New Emerging Standards for Embedded Vision

Radhakrishna Giduthuri, AMD

September 2017
Khronos Connects Software to Silicon

Industry Consortium creating OPEN STANDARD APIs for hardware acceleration
Any company is welcome - one company one vote

ROYALTY-FREE specifications
State-of-the art IP framework protects members AND the standards

Conformance Tests and Adopters Programs for specification integrity and cross-vendor portability

Software

Low-level silicon APIs needed on every platform: graphics, parallel compute, vision, neural networks, augmented and virtual reality...

Silicon

International, non-profit organization
Membership and Adopters fees cover operating and engineering expenses

Strong industry momentum
100s of man years invested by industry experts

Well over a BILLION people use Khronos APIs Every Day...
Khronos Open Standards

3D for the Web
- VR/AR and games in-browser
- Efficiently delivering runtime 3D assets

Real-time 2D/3D
- Virtual and Augmented Reality displays
- Cross-platform gaming and UI
- CAD and Product Design

Vision and Neural Networks
- Embedded vision processing
- Scene analysis/understanding
- Neural Network inferencing

Parallel Computation
- Machine Learning acceleration
- Embedded vision processing
- High Performance Computing (HPC)
OpenVX

Wide range of vision hardware architectures
OpenVX provides a high-level Graph-based abstraction
->
Enables Graph-level optimizations!
Can be implemented on almost any hardware or processor!
->
Portable, Efficient Vision Processing!

- Vision Engines
- Middleware
- Applications
- GPU
- DSP
- Hardware

Vision Processing Graph

Power Efficiency

Computation Flexibility

Dedicated Hardware
Vision DSPs
GPU Compute
Multi-core CPU

Software Portability

Vision Node
CNN Nodes
Vision Node
## How OpenVX Compares to Alternatives

<table>
<thead>
<tr>
<th></th>
<th>OpenVX</th>
<th>OpenCV</th>
<th>OpenCL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Governance</strong></td>
<td>Open standard API designed to be implemented and shipped by IHVs</td>
<td>Community-driven, open source library</td>
<td>Open standard API designed to be implemented and shipped by IHVs</td>
</tr>
<tr>
<td><strong>Programming Model</strong></td>
<td>Graph defined with C API and then compiled for run-time execution</td>
<td>Immediate runtime function calls - reading to and from memory</td>
<td>Explicit kernels are compiled and executed via run-time API</td>
</tr>
<tr>
<td><strong>Built-in Vision Functionality</strong></td>
<td>Small but growing set of popular functions</td>
<td>Vast. Mainly on PC/CPU</td>
<td>None. User programs their own or call vision library over OpenCL</td>
</tr>
<tr>
<td><strong>Target Hardware</strong></td>
<td>Any combination of processors or non-programmable hardware</td>
<td>Mainly PCs and GPUs</td>
<td>Any heterogeneous combination of IEEE FP-capable processors</td>
</tr>
<tr>
<td><strong>Optimization Opportunities</strong></td>
<td>Pre-declared graph enables significant optimizations</td>
<td>Each function reads/writes memory. Power performance inefficient</td>
<td>Any execution topology can be explicitly programmed</td>
</tr>
<tr>
<td><strong>Conformance</strong></td>
<td>Implementations must pass conformance to use trademark</td>
<td>Extensive Test Suite but no formal Adopters program</td>
<td>Implementations must pass conformance to use trademark</td>
</tr>
<tr>
<td><strong>Consistency</strong></td>
<td>All core functions must be available in conformant implementations</td>
<td>Available functions vary depending on implementation / platform</td>
<td>All core functions must be available in all conformant implementations</td>
</tr>
</tbody>
</table>
Neural Net Workflow with Khronos Standards

OpenVX is an open, royalty-free standard for cross platform acceleration of computer vision and neural network applications.

NNEF is a Cross-vendor Neural Net file format
Encapsulates network formal semantics, structure, data formats, commonly-used operations (such as convolution, pooling, normalization, etc.)
Problem: Neural Network Fragmentation

Neural Network Inferencing Fragmentation toll on Applications

Every Application Needs know about Every Accelerator API

Neural Network Training and Inferencing Fragmentation

Every Tool Needs an Exporter to Every Accelerator
Solving Neural Network Fragmentation

Neural Network Inferencing Fragmentation toll on Applications

Vision/Al Application → OpenVX → Inference Engine 1 → Hardware 1
Inference Engine 2 → Hardware 2
Inference Engine 3 → Hardware 3

Neural Network Training and Inferencing Fragmentation

Caffe, TensorFlow, CNTK, Theano → NNEF → Inference Engine 1
Inference Engine 2
Inference Engine 3

Optimization and processing tools
OpenVX in a nut-shell

- **OpenVX contains**
  - a set of memory objects that abstract the physical memory
  - a library of predefined and customizable vision and neural network functions
    - Image processing and image filtering
    - Feature detectors
    - Neural networks
    - Control flow
  - a graph-based execution model to combine function enabling both task and data-independent execution

- **OpenVX is defined as a C API**
  - object-oriented design
  - synchronous and asynchronous execution model
  - extend functionality using enum and callback functions
OpenVX - Graph-Level Abstraction

- OpenVX developers express a graph of image operations (‘Nodes’)
  - Using a C API
- Nodes can be executed on any hardware or processor coded in any language
  - Implementers can optimize under the high-level graph abstraction
- Graphs are the key to run-time power and performance optimizations
  - E.g. Node fusion, tiled graph processing for cache efficiency etc.

Feature Extraction Example Graph
OpenVX Efficiency through Graphs..

**Graph Scheduling**
- Split the graph execution across the whole system: CPU / GPU / dedicated HW

**Memory Management**
- Reuse pre-allocated memory for multiple intermediate data

**Kernel Fusion**
- Replace a sub-graph with a single faster node

**Data Tiling**
- Execute a sub-graph at tile granularity instead of image granularity

- Faster execution or lower power consumption
- Less allocation overhead, more memory for other applications
- Better memory locality, less kernel launch overhead
- Better use of data cache and local memory
Simple Edge Detector in OpenVX

```c
vx_graph g = vxCreateGraph();

vx_image input = vxCreateImage(1920, 1080);
vx_image output = vxCreateImage(1920, 1080);
vx_image horiz = vxCreateVirtualImage(g);
vx_image vert = vxCreateVirtualImage(g);
vx_image mag = vxCreateVirtualImage(g);

vxSobel3x3Node(g, input, horiz, vert);
vxMagnitudeNode(g, horiz, vert, mag);
vxThresholdNode(g, mag, THRESH, output);

status = vxVerifyGraph(g);
status = vxProcessGraph(g);
```

Declare Input and Output Images
Declare Intermediate Images
Construct the Graph topology
Compile the Graph
Execute the Graph
OpenVX Evolution

OpenVX 1.0 and OpenVX 1.1

AMD OpenVX Tools
- Open source, highly optimized for x86 CPU and OpenCL for GPU
- “Graph Optimizer” looks at entire processing pipeline and removes, replaces, merges functions to improve performance and bandwidth
- Scripting for rapid prototyping, without re-compiling, at production performance levels

New Functionality
- Conditional node execution
- Feature detection
- Classification operators
- Expanded imaging operations

Extensions
- Neural Network Acceleration
- Graph Save and Restore
- 16-bit image operation

Safety Critical
- OpenVX 1.1 SC for safety-certifiable systems

New Functionality Under Discussion
- NNEF Import
- Programmable user kernels with accelerator offload
- Streaming/pipelining

OpenVX Roadmap

OpenVX 1.2
Spec released May 2017
Will release conformance tests soon

Conformant Implementations

AMD
Cadence
CEVA
Exxelia
Imagination
Intel
NVIDIA
Synopsys
Texas Instruments
VeriSilicon

© Copyright Khronos Group 2017 - Page 13
OpenVX SC - Safety Critical Vision Processing

- OpenVX 1.1 - based on OpenVX 1.1 main specification
  - Enhanced determinism
  - Specification identifies and numbers requirements
- MISRA C clean per KlocWorks v10
- Divides functionality into “development” and “deployment” feature sets
  - Adds requirement to support import/export extension

Diagram:
- OpenVX SC Development Feature Set (Create Graph) → Verify Export
- Binary format
- Import
- OpenVX SC Deployment Feature Set (Execute Graph)

Entire graph creation API
- Implementation dependent format
- No graph creation API
OpenVX 1.2 and Neural Net Extension

- Convolution Neural Network topologies can be represented as OpenVX graphs
  - Layers are represented as OpenVX nodes
  - Layers connected by multi-dimensional tensors objects
  - Layer types include convolution, activation, pooling, fully-connected, soft-max
  - CNN nodes can be mixed with traditional vision nodes
  - Only inference is supported

- Import/Export Extension
  - Efficient handling of network Weights/Biases or complete networks

- OpenVX will be able to import NNEF files into OpenVX Neural Nets
OpenVX Neural Network Extension

- Tensor types of INT16, INT7.8, INT8, and U8 are supported
  - Other types may be supported by a vendor
  - Algorithms for quantization are needed for conversion from trained float models to INT16, INT7.8, INT8, and U8.
  - Extension is split to 2 variants. Full 16 and 8 bit support. And 8 bit only support.

- Conformance tests will be up to some “tolerance” in precision
  - To allow for optimizations, e.g., weight compression

- Eight neural network “layer” nodes:

<table>
<thead>
<tr>
<th>vxActivationLayer</th>
<th>vxConvolutionLayer</th>
<th>vxDeconvolutionLayer</th>
</tr>
</thead>
<tbody>
<tr>
<td>vxFullyConnectedLayer</td>
<td>vxNormalizationLayer</td>
<td>vxPoolingLayer</td>
</tr>
<tr>
<td>vxSoftmaxLayer</td>
<td>vxROIPoolingLayer</td>
<td>...</td>
</tr>
</tbody>
</table>
Layered Vision/ Neural Net Ecosystem

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

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

The OpenVX graph abstraction enables implementers to *optimize* execution across diverse hardware architectures for optimal power and performance.

**OpenVX Roadmap**
Enhanced compute - especially useful for vision and inferencing on mobile and Android platforms.

**OpenCL Roadmap**
Flexible precision for widespread deployment on low-cost embedded processors.

And then implementors can use OpenVX to enable a developer to easily *connect* those nodes into a graph.
OpenVX Benefits and Resources

- **Faster development of efficient and portable vision applications**
  - Developers are protected from hardware complexities
  - No platform-specific performance optimizations needed

- **Graph description enables significant automatic optimizations**
  - Scheduling, memory management, kernel fusion, and tiling

- **Performance portability to diverse hardware**
  - Hardware agility for different use case requirements
  - Application software investment is protected as hardware evolves

- **OpenVX Resources**
  - OpenVX Overview
    - [https://www.khronos.org/openvx](https://www.khronos.org/openvx)
  - OpenVX Specifications: current, previous, and extensions
    - [https://www.khronos.org/registry/OpenVX](https://www.khronos.org/registry/OpenVX)
  - OpenVX Resources: implementations, tutorials, reference guides, forum, etc.
    - [https://www.khronos.org/openvx/resources](https://www.khronos.org/openvx/resources)
    - [https://forums.khronos.org](https://forums.khronos.org)