVX and OpenCV are Complementary

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Community driven open source library</th>
<th>Open standard API designed to be implemented by hardware vendors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conformance</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>Consistency</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>Scope</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>Efficiency</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>Typical Use Case</td>
<td>Rapid experimentation and prototyping - especially on desktop</td>
<td>Production development &amp; deployment on mobile and embedded devices</td>
</tr>
<tr>
<td>Embedded Deployment</td>
<td>Re-usable code</td>
<td>Callable API</td>
</tr>
</tbody>
</table>
OpenVX - Low Power Vision Acceleration

- Precisely defined API for production deployment of vision acceleration
  - Targeted at real-time mobile and embedded platforms
- Higher abstraction than OpenCL for performance portability across diverse architectures
  - Multi-core CPUs, GPUs, DSPs and DSP arrays, ISPs, Dedicated hardware...
- Extends portable vision acceleration to very low power domains
  - Doesn’t require high-power CPU/GPU Complex or OpenCL precision
  - Low-power host can setup and manage frame-rate graph
OpenVX Graphs

- OpenVX developers express a graph of image operations (‘Nodes’)
  - Nodes can be on any hardware or processor coded in any language
- Graphs can execute almost autonomously
  - Possible to Minimize host interaction during frame-rate graph execution
- Graphs are the key to run-time optimization opportunities...

[Diagram of OpenVX Graph]
OpenVX Efficiency through Graphs..

Graph Scheduling
- Split the graph execution across the whole system: CPU / GPU / dedicated HW
- Faster execution or lower power consumption

Memory Management
- Reuse pre-allocated memory for multiple intermediate data
- Less allocation overhead, more memory for other applications

Kernel Merge
- Replace a sub-graph with a single faster node
- Better memory locality, less kernel launch overhead

Data Tiling
- Execute a sub-graph at tile granularity instead of image granularity
- Better use of data cache and local memory
Example Relative Performance

<table>
<thead>
<tr>
<th>Category</th>
<th>OpenCV (GPU accelerated)</th>
<th>OpenVX (GPU accelerated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic</td>
<td>1.1</td>
<td></td>
</tr>
<tr>
<td>Analysis</td>
<td>2.9</td>
<td></td>
</tr>
<tr>
<td>Filter</td>
<td>8.7</td>
<td></td>
</tr>
<tr>
<td>Geometric</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>2.5</td>
<td></td>
</tr>
</tbody>
</table>

NVIDIA implementation experience. Geometric mean of >2200 primitives, grouped into each categories, running at different image sizes and parameter settings.
Layered Vision Processing Ecosystem

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

And then developers can use OpenVX to enable a developer to easily *connect* those nodes into a graph.

<table>
<thead>
<tr>
<th>Application Software</th>
<th>Engines/frameworks</th>
<th>Processor Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenVX</td>
<td>C/C++</td>
<td></td>
</tr>
<tr>
<td>Programmable Vision</td>
<td>Dedicated Vision</td>
<td></td>
</tr>
<tr>
<td>Processors</td>
<td>Hardware</td>
<td></td>
</tr>
</tbody>
</table>

OpenVX enables the graph to be *extended* to include hardware architectures that don’t support programmable APIs.

The OpenVX graph enables implementers to *optimize* execution across diverse hardware architectures and drive to lower power implementations.
OpenVX 1.0 Shipping, OpenVX 1.1 Released!

- Multiple OpenVX 1.0 Implementations shipping - spec in October 2014
  - Open source sample implementation and conformance tests available

- OpenVX 1.1 Specification released 2\textsuperscript{nd} May 2016 at Embedded Vision Summit
  - Expands node functionality AND enhances graph framework
  - Sample source and conformance tests will be updated to OpenVX 1.1 in 1H16

- OpenVX is EXTENSIBLE
  - Implementers can add their own nodes at any time to meet customer and market needs
What’s New in OpenVX 1.1?

• Expanded node functionality AND enhanced graph framework
  - Plus many minor improvements and clarifications

• Laplacian pyramids
  - Computational photography use cases

• Targets - for execution flexibility on heterogeneous devices
  - Application can control on which accelerator to run nodes

• Median, erode and dilate image filters
  - Including custom patterns

• Improved read and write data to and from OpenVX objects
  - Easier to use and less error prone

• Improved API for extending OpenVX with user kernels
  - More convenience and flexibility
OpenVX Roadmap and Safety Critical APIs

OpenGL ES 1.0 - 2003
Fixed function graphics

OpenGL ES 2.0 - 2007
Shader programmable pipeline subset

OpenGL SC 1.0 - 2005
Fixed function graphics subset

OpenGL SC 2.0 - April 2016
Shader programmable pipeline subset

Experience and Guidelines

New Generation APIs for safety certifiable vision, graphics and compute
e.g. ISO 26262 and DO-178B/C

OpenVX Roadmap Discussions
Significantly broaden node functionality
In-graph neural nets
Programmable nodes (OpenCL or SPIR-V?)
Market-specific feature sets

OpenVX SC?

Small driver size
Advanced functionality
Graphics and compute
Thank You for Coming Today!

- PDF Quick Reference Cards available for all versions of Khronos specifications

- OpenVX Forums!

- Please give us your feedback on today’s workshop
  - We want to work to make these days as useful as possible

- Please talk to us about what you would like to see in the OpenVX roadmap
  - We are genuinely interested in your input!

- Any company or organization is welcome to join Khronos for a voice and a vote in any of its standards
  - www.khronos.org

- Neil Trevett
  - ntrevett@nvidia.com
  - @neilt3d