PyTIOVX: A Code Generation Tool for OpenVX

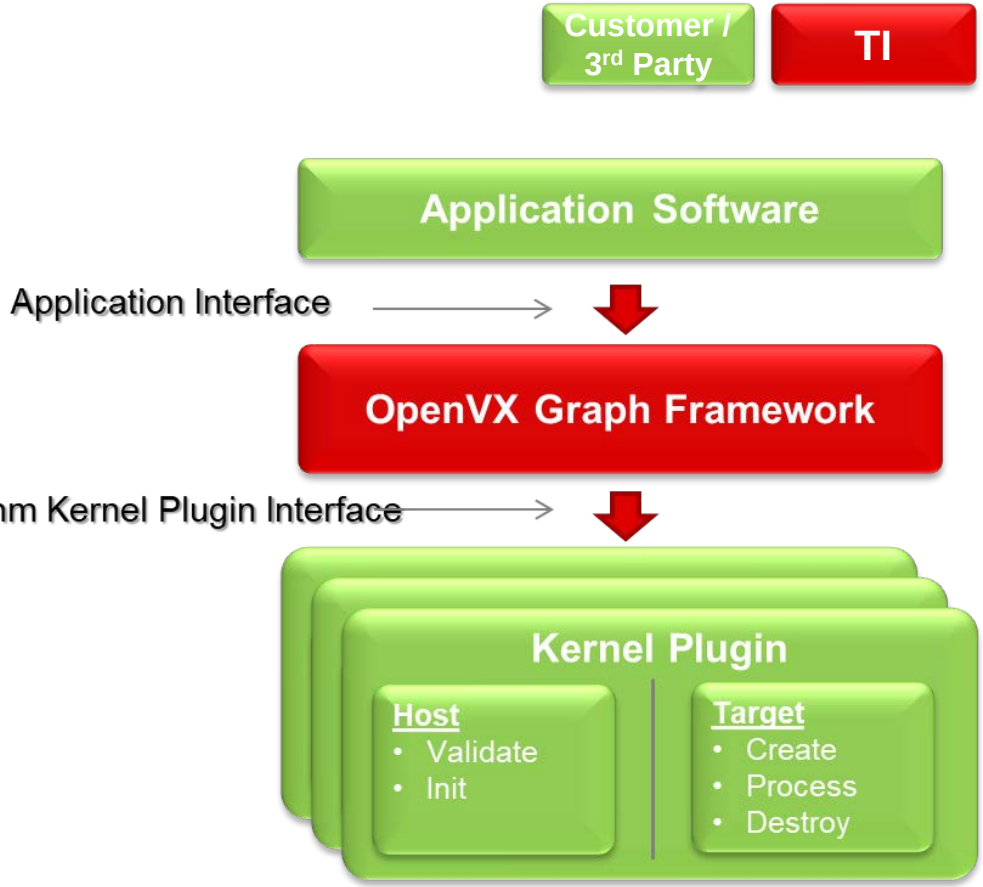
21 November 2019

Lucas Weaver

Automotive Processors, ADAS

Background and Motivation

- OpenVX users must integrate their algorithms into an OpenVX graph
- Legacy VisionSDK Links FW Background
  - Alg plugins used to integrate new algorithm to links framework
  - No existing tool for scripting this process



What is PyTIOVX?

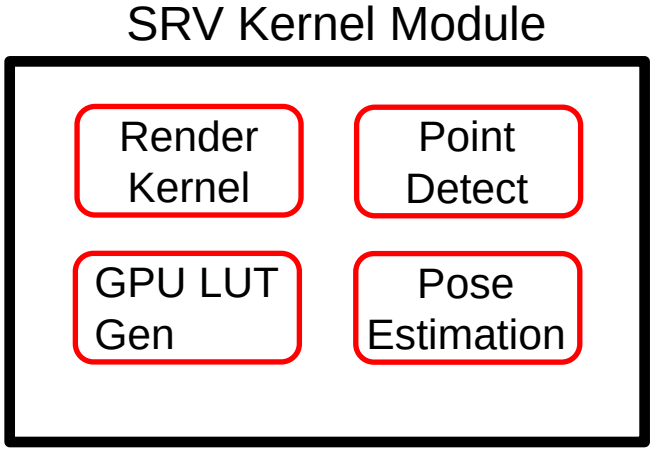
- Simple Python-based Python-based API for generating OpenVX kernel wrappers
- Enables rapid development of custom kernels
- Packaged as a part of the TIOVX project (tiovx/tools/PyTIOVX)
- PSDKRA 7.1 release to include support for pipelined application generation

What benefit does PyTIOVX provide?

- Generates significant amount of OpenVX code per amount of Python code:
  - For example: 9 lines of Python code = 650+ lines of OpenVX code
- Avoids common errors in boilerplate code
- Error checking provided
- “Developer TODO’s” file provided to guide users
- The PyTIOVX generated code details:
  - Code will compile without errors
  - Designed with MISRA-C compliance in mind
- Easy to make API change after the fact
- Concerto files to link into rest of project

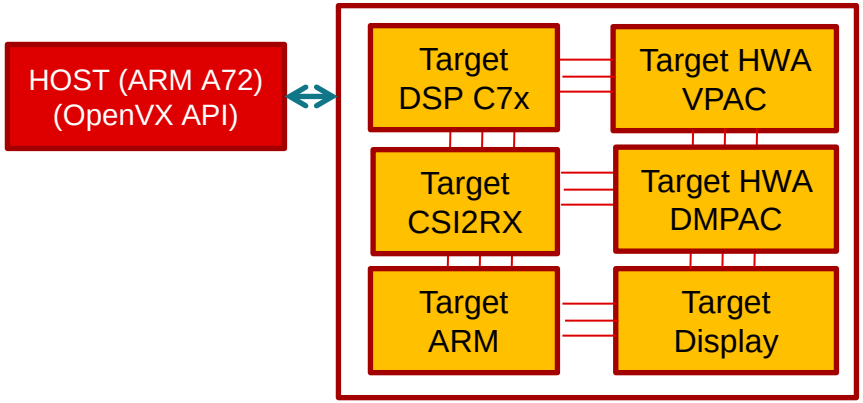
OpenVX Background Concepts

- Kernel: client-defined function
- Node: an instance of a kernel
- Kernel Module: OpenVX specified library of kernels
- Example: SurroundView
  - GPU Kernel for rendering
  - DSP Kernels for calibration and LUT generation
- Kernel Module *must* be loaded into OpenVX context *and* target callbacks registered on intended core in order to run within an OpenVX graph

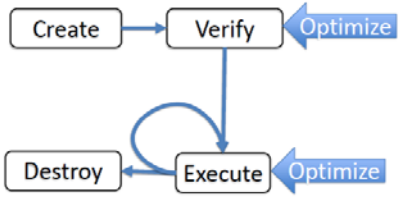


OpenVX Background Concepts

- OpenVX Host
  - Purpose: Set up OpenVX graph
  - Host kernel callbacks:
    - Validation
    - Initialization
- OpenVX Target
  - Purpose: Execute kernel algorithm
  - Target kernel callbacks
    - Create
    - Delete
    - Process
    - Control



OpenVX graph states



Create

- Allocate & initialize object data structures
- Does NOT allocate data buffers.

Verify

- Topological Sort and cycle check.
- Calls "***validate***" kernel callbacks
- Calls "***init***" and "***create***" kernel callbacks
- Reconfigures graph node fusion (BAM)
- Allocate remaining data objects.
- Allocate extra buffers and configure for pipelining

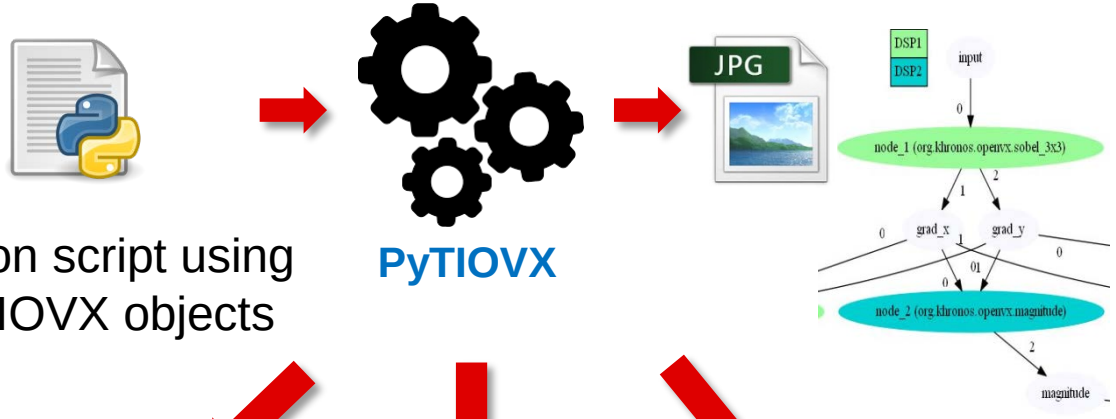
Execute

- Calls "***process***" kernel callbacks of head nodes in the graph
- Nodes trigger dependences directly (calling "***process***" callbacks accordingly)

Destroy

- Calls "***delete***" kernel callbacks
- Frees data buffers
- Deinitialize and free object data structures

What does PyTIOVX provide?



Python script using PyTIOVX objects

PyTIOVX

.C

Host Callbacks

.C

Target Callbacks

.C

Unit Test Plugin

What does the PyTIOVX API provide?

- Specifies user's custom OpenVX kernel API
- Option to create a new kernel module or include within existing kernel module
- Default and custom parameter validation with logging
- Local memory allocation
- Easy to make API change after the fact
  - Regenerate kernel in parallel directory
  - Use a diff tool to compare differences in generated files

```
vx_node VX_API_CALL tivxGlSrvNode(
                    vx_graph              graph,
                    vx_user_data_object   configuration,
                    vx_object_array       input,
                    vx_object_array       srv_views,
                    vx_array              galign_lut,
                    vx_image              output)
```

PyTIOVX API: Create a kernel parameter

```
kernel.setParameter(
 Type.IMAGE,                                    # OpenVX Data Type
 Direction.INPUT,                               # Parameter Direction (INPUT or OUTPUT)
 ParamState.REQUIRED,                           # Required or Optional parameter
 "IN",                                          # Parameter name used in API
 ['VX_DF_IMAGE_U8‘, 'VX_DF_IMAGE_U16’]          # List of possible formats (used in verification)
)

kernel.setParameter(
 Type.USER_DATA_OBJECT,                         # OpenVX Data Type
 Direction.INPUT,                               # Parameter Direction (INPUT or OUTPUT)
 ParamState.REQUIRED,                           # Required or Optional parameter
 "CONFIGURATION",                               # Parameter name used in API
 ['tivx_srv_params_t’]                          # List of possible formats (used in verification)
)
```

PyTIOVX API: Compare parameter attributes

```
kernel.setParameterRelationship(
                ["IN", "OUT"],                  # List of OpenVX objects to compare
                [Attribute.Image.WIDTH,     # List of OpenVX object attributes to use in comparison
                 Attribute.Image.HEIGHT]
                )
```

PyTIOVX API: Allocate local memory

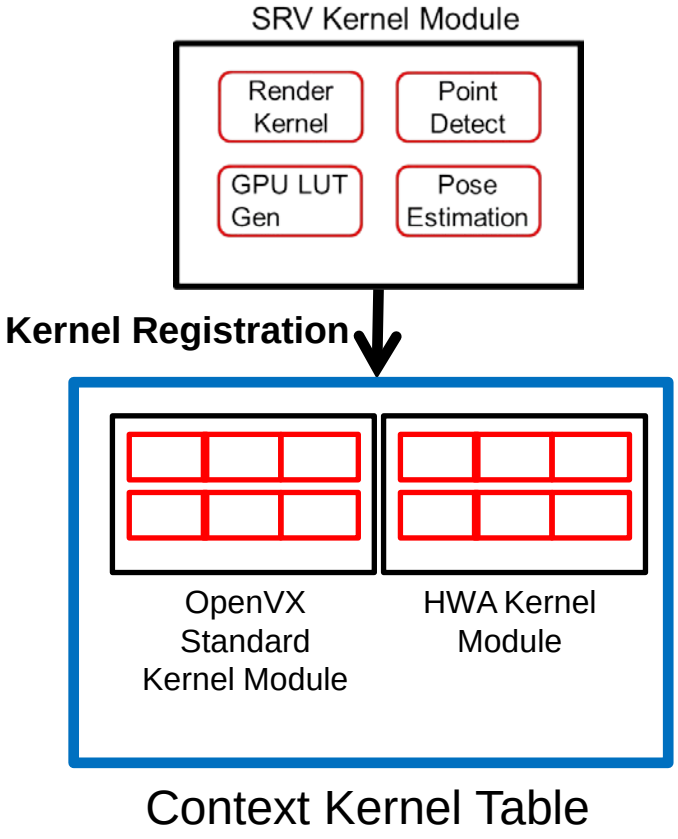
```
kernel.allocateLocalMemory(
        "img_scratch_mem“                                   # Name of pointer to memory
        [Attribute.Image.WIDTH, Attribute.Image.HEIGHT],    # Attributes used for mem allocation size
        "IN_IMAGE“                                          # Name of reference image for allocation size
        )


kernel.allocateLocalMemory(
        "scratch_mem“                                       # Name of pointer to memory
        [“MEM_BUFFER_SIZE”]                                 # Passed as literal to mem allocation call
)
```

What is remaining for kernel developer?

- Developer TODO file
- Host Side Callbacks
  - Custom validation not already provided by PyTIOVX (validate callback)
    - Ex: restriction to certain image resolutions
- Target Side Callbacks
  - Create time initialization on target (create callback)
    - Ex: FVID2_create
  - Algorithm function call (process callback)
    - Ex: Surround view render call

How do I integrate custom OpenVX kernel into application?

- In order for OpenVX node to be used in application, the host and target wrappers must be registered on the appropriate target
- OpenVX context contains a table of all kernels available to application
- Simple interface provided for loading kernels on host and target
  - Host registration: calling during application init
    - Ex: tivxSrvLoadKernels(context)
  - Target registration:
    - Ex: tivxRegisterSrvTargetKernelsC66() (vision_apps/apps/basic_demos/app_tirtos/common/app_init.c)



Surround View Case Study

- Surround view algos already present in legacy VisionSDK
- Defined the API for Surround View kernels on new PSDKRA OpenVX
- Created associated PyTIOVX script
  - vision_apps/kernels/srv/scripts
- Integrated algo calls to generated kernel wrappers
- Created a common test framework for vision apps
- Existing test framework in TIOVX

Summary

- PyTIOVX provides a valuable resource for OpenVX kernel and use case development on TI’s implementation
- Kernel wrapper development using PyTIOVX significantly alleviates customer burden in integrating custom algos to framework
- Automates this process which was manual in Links framework (VisionSDK)
- Future work of creating pipelined applications will further accelerate customer’s development

PyTIOVX – Code Generation Example

```
from tiovx import *

code = KernelExportCode(Module.SRV, Core.A72, "VISION_APPS_PATH")

kernel = Kernel("gl_srv")

kernel.setParameter((Type.USER_DATA_OBJECT, Direction.INPUT, ParamState.REQUIRED, "CONFIGURATION", ['tivx_srv_params_t'])
kernel.setParameter(Type.OBJECT_ARRAY, Direction.INPUT, ParamState.REQUIRED, "INPUT", ['VX_TYPE_NV12'])
kernel.setParameter(Type.OBJECT_ARRAY, Direction.INPUT, ParamState.OPTIONAL, "SRV_VIEWS", ['tivx_srv_coords_t'])
kernel.setParameter(Type.ARRAY, Direction.INPUT, ParamState.OPTIONAL, "GALIGN_LUT")
kernel.setParameter(Type.IMAGE, Direction.OUTPUT, ParamState.REQUIRED, "OUTPUT", ['VX_DF_IMAGE_U8'])

kernel.setTarget(Target.A72_0)

code.export(kernel)
code.exportDiagram(kernel)
```

PyTIOVX – Generated Kernel Host File

- Contains the OpenVX specified callbacks that will run on the host
- Initialize Callback
  - Contains valid region calculation and padding based on kernel parameters
  - Any other initialized that must be performed on the host for this kernel
- Validate Callback
  - Queries attributes of parameters as defined in script
  - Verifies parameter attributes where possible
  - Verifies equality between parameters of which a relationship was set

PyTIOVX – Generated Kernel Target File

- Contains the kernel-specific callbacks that will run on the specified target
- Corresponding callbacks in links framework
  - Create
  - Process
  - Control
  - Delete
- Contains most of the boilerplate code specific to the data types of the kernel
- TI-specific helper functions exist for extracting pointer and attributes from data object descriptor