OpenVX 1.3 - Open Standard for Computer Vision Software Acceleration

Kiriti Nagesh Gowda
OpenVX Chair | The Khronos Group
Staff Engineer | AMD

AMD

KHRONOS GROUP

Agenda

- OpenVX - Open, Royalty-free Standard for Computer Vision
- OpenVX 1.3 - An Overview
- Conformant Implementations
- Sample Applications - Case Study
- Open-Source Tools
- OpenVX for Raspberry Pi - Updates & New Features
- Camera Extension - Future Work
- Summary

Open, Royalty-free Standard for Computer Vision

OpenVX™ is an open, royalty-free API standard for a cross-platform acceleration of computer vision applications

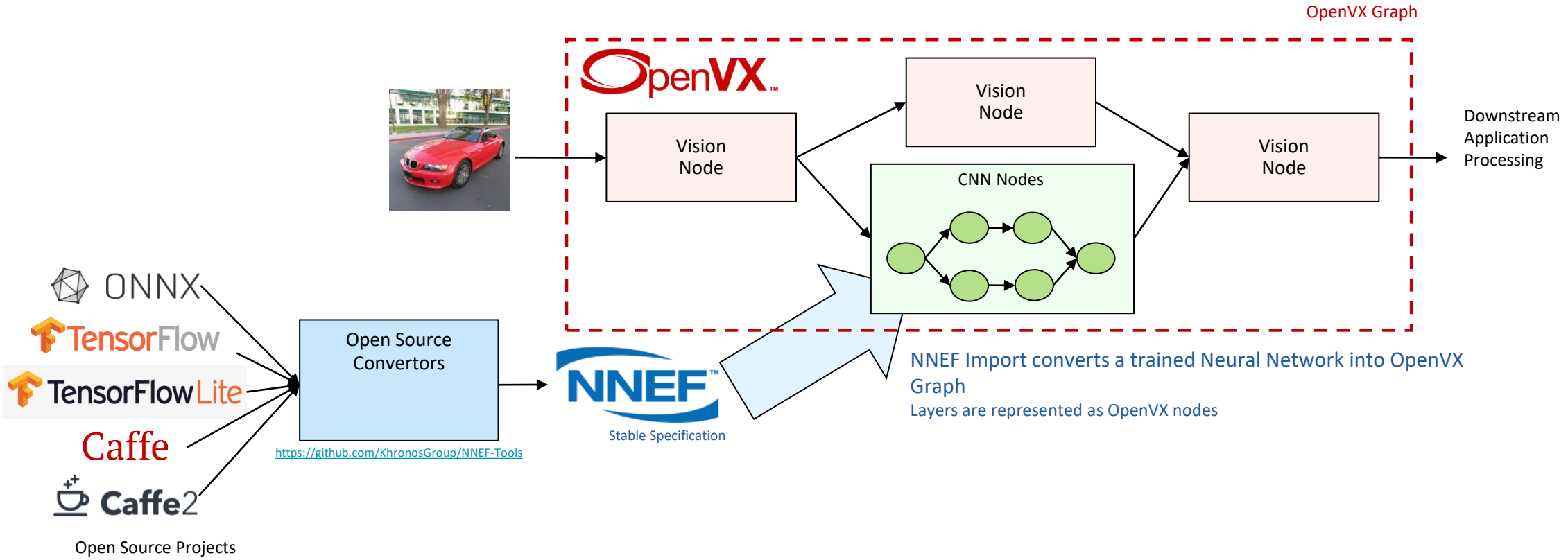
High-level graph-based abstraction for portable, efficient vision processing

Open, Royalty-free Standard for Computer Vision

- Optimized OpenVX drivers created, optimized and shipped by processor vendors
- Implementable on almost any hardware or processor with performance portability
- Graph can contain vision and neural net nodes for global optimization
- Run-time graph execution need very little host CPU interaction

Open, Royalty-free Standard for Computer Vision

OS Support: Linux, Windows, Android, iOS, Raspbian, Embedded OS, no OS



OpenVX 1.3 - An Overview

- OpenVX API had grown to an extensive set of functions
- There was an interest in creating implementations that target a set of features rather than covering the entire OpenVX API.
- Preventing fragmentation regarding which implementations offer which features

OpenVX 1.3 - An Overview

- The specification defines a collection of “feature sets” that form coherent and useful subsets of the OpenVX API.
- Implementors have the option to test for conformance to only one or a few feature sets rather than the entire API.
- Implementations that choose this option must clearly identify which feature sets they support in their documentation.

OpenVX 1.3 - An Overview

- Three Conformance Feature Sets:
  - Vision - OpenVX 1.1 equivalent vision functions
  - Neural Network - OpenVX 1.2 equivalent neural-network functions, plus the Neural Network extension, and the tensor object
  - NNEF Import - Kernel import plus the tensor object

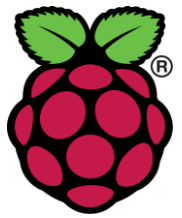
OpenVX 1.3 - An Overview

- Two Optional Feature Sets:
  - U1 - binary image support
  - Enhanced Vision - vision functions introduced in OpenVX 1.2
- One Organizational Feature Set:
  - Base Feature Set - basic graph infrastructure
- One Informational Feature Set:
  - Deployment Feature Set - for Safety Critical usage

Conformant OpenVX Implementations

- Conformant Implementations must pass exhaustive conformance test suite
- Hardware vendors provide optimized OpenVX drivers, architected to get the best performance from their silicon architecture and ready for developers to use

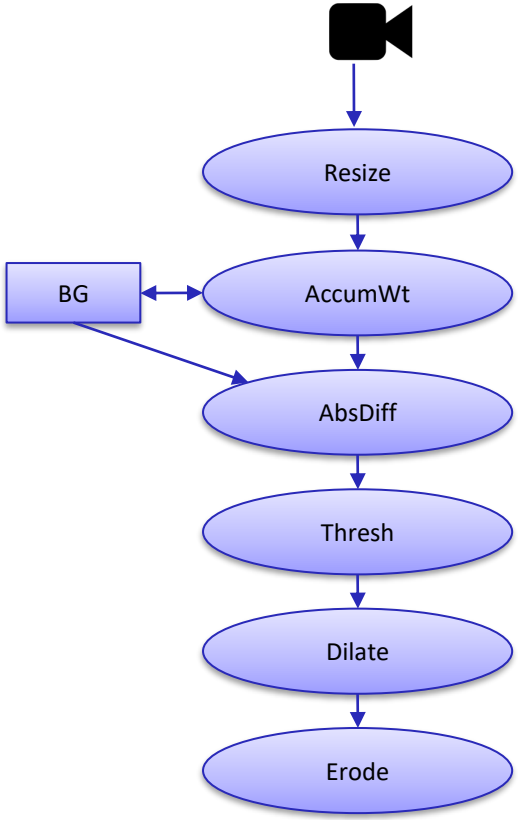
Conformant Implementations of OpenVX from the following vendors

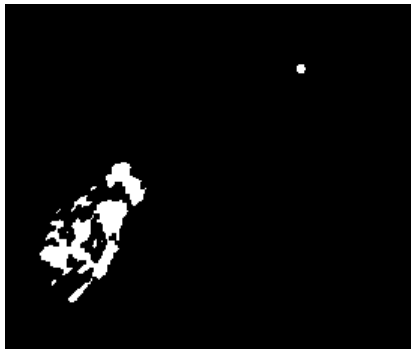


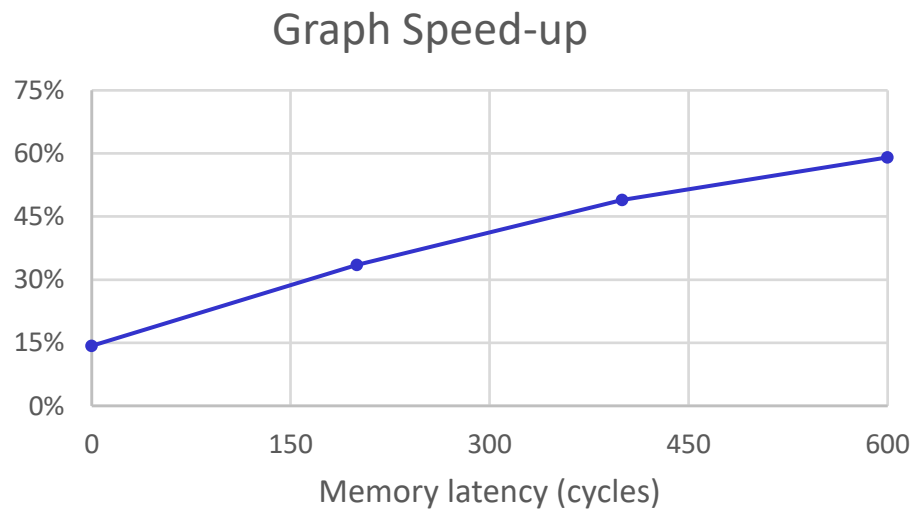

Sample Applications - Case Study

cadence

Application: Background Subtraction for Video Security







Higher memory access penalty → greater graph benefits

KHRONOS GROUP

Sample Applications - Case Study

AXIS COMMUNICATIONS

Application: Reliable Motion Detection

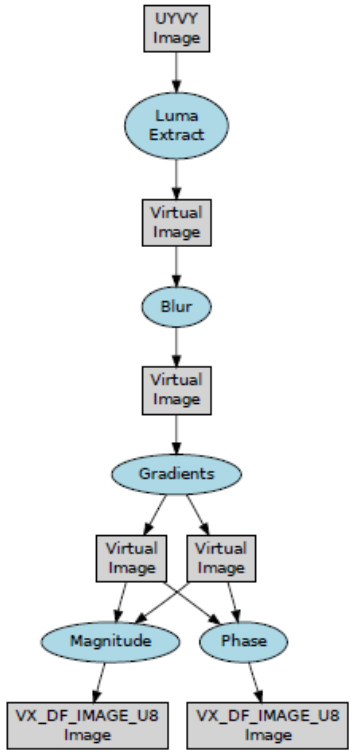
- Uses OpenVX API internally for accelerating algorithm on custom HW blocks
- Compute heavy algorithm for reliable motion detection

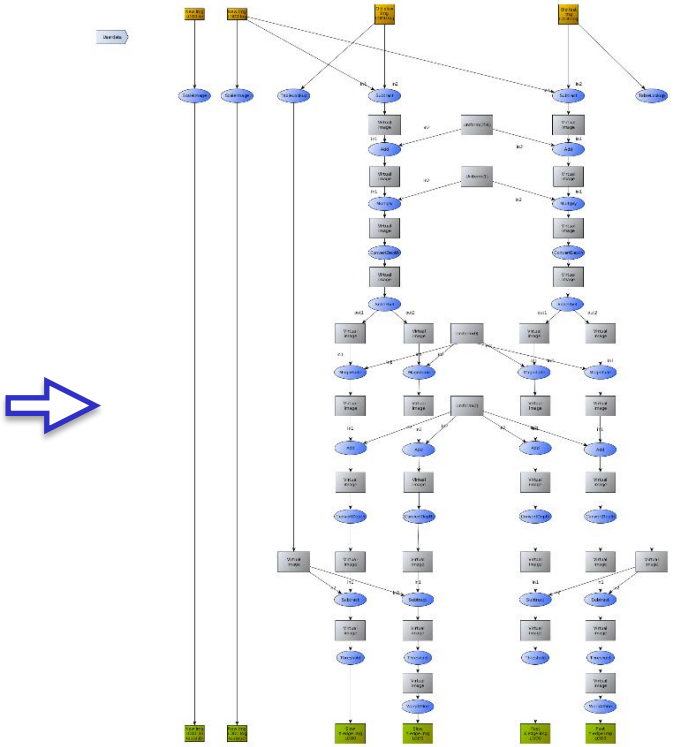
Before OpenVX:

- Hand optimized custom assembler by algorithm developers

After OpenVX:

- Algorithm developers “draw” algorithms as graphs
- Driver developers implement the needed graph API

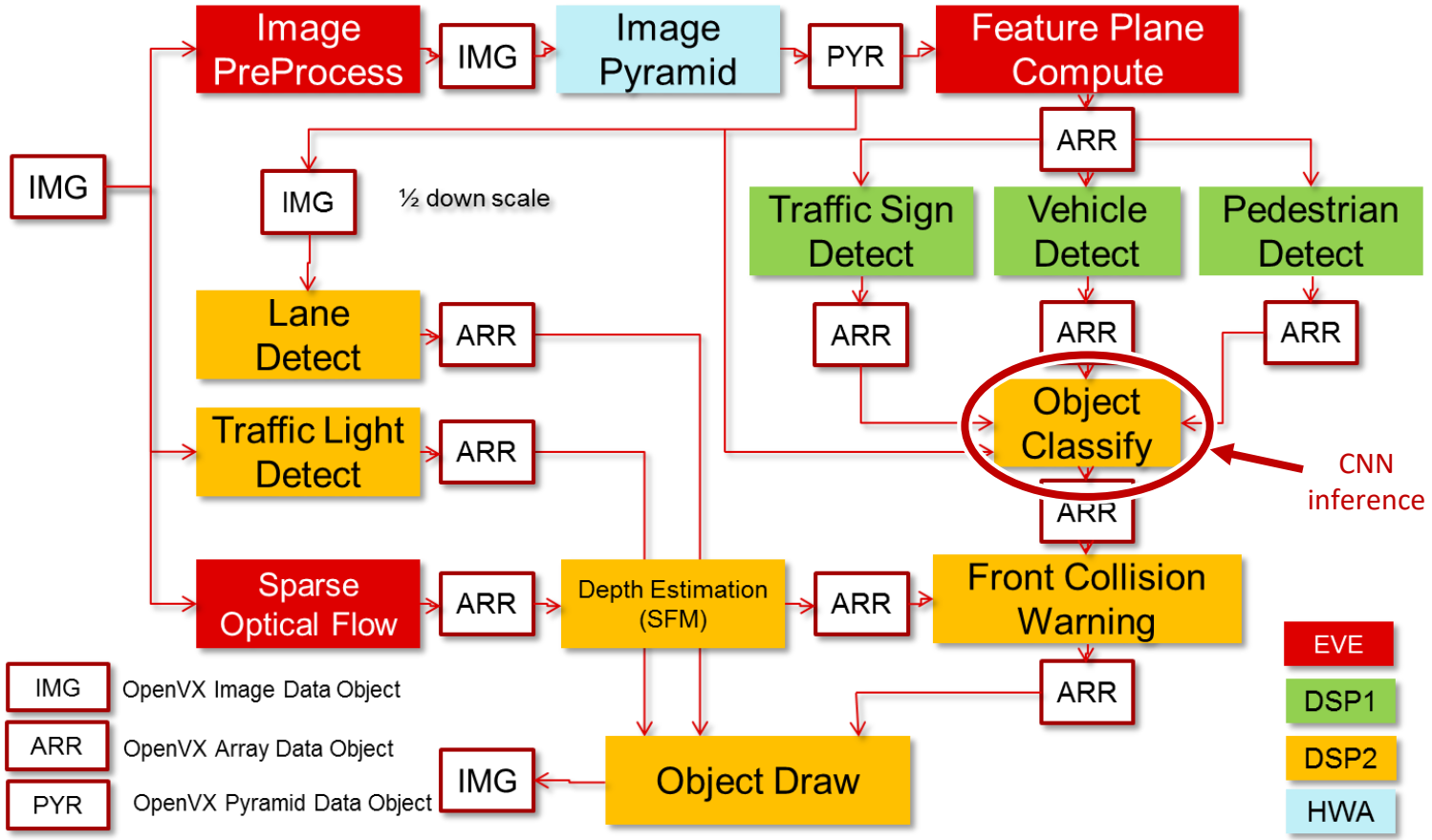




Sample Applications - Case Study

Texas Instruments

Application: Front Camera ADAS Use-case

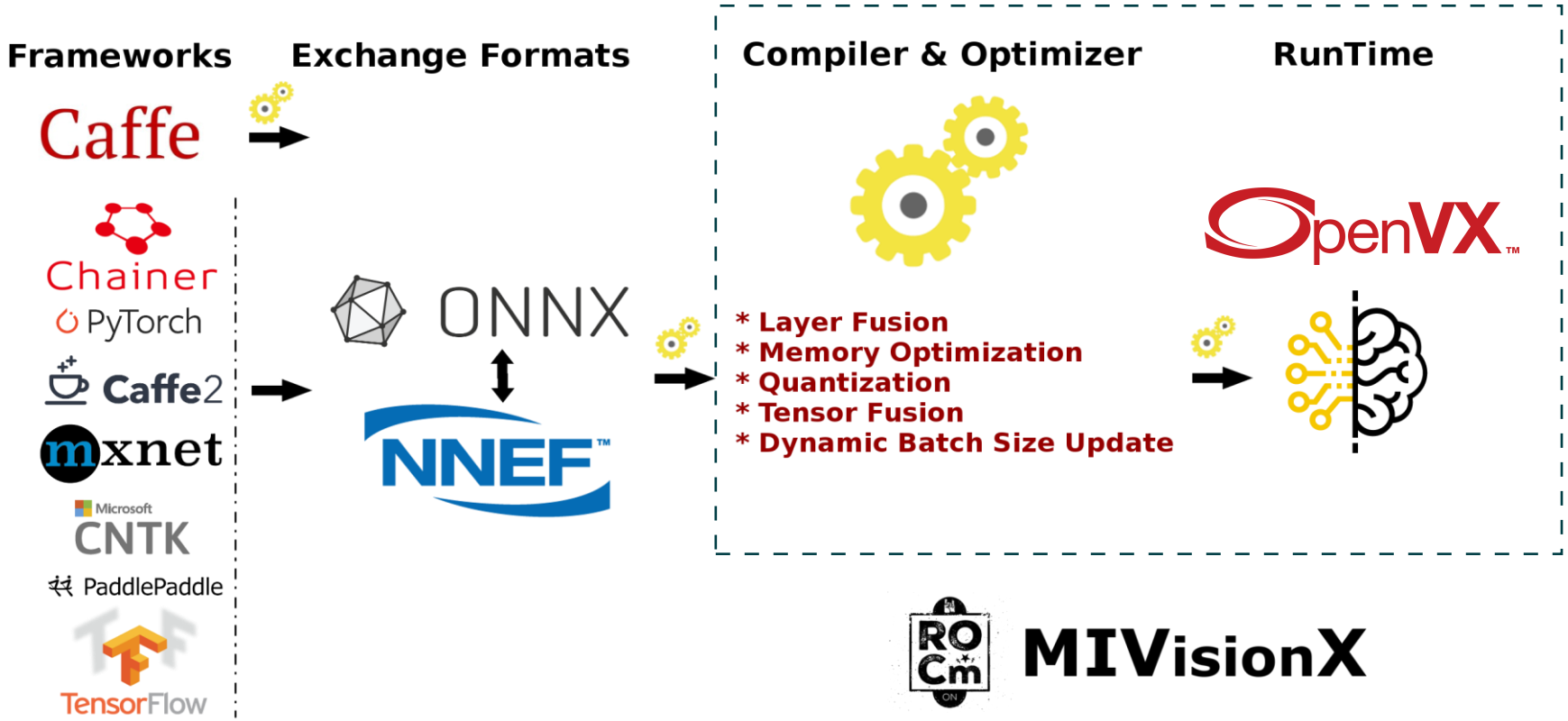


Application using TIOVX with OpenVX Extensions and Custom Extensions from TI

Sample Applications - Case Study

AMD

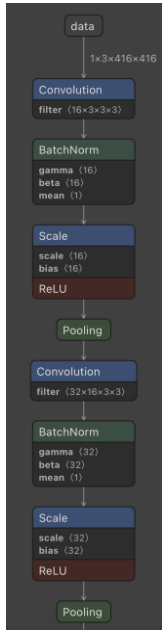
Application: Neural Net Use-case



Sample Applications - Case Study

AMD

Application: Neural Net Use-case



YoloV2

MIVisionX

Model optimization

OpenVX

Graph

Classification
Localization
Annotation

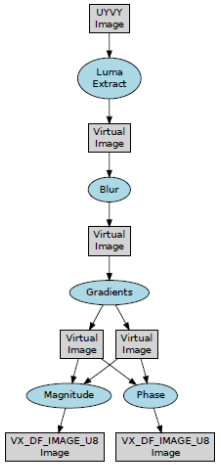
Pre-Processing

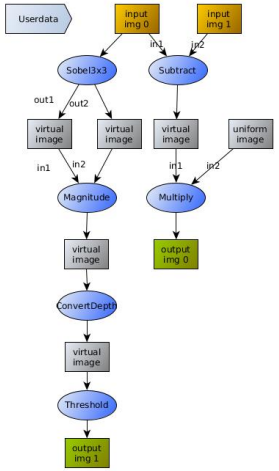
AXIS COMMUNICATIONS

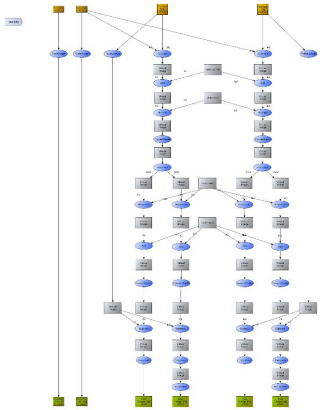
Graph Tool - openvx-graph-parser

Tool for OpenVX C code generation from graphical calculation graph definitions. Includes a Python based parser which takes graphs defined in graphml format and generates valid OpenVX C code.

Open-Sourced on GitHub - https://github.com/AxisCommunications/openvx-graph-parser







```
vx_node function_node;

function_node = vxSubtractNode(graph_skeleton, input_images[0], input_images[1],
                               VX_CONVERT_POLICY_SATURATE, internal_images[1]);
vxReleaseNode(&function_node);

vx_float32 multiply_value;
vx_scalar multiply_scalar;
multiply_value = 0.5;
multiply_scalar = vxCreateScalar(graphmanager_get_context(graph_manager), VX_TYPE_FLOAT32,
                                 &multiply_value);
function_node = vxMultiplyNode(graph_skeleton, internal_images[1], internal_images[0],
                               multiply_scalar, VX_CONVERT_POLICY_SATURATE,
                               VX_ROUND_POLICY_TO_NEAREST_EVEN, output_images[0]);
vxReleaseScalar(&multiply_scalar);
vxReleaseNode(&function_node);

function_node = vxSobel3x3Node(graph_skeleton, input_images[0], internal_images[2],
                               internal_images[3]);
vx_border_t border_mode;
border_mode.mode = VX_BORDER_REPLICATE;
vxSetNodeAttribute(function_node, VX_NODE_BORDER, &border_mode, sizeof(vx_border_t));
vxReleaseNode(&function_node);

function_node = vxMagnitudeNode(graph_skeleton, internal_images[2], internal_images[3],
                                internal_images[4]);
vxReleaseNode(&function_node);

vx_int32 depth_value;
vx_scalar depth_scalar;
depth_value = 0;
depth_scalar = vxCreateScalar(graphmanager_get_context(graph_manager), VX_TYPE_INT32,
                              &depth_value);
function_node = vxConvertDepthNode(graph_skeleton, internal_images[4], internal_images[5],
                                   VX_CONVERT_POLICY_SATURATE, depth_scalar);
vxReleaseScalar(&depth_scalar);
vxReleaseNode(&function_node);

vx_pixel_value_t thresh_value;
vx_threshold thresh;
thresh_value.U8 = 128;
thresh = vxCreateThresholdForImage(graphmanager_get_context(graph_manager),
                                   VX_THRESHOLD_TYPE_BINARY, VX_DF_IMAGE_U8, VX_DF_IMAGE_U8);
vxCopyThresholdValue(thresh, &thresh_value, VX_WRITE_ONLY, VX_MEMORY_TYPE_HOST);
function_node = vxThresholdNode(graph_skeleton, internal_images[5], thresh, output_images[1]);
vxReleaseThreshold(&thresh);
vxReleaseNode(&function_node);
```

KHRONOS GROUP

AMD

Scripting Tool - RunVX

- RunVX is a command-line tool to execute OpenVX graphs
- It encapsulates most of the routine OpenVX calls, thus speeding up development and enabling rapid prototyping.
- As input, RunVX takes a GDF (Graph Description Format) file, a simple and intuitive syntax to describe the various data, nodes, and dependencies.
- The tool has other useful features, such as, file read/write, data compares, image and keypoint data visualization, etc.

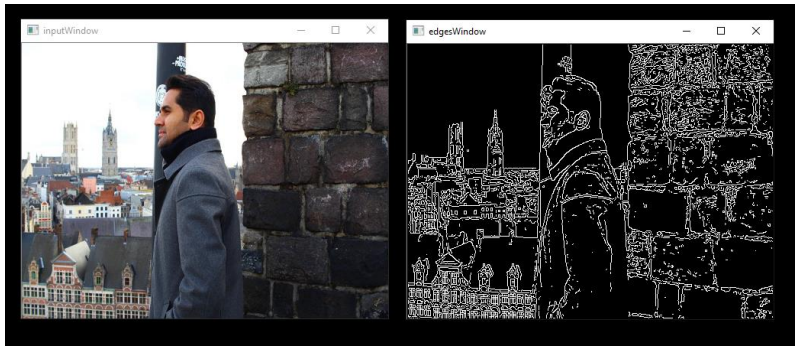
Open-Sourced on GitHub - https://github.com/GPUOpen-ProfessionalCompute-Libraries/MIVisionX

```
# create input and output images
data input  = image:480,360,RGB2
data output = image:480,360,U008

# specify input source for input image and request for displaying input and output images
read input  ../images/face.jpg
view input  inputWindow
view output edgesWindow

# compute luma image channel from input RGB image
data yuv  = image-virtual:0,0,IYUV
data luma = image-virtual:0,0,U008
node org.khronos.openvx.color_convert input yuv
node org.khronos.openvx.channel_extract yuv !CHANNEL_Y luma

# compute edges in luma image using Canny edge detector
data hyst = threshold:RANGE,U008,U008:INIT,80,100
data gradient_size = scalar:INT32,3
node org.khronos.openvx.canny_edge_detector luma hyst gradient_size !NORM_L1 output
```



KHRONOS GROUP

Open-Source Tools

Python Binding - PyVX

pycbindgen is a python based frontend tool for generating python module for a given C library. It uses pycaparser and CFFI python modules underneath.

OpenVX functionality in Python

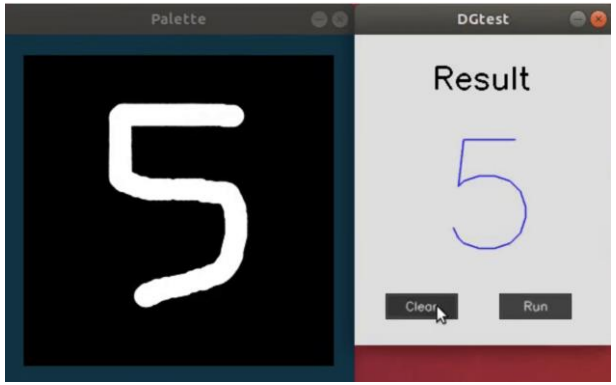
Will be Open-Sourced on GitHub - https://github.com/KhronosGroup

OpenVX for Raspberry Pi - Updates & New Features

- The Khronos Group and the Raspberry Pi Foundation have worked together to implement an open-source implementation of OpenVX™ 1.3, which passes the conformance on Raspberry Pi
- The open-source implementation passes the Vision, Enhanced Vision, Neural Net & NNEF Import Conformance Profiles specified in OpenVX 1.3 on Raspberry Pi
- The Implementation is NEON optimized

Conformant hardware

- Raspberry Pi 3 Model B Rev 1.2
- Raspberry Pi 4 Model B Rev 1.2



OpenVX Camera Capture Extension

- OpenVX relies on external libraries for input
  - Libraries may not be supported across required operating systems
  - Libraries may be too large for embedded systems
  - Libraries have different functionality gaps
- OpenVX can use a simple camera capture API library that works everywhere
  - Linux, Windows, Android, iOS, Raspbian, Embedded OS, no OS
  - initialize()/reinitialize(), capture(), release(), getAttributes(): size, format, etc.
- Once the image is captured, OpenVX will wrap it (not copy it) into an vx_image object

Camera Extension - Future work

Embedded Camera API Exploratory Group Goals

Enable industry dialog to seek consensus on:

IS industry cooperation over camera/sensor/ISP interoperability API(s) beneficial?

And IF so, what API(s) are needed...

...and how and where should the industry organize to create those API(s)?

All companies, universities, consortia, open-source participants welcome!

Explore the creation of open royalty-free API standards for embedded cameras and sensors!

Open to all at no cost!

The right open standard at the right time can be a win-win for all in the industry

https://www.khronos.org/embedded-camera/#getinvolved

Summary

- OpenVX is unique in being the only vision API shipped as an optimized driver
- OpenVX delivers performance comparable to hand-optimized, non-portable code
- Acceleration on a 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!

Acknowledgement

Thanks To

- Mike Schmit - Director of Software Engineering, AMD
- Neil Trevett – President, The Khronos Group
- AMD's MIVisionX Team
- OpenVX Working Group
- Embedded Vision Summit 2021 Organizers

Resource Slide

OpenVX Resources - Khronos

Sample Implementation:

https://github.com/KhronosGroup/OpenVX-sample-impl

Sample Applications:

https://github.com/KhronosGroup/openvx-samples

Tutorial Material:

https://github.com/rgiduthuri/openvx_tutorial

Conformant Implementations

https://www.khronos.org/conformance/adopters/conformant-products/openvx

Khronos OpenVX API Registry

https://www.khronos.org/registry/OpenVX/

OpenVX for Raspberry Pi

https://www.raspberrypi.org/blog/openvx-api-for-raspberry-pi/

AMD ROCm MIVisionX - OpenVX

https://gpuopen-professionalcompute-libraries.github.io/MIVisionX/

Disclaimers and attributions

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED 'AS IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AMD, the AMD Arrow logo, EPYC, Radeon, ROCm and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

KHRONOS GROUP