Standards for Vision Processing and Neural Networks

www.khronos.org
Topics and Speakers

- **Intro and OpenCL**
  - Neil Trevett, NVIDIA

- **SYCL**
  - Ralph Potter, Codeplay

- **NNEF**
  - Peter McGuinness, Independent Consultant

- **OpenVX**
  - Radha Giduthuri, AMD
Khronos is an International Industry Consortium of over 100 companies creating royalty-free, open standard APIs to enable software to access hardware acceleration for 3D graphics, Virtual and Augmented Reality, Parallel Computing, Neural Networks and Vision Processing.
Vision and Neural Net Acceleration Challenge

Neural Net Training Frameworks

Desktop and Cloud Hardware

Vision and Neural Net Inferencing Runtime

Trained Networks

Vision/Al Applications

Embedded/Mobile/Desktop Vision/Inferencing Hardware

Neural Net Training Frameworks

Desktop and Cloud Hardware
OpenCL - Low-level Parallel Programming

- Low-level, explicit programming of heterogeneous parallel compute resources
  - One code tree can be executed on CPUs, GPUs, DSPs and FPGA ...

- OpenCL C or C++ language to write kernel programs to execute on any compute device
  - Platform Layer API - to query, select and initialize compute devices
  - Runtime API - to build and execute kernels programs on multiple devices

- The programmer gets to control:
  - What programs execute on what device
  - Where data is stored in various speed and size memories in the system
  - When programs are run, and what operations are dependent on earlier operations
OpenCL 2.2 Released in May 2017

2011
OpenCL 1.2
Becomes industry baseline for heterogeneous parallel computing

2013
OpenCL 2.0
Enables new class of hardware
SVM
Generic Addresses
On-device dispatch

2015
OpenCL 2.1
SPIR-V 1.0
SPIR-V in Core
Kernel Language
Flexibility

2017
OpenCL 2.2
SPIR-V 1.2
OpenCL C++ Kernel Language
Static subset of C++14
Templates and Lambdas
SPIR-V 1.2
OpenCL C++ support
Pipes
Efficient device-scope communication between kernels
Code Generation Optimizations
- Specialization constants at SPIR-V compilation time
- Constructors and destructors of program scope global objects
- User callbacks can be set at program release time

https://www.khronos.org/opencl/
New Open Source Engagement Model

- **Khronos is open sourcing specification sources, conformance tests, tools**
  - Merge requests welcome from the community (subject to review by OpenCL working group)

- **Deeper Community Enablement**
  - Mix your own documentation!
  - Contribute and fix conformance tests
  - Fix the specification, headers, ICD etc.
  - Contribute new features (carefully)

- **Spec and Ref Language Source**
  - Contributions and Distribution under Apache 2.0
  - Khronos builds and Ratifies Canonical Specification under Khronos IP Framework. No changes or re-hosting allowed

- **Spec Build System and Scripts**
  - Spec and Ref Language Source and derivative materials.
  - Re-mixable under CC-BY 4.0 by the industry and community

- **Conformance Test Suite Source**
  - Contributions and Distribution under Apache 2.0

- **Kkhronos Adopters Program**
  - Anyone can test any implementation at any time
  - Conformant Implementations can use trademark and are covered by Khronos IP Framework

- **Source Materials for Specifications and Reference Documentation**
  - CONTRIBUTED Under Khronos IP Framework
  - You won’t assert patents against conformant implementations, and license copyright for Khronos use
Khronos Advisory Panels

Specification drafts and invitations for requirements and feedback

Working Group

Khronos Members
Any company can join Membership Fee Sign NDA and IP Framework

Shared Email list and Repository
Hosted by Khronos Under Khronos NDA

Requirements and feedback on specification drafts

Advisory Panel
Khronos Advisors Invited industry experts $0 Cost Sign NDA and IP Framework

Advisory Panels Active for Vulkan, OpenCL/SYCL and NNEF
OpenCL Ecosystem

Hardware Implementers
Desktop/Mobile/Embedded/FPGA

OpenCL 2.2 - Top to Bottom C++

SYCL™
Single Source C++ Programming

OpenCL
Core API and Language Specs

SPIR™
Portable Kernel Intermediate Language

100s of applications using OpenCL acceleration
Rendering, visualization, video editing, simulation, image processing, vision and neural network inferencing

© Copyright Khronos Group 2017 - Page 9
OpenCL as Language/Library Backend

- Caffe: C++ based Neural network framework
- Halide: Language for image processing and computational photography
- MulticoreWare: open source project on Bitbucket
- SYCL: Single Source C++ Programming for OpenCL
- OpenCV: Java language extensions for parallelism
- OpenACC: Vision processing open source project
- Compiler directives for Fortran, C and C++
- aparapi: Open source software library for machine learning
- Tensorflow: Open source software library

Hundreds of languages, frameworks and projects using OpenCL to access vendor-optimized, heterogeneous compute runtimes

Over 4,000 GitHub repositories using OpenCL: tools, applications, libraries, languages - up from 2,000 two years ago
Market Demand for Universal 3D Portability

Games Engines  Native 3D Apps  Browser Engines

Community Outreach at GDC 2017
Create a hybrid Portability API?

Feedback - AVOID CREATING A FOURTH API!!!
Would need new specification, CTS, Documentation.
Additional developer learning curve.
A whole new specification to name, brand, promote.
Would INCREASE industry fragmentation

Vulkan Universally Portable Subset

Games Engines  Native 3D Apps  Browser Engines

JavaScript and WebAssembly Native bindings for 'nexgen WebGL'

MAP Vulkan to Metal and DX12
Tools Layers API Libraries Shader Translators

MAP Vulkan to Metal and DX12
Tools Layers API Libraries Shader Translators
Vulkan Portability TSG Process

Vulkan Portability Deliverables
1. Vulkan Subset Diff Spec
2. Vulkan Subset Development Layer
3. Vulkan Subset API Library over DX12/Metal
4. SPIRV-Cross Translator
5. Vulkan Subset Conformance Tests

Layers, APIs, Translators and Tests all to be developed and released in open source

Possible proposals for Vulkan extensions for enhanced portability (and possibly Web robustness) sent to Vulkan WG

Open source project with identical goals already underway - come and help!
https://github.com/gfx-rs/gfx

API Overlap Analysis

Identify Vulkan features not directly mappable to DX12 and Metal

Expand/test existing open source SPIRV-Cross Tool

New Vulkan functionality may affect the overlap analysis

Metal Shading Language
HLSL

SPIR-V

© Copyright Khronos Group 2017 - Page 13
OpenCL and Vulkan

OpenCL for DSPs
- Embedded imaging, vision and inferencing
- Flexible reduced precision
- Conformance without IEEE 32 Floating Point
- Explicit DMA

SYCL 1.2
C++11 Single source programming
2011

SYCL 2.1
SPIR-V in Core
OpenCL 2.1

SYCL 2.2
C++14 Single source programming
2015

SYCL 2.2
C++ Kernel Language
OpenCL 2.2

SPIR
OpenCL

Industry working to bring Heterogeneous compute to standard ISO C++
C++17 Parallel STL hosted by Khronos
Executors - for scheduling work
“Managed pointers” or “channels” - for sharing data

Help bring OpenCL-class compute to Vulkan

© Copyright Khronos Group 2017 - Page 14
Clspv OpenCL C to Vulkan Compiler

- Experimental collaboration between Google, Codeplay, and Adobe
  - Successfully tested on over 200K lines of Adobe OpenCL C production code
  - Released in open source https://github.com/google/clspv

- Uses new Vulkan extensions to support OpenCL C compute operations
  - VK_KHR_16bit_storage/SPV_KHR_16bit_storage
  - VK_KHR_variable_pointers/SPV_KHR_variable_pointers

- Compiles OpenCL C’s programming model to Vulkan’s SPIR-V execution environment
  - Proof-of-concept that OpenCL compute can be brought seamlessly to Vulkan
What is SYCL?

High-level C++ abstraction layer for OpenCL
Full coverage for all OpenCL features
Interop to enable existing OpenCL code with SYCL
Single-source compilation
Automatic scheduling of data movement

Diagram:
- Standard C++ Source
- SYCL Device Compiler
- Host Executable
- Device Code

© Copyright Khronos Group 2017 - Page 17
SYCL Example

// Create a device queue.
cl::sycl::queue device_queue;

// Create buffers.
cl::sycl::range<1>  n_items{array_size};
cl::sycl::buffer<cl::sycl::cl_int, 1> in_buffer(in.data(), n_items);
cl::sycl::buffer<cl::sycl::cl_int, 1> out_buffer(out.data(), n_items);

// Submit a kernel and associated data movement operations.
device_queue.submit([&](cl::sycl::handler &cgh) {
    // Defines the kernels access requirements.
    auto in_accessor = in_buffer.get_access<cl::sycl::access::mode::read>(cgh);
    auto out_accessor = out_buffer.get_access<cl::sycl::access::mode::write>(cgh);
    // Defines the kernel itself.
    cgh.parallel_for<class VecScalMul>(n_items, [=](cl::sycl::id<1> wiID) {
        out_accessor[wiID] = in_accessor[wiID]*2;
    });
});
SYCL Ecosystem

- Applications
- C++ Template Libraries
- SYCL for OpenCL
- OpenCL
- OpenCL-enabled Accelerators
Separating Storage & Access

Buffers manage data across host CPU and one or more devices.

Buffers and accessors type safe access across host and device.

Accessors are used to describe access.

Command Group

CPU

GPU
Data Dependency Task Graphs

Buffer A
- Read Accessor
- Write Accessor

Buffer B
- Read Accessor
- Write Accessor

Buffer C
- Read Accessor
- Read Accessor
- Write Accessor

Buffer D

CG A

CG B

CG C
Benefits of Data Dependency Graphs

- Allows you to describe your problems in terms of relationships
  - Don’t need to enqueue explicit copies
- Removes the need for complex event handling
  - Dependencies between device functions are automatically constructed
  - Allows the runtime to make data movement optimizations
  - Preemptively copy data to a device before executing device functions
- Avoid unnecessarily copying data back to the host after executing a device function
  - Avoid copying data to a device that you don’t need
## Compared to C++ Kernel Language

<table>
<thead>
<tr>
<th>C++ Kernel Language</th>
<th>SYCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard OpenCL host API</td>
<td>Modern C++ abstraction of OpenCL APIs</td>
</tr>
<tr>
<td>Explicit control of data movement</td>
<td>Automatic scheduling of data movement</td>
</tr>
<tr>
<td>C++ templates within kernels</td>
<td>Kernels themselves can be templates</td>
</tr>
<tr>
<td>Extend existing OpenCL kernels with templates</td>
<td>Provide OpenCL support to template-heavy libraries and applications</td>
</tr>
</tbody>
</table>
Convergence with ISO C++

Industry working to bring heterogeneous compute to standard ISO C++
C++17 Parallel STL hosted by Khronos
SYCL has lead to several ISO C++ proposals for enabling heterogeneous compute
  Executors - for scheduling work
  Data Movement in C++ - position paper on data movement for heterogeneous compute
  “Managed pointers” or “channels” - for sharing data
Hoping to target C++ 20 but timescales are tight
Parallel STL

Open-source implementation of the ISO C++ parallelism TS (N4505)
https://github.com/KhronosGroup/SyclParallelSTL

```cpp
std::vector<int> v = {...};

sycl::sycl_execution_policy<class sort_kernel> policy1;
sort(policy1, v.begin(), v.end());

sycl::sycl_execution_policy<class negate_kernel> policy2;
transform(policy2, v.begin(), v.end(), v.begin(), std::negate<int>(()));
```
Eigen Linear Algebra Library

SYCL backend in mainline
Focused on Tensor support, providing support for machine learning/CNNs
Equivalent coverage to CUDA
Working on optimization for various hardware architectures (CPU, desktop and mobile GPUs)
https://bitbucket.org/eigen/eigen/
TensorFlow

SYCL backend support for all major CNN operations
Complete coverage for major image recognition networks
  GoogLeNet, Inception-v2, Inception-v3, ResNet, ....
Ongoing work to reach 100% operator coverage and optimization for various hardware architectures (CPU, desktop and mobile GPUs)

https://github.com/tensorflow/tensorflow/tensorflow

TensorFlow, the TensorFlow logo and any related marks are trademarks of Google Inc.
SYCL Ecosystem

• Single-source heterogeneous programming using STANDARD C++
  - Use C++ templates and lambda functions for host & device code
  - Layered over OpenCL

• Fast and powerful path for bring C++ apps and libraries to OpenCL
  - C++ Kernel Fusion - better performance on complex software than hand-coding
  - Halide, Eigen, Boost.Compute, SYCLBLAS, SYCL Eigen, SYCL TensorFlow, SYCL GTX
  - triSYCL, ComputeCpp, VisionCpp, ComputeCpp SDK ...

• More information at http://sycl.tech

Developer Choice
The development of the two specifications are aligned so code can be easily shared between the two approaches

C++ Kernel Language
Low Level Control
‘GPGPU’-style separation of device-side kernel source code and host code

Single-source C++
Programmer Familiarity
Approach also taken by C++ AMP and OpenMP
**NNEF - Solving Neural Net Fragmentation**

Before NNEF - NN Training and Inferencing Fragmentation

- Caffe
- TensorFlow
- Theano
- NN Authoring Framework 1
- NN Authoring Framework 2
- NN Authoring Framework 3
- Inference Engine 1
- Inference Engine 2
- Inference Engine 3

Every Tool Needs an Exporter to Every Accelerator

With NNEF - NN Training and Inferencing Interoperability

- Caffe
- TensorFlow
- Theano
- NN Authoring Framework 1
- NN Authoring Framework 2
- NN Authoring Framework 3

Optimization and processing tools

**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.)
NNEF Usage Model

Structure only
OR
Structure plus weights
(quantized or not)

Trained Format

Training Framework

NNEF

Graph Compiler

Inference Engine

Third Party Tools

Vendor Format

Structure plus weights
(quantized or not). Can be augmented with private data

(quantized or not)

Can be augmented with private data
NNEF Status and Roadmap

- V1.0 is under development, industry comments are being sought now
  - NNEF has formed an advisory panel, you are invited today to participate

- First version will focus on interface between framework and embedded inference engines
  - But will allow training as secondary goal

- Support ‘First cut’ range of network types
  - Field is moving very fast but we aim to keep up with developments

- NNEF Roadmap
  - Track development of new network types
  - Allow authoring and retraining (3rd party tools)
  - Address a wider range of applications (outside vision apps)
  - Increase the expressive power of the format
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 Processing Graph

Vision Engines
Middleware
Applications
GPU
DSP
Hardware
Software Portability

Vision Node
Vision Node
Vision Node

Dedicated Hardware
Vision DSPs
GPU Compute
Multi-core CPU

Power Efficiency
Computation Flexibility

© Copyright Khronos Group 2017 - Page 34
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.

```
OpenVX Graph

Camera Input
  →

RGB Frame
  → Color Conversion
  → YUV Frame
  → Channel Extract
  → Gray Frame
  → Image Pyramid

Optical Flow
  → Array of Keypoints

Pyr_t

Ftr_{t-1}

Harris Track

Array of Features

Rendering Output
```
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 Fusion**
- 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
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

<table>
<thead>
<tr>
<th>Conformant Implementations</th>
<th>Conformant Implementations</th>
<th>New Functionality</th>
<th>New Functionality</th>
<th>Safety Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OpenVX 1.0</strong> Spec released October 2014</td>
<td><strong>OpenVX 1.1</strong> Spec released May 2016</td>
<td><strong>OpenVX 1.2</strong> Spec released May 2017</td>
<td><strong>OpenVX 1.1 SC for safety-certifiable systems</strong></td>
<td><strong>OpenVX 1.2 SC for safety-certifiable systems</strong></td>
</tr>
</tbody>
</table>

### 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
- Expanded Nodes Functionality
- Enhanced Graph Framework
- Conditional node execution
- Feature detection
- Classification operators
- Expanded imaging operations
- Neural Network Acceleration
- Graph Save and Restore
- 16-bit image operation

### Extensions
- Programmable user kernels with accelerator offload
- Streaming/pipelining

### Safety Critical
- OpenVX 1.1 SC for safety-certifiable systems
- OpenVX 1.2 SC for safety-certifiable systems
New OpenVX 1.2 Functions

- **Feature detection**: find features useful for object detection and recognition
  - Histogram of gradients - HOG
  - Local binary patterns - LBP
- **Template matching**: find features useful for object detection and recognition
  - Line finding
- **Classification**: detect and recognize objects in an image based on a set of features
  - Import a classifier model trained offline
  - Classify objects based on a set of input features
- **Image Processing**: transform an image
  - Generalized nonlinear filter: Dilate, erode, median with arbitrary kernel shapes
  - Non maximum suppression: Find local maximum values in an image
  - Edge-preserving noise reduction
- **Conditional execution & node predication**: Selectively execute portions of a graph based on a true/false predicate
- **Many, many minor improvements**
- **New Extensions**
  - **Import/export**: compile a graph; save and run later
  - **16-bit support**: signed 16-bit image data
  - **Neural networks**: Layers are represented as OpenVX nodes
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

- 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

- Two main parts: (1) a tensor object and (2) a set of CNN layer nodes
- A vx_tensor is a multi-dimensional array that supports at least 4 dimensions

- Tensor creation and deletion functions
- Simple math for tensors
  - Element-wise Add, Subtract, Multiply, TableLookup, and Bit-depth conversion
  - Transposition of dimensions and generalized matrix multiplication
  - vxCopyTensorPatch, vxQueryTensor (#dims, dims, element type, Q)

1-D tensor: [6]  i.e., 6-element vector
2-D tensor: [6, 4]  i.e., 6 by 4 matrix
3-D tensor: [6, 4, 5]
4-D tensor: [6, 4, 5, 3]
OpenVX Neural Network Extension

- Tensor types of INT16, INT7.8, INT8, and U8 are supported
  - Other types may be supported by a vendor
- 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>vxROI PoolingLayer</td>
<td>vxROI PoolingLayer</td>
<td></td>
</tr>
</tbody>
</table>
Safety Critical APIs

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

OpenGL SC 1.0 - 2005
Fixed function graphics subset

OpenGL SC 2.0 - April 2016
Shader programmable pipeline subset

OpenGL ES 1.0 - 2003
Fixed function graphics

OpenGL ES 2.0 - 2007
Shader programmable pipeline

OpenVX SC 1.1 Released 1st May 2017
Restricted “deployment” implementation executes on the target hardware by reading the binary format and executing the pre-compiled graphs

Khronos SCAP
‘Safety Critical Advisory Panel’
Guidelines for designing APIs that ease system certification.
Open to Khronos member AND industry experts

OpenCL SC TSG Formed
Working on OpenCL SC 1.2
Eliminate Undefined Behavior
Eliminate Callback Functions
Static Pool of Event Objects

Experience and Guidelines

OpenCL SC TSG Formed
Working on OpenCL SC 1.2
Eliminate Undefined Behavior
Eliminate Callback Functions
Static Pool of Event Objects

Khronos SCAP
‘Safety Critical Advisory Panel’
Guidelines for designing APIs that ease system certification.
Open to Khronos member AND industry experts

OpenVX SC 1.1 Released 1st May 2017
Restricted “deployment” implementation executes on the target hardware by reading the binary format and executing the pre-compiled graphs

OpenGL SC 1.0 - 2005
Fixed function graphics subset

OpenGL SC 2.0 - April 2016
Shader programmable pipeline subset

OpenGL ES 1.0 - 2003
Fixed function graphics

OpenGL ES 2.0 - 2007
Shader programmable pipeline

OpenCL SC TSG Formed
Working on OpenCL SC 1.2
Eliminate Undefined Behavior
Eliminate Callback Functions
Static Pool of Event Objects

Khronos SCAP
‘Safety Critical Advisory Panel’
Guidelines for designing APIs that ease system certification.
Open to Khronos member AND industry experts

OpenVX SC 1.1 Released 1st May 2017
Restricted “deployment” implementation executes on the target hardware by reading the binary format and executing the pre-compiled graphs

OpenGL SC 1.0 - 2005
Fixed function graphics subset

OpenGL SC 2.0 - April 2016
Shader programmable pipeline subset

OpenGL ES 1.0 - 2003
Fixed function graphics

OpenGL ES 2.0 - 2007
Shader programmable pipeline

OpenCL SC TSG Formed
Working on OpenCL SC 1.2
Eliminate Undefined Behavior
Eliminate Callback Functions
Static Pool of Event Objects

Khronos SCAP
‘Safety Critical Advisory Panel’
Guidelines for designing APIs that ease system certification.
Open to Khronos member AND industry experts

OpenVX SC 1.1 Released 1st May 2017
Restricted “deployment” implementation executes on the target hardware by reading the binary format and executing the pre-compiled graphs

OpenGL SC 1.0 - 2005
Fixed function graphics subset

OpenGL SC 2.0 - April 2016
Shader programmable pipeline subset

OpenGL ES 1.0 - 2003
Fixed function graphics

OpenGL ES 2.0 - 2007
Shader programmable pipeline

OpenCL SC TSG Formed
Working on OpenCL SC 1.2
Eliminate Undefined Behavior
Eliminate Callback Functions
Static Pool of Event Objects

Khronos SCAP
‘Safety Critical Advisory Panel’
Guidelines for designing APIs that ease system certification.
Open to Khronos member AND industry experts

OpenVX SC 1.1 Released 1st May 2017
Restricted “deployment” implementation executes on the target hardware by reading the binary format and executing the pre-compiled graphs
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

Entire graph creation API

Implementation dependent format

No graph creation API
## 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>
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
  - Tiling and kernel fusion

- 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, etc.
  - [https://www.khronos.org/openvx/resources](https://www.khronos.org/openvx/resources)
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.

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

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

**Vulkan 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.

**Application**

- **Programmable Vision Processors**
- **Dedicated Vision Hardware**

© Copyright Khronos Group 2017 - Page 47
Any Questions?

- Khronos working on a comprehensive set of solutions for vision and inferencing
  - Layered ecosystem that include multiple developer and deployment options
- These slides and further information on all these standards at Khronos
  - [www.khronos.org](http://www.khronos.org)
- Please let us know if we can help you participate in any Khronos activities!
  - You are very welcome - and we appreciate your input!
- Please contact us with any comments or questions!
  - Neil Trevett | ntrevett@nvidia.com | @neilt3d
BOF BLITZ
AFTER-PARTY

Come Back at 5:45
for the After-Party

SPONSORED BY:

NHK
NHK

WWW.KHRONOS.ORG