The OpenVX™ Classifier Extension

The Khronos OpenVX Working Group, Editors: Tomer Schwartz, Radhakrishna Giduthuri

Version 1.2.1 (provisional), Wed, 15 Aug 2018 06:03:09 +0000

Table of Contents

Chapter 1. Classifiers Extension

1.1. Acknowledgements

This specification would not be possible without the contributions from this partial list of the following individuals from the Khronos Working Group and the companies that they represented at the time:

- Radhakrishna Giduthuri - AMD
- Frank Brill - Cadence Design Systems
- Thierry Lepley - Cadence Design Systems
- Tomer Schwartz - Intel
- Victor Eruhimov - Itseez3D
- Jesse Villarreal - Texas Instruments
- Xin Wang - Verisilicon

1.2. Background and Terminology

Classification in computer vision is the process of categorizing an image into a finite set of classes or labels. The process normally involves recognition of the dominant content in an image scene. The dominant content should get the strongest confidence score irrespective of the transformation of that content such as scaling, location or rotation.

In this extension we enable the usage of classification methods on an image as a specific class detector. Possible methods can be cascade, SVM, etc. We do not standardize each of these methods, but rather enable their deployment in a standard way. We add to OpenVX a method to import an abstract model: `vx_classifier_model`. The classifier model can be any kind of classifying technology, and the import API can import any kind of file format. As an example, a vendor can implement in `vxImportClassifierModel` a parser of the OpenCV cascade XML, and create a cascade classification model similar to the one used in OpenCV.

1.3. Kernel Names

When using `vxGetKernelByName` the following are strings specifying the Classifier extension kernel names:

- org.khronos.clasifier_extension.scan_classifier

Chapter 2. Module Documentation

2.1. Group_object_classifier_model

An Opaque object that contain a classifier model. The model can be cascade model or SVM model or any other machine learning model.

The Object is created by importing data from a binary format. The specification will not define such a format. Extensions to the specification will be added in order to define such binary formats.

Typedefs

- vx_classifier_model

Enumerations

- vx_classifier_model_format_e
- vx_classifier_type_e

Functions

- vxImportClassifierModel
- vxReleaseClassifierModel

2.1.1. Typedefs

vx_classifier_model

classification model to be used in vxScanClassifierNode. The classification models are loadable by undefined binary format see vxImportClassifierModel. Extensions will be added to the specification, to support a defined binary format.

```
typedef struct _vx_classifier_model *vx_classifier_model;
```

2.1.2. Enumerations

vx_classifier_model_format_e

Classifier model format enums. In the main specification only undefined binary format is supported. Extensions to the specification will be added in order to support specific binary format.

```
enum vx_classifier_model_format_e {
    VX_CLASSIFIER_MODEL_UNDEFINED = VX_ENUM_BASE( VX_ID_KHRONOS,
VX_ENUM_CLASSIFIER_MODEL ) + 0x0,
};
```

Enumerator

- VX_CLASSIFIER_MODEL_UNDEFINED - Undefined binary format. Using this enumeration will result in an implementation defined behaviour.

vx_classifier_type_e

The type enumeration lists all classifier extension types.

```
enum vx_classifier_type_e {
    VX_TYPE_CLASSIFIER_MODEL = 0x02C,
};
```

Enumerator

- VX_TYPE_CLASSIFER_MODEL - A vx_classifier_model type.

2.1.3. Functions

vxImportClassifierModel

Creates an opaque reference classifier model This function creates a classifier model to be used in vxScanClassifierNode. The object classifier object is a read-only constant object. It cannot be changed during graph execution.

```
vx_classifier_model vxImportClassifierModel(
   vx_context                                  context,
   vx_enum                                     format,
   const vx_uint8*                             ptr,
   vx_size                                     length);
```

Parameters

- [in] *context* - Reference to the context where to create the ClassifierModel.
- [in] *format* - The binary format which contain the classifier model. See vx_classifier_model_format_e. Currently only undefined binary format is supported. Extensions will be added to the specification, to support a classification model defined binary format.
- [in] *ptr* - A memory pointer to the binary format.
- [in] *length* - size in bytes of binary format data.

Returns: A ClassifierModel reference vx_classifier_model. Any possible errors preventing a successful creation should be checked using vxGetStatus.

vxReleaseClassifierModel

Releases a reference of an ClassifierModel object. The object may not be garbage collected until its total reference and its contained objects count is zero. After returning from this function the reference is zeroed/cleared.

```
vx_status vxReleaseClassifierModel(
    vx_classifier_model*                        model);
```

Parameters

- [in] *model* - The pointer to the ClassifierModel to release.

Returns: A vx_status_e enumeration.

Return Values

- <tt> -

2.2. Scan Classifier

Scans a feature-map (input_feature_map) and do the classification for each scan-window.

This function scans a feature-map. Each window in the feature map is classified by a classification model. The classification models are loadable by undefined binary format see vxImportClassifierModel. Extensions will be added to the specification, to support a defined binary format. Classification models can be any machine learning classification method. Examples are Cascade, SVM, and Neural Networks.

Functions

- vxScanClassifierNode

2.2.1. Functions

vxScanClassifierNode

[Graph] Scans a feature-map (input_feature_map) and detect the classification for each scan-window.

```
vx_node vxScanClassifierNode(
    vx_graph                                    graph,
    vx_tensor                                   input_feature_map,
    vx_classifier_model                         model,
    vx_int32                                    scanwindow_width,
    vx_int32                                    scanwindow_height,
    vx_int32                                    step_x,
    vx_int32                                    step_y,
    vx_array                                    object_confidences,
    vx_array                                    object_rectangles,
    vx_scalar                                   num_objects);
```

Parameters

- [in] *graph* - The reference to the graph
- [in] *input_feature_map* - The Feature-map, example is the output of vxHOGFeaturesNode.
- [in] *model* - The pre-trained model loaded. Loaded using vxImportClassifierModel
- [in] *scan_window_width* - Width of the scan window
- [in] *scan_window_height* - Height of the scan window
- [in] *step_x* - Horizontal step-size (along x-axis)
- [in] *step_y* - Vertical step-size (along y-axis)
- [out] *object_confidences* - [Optional] An array of confidences measure, the measure is of type VX_TYPE_UINT16. The confidence measure is defined by the extensions which define classification model with defined binary format. This output can be used as class index as well. In case we detect several different classes in single execution. The output will be an array of indexes of the classes.
- [out] *object_rectangles* - An array of object positions, in VX_TYPE_RECTANGLE
- [out] *num_objects* - [optional] The number of object detected in a VX_SIZE scalar

Note

The border mode VX_NODE_BORDER value VX_BORDER_UNDEFINED is supported.

Returns: vx_node.

Return Values

- vx_node - A node reference. Any possible errors preventing a successful creation should be checked using vxGetStatus