



The **OpenVX™** Classifier Extension

Version 1.2

Document Revision: 9249ea0

Generated on Mon Jul 10 2017 15:49:36

Khronos Vision Working Group

Editor: Schwartz Tomer

Copyright ©2017 The Khronos Group Inc.

Copyright ©2017 The Khronos Group Inc. All Rights Reserved.

This specification is protected by copyright laws and contains material proprietary to the Khronos Group, Inc. It or any components may not be reproduced, republished, distributed, transmitted, displayed, broadcast, or otherwise exploited in any manner without the express prior written permission of the Khronos Group. You may use this specification for implementing the functionality therein, without altering or removing any trademark, copyright or other notice from the specification, but the receipt or possession of this specification does not convey any rights to reproduce, disclose, or distribute its contents, or to manufacture, use, or sell anything that it may describe, in whole or in part. Khronos Group grants express permission to any current Promoter, Contributor or Adopter member of Khronos to copy and redistribute UNMODIFIED versions of this specification in any fashion, provided that NO CHARGE is made for the specification and the latest available update of the specification for any version of the API is used whenever possible. Such distributed specification may be reformatted AS LONG AS the contents of the specification are not changed in any way. The specification may be incorporated into a product that is sold as long as such product includes significant independent work developed by the seller. A link to the current version of this specification on the Khronos Group website should be included whenever possible with specification distributions.

Khronos Group makes no, and expressly disclaims any, representations or warranties, express or implied, regarding this specification, including, without limitation, any implied warranties of merchantability or fitness for a particular purpose or non-infringement of any intellectual property. Khronos Group makes no, and expressly disclaims any, warranties, express or implied, regarding the correctness, accuracy, completeness, timeliness, and reliability of the specification. Under no circumstances will the Khronos Group, or any of its Promoters, Contributors or Members or their respective partners, officers, directors, employees, agents or representatives be liable for any damages, whether direct, indirect, special or consequential damages for lost revenues, lost profits, or otherwise, arising from or in connection with these materials. SAMPLE CODE and EXAMPLES, as identified herein, are expressly depicted herein with a "grey" watermark and are included for illustrative purposes only and are expressly outside of the Scope as defined in Attachment A - Khronos Group Intellectual Property (IP) Rights Policy of the Khronos Group Membership Agreement. A Member or Promoter Member shall have no obligation to grant any licenses under any Necessary Patent Claims covering SAMPLE CODE and EXAMPLES.

Contents

- 1 Classifiers Extension** **2**
- 1.1 Acknowledgements 2
- 1.2 Background and Terminology 2
- 1.3 Kernel names 2

- 2 Module Documentation** **3**
- 2.1 Group_object_classifier_model 3
- 2.1.1 Detailed Description 3
- 2.1.2 Enumeration Type Documentation 3
- vx_classifier_model_format_e 3
- 2.1.3 Function Documentation 5
- vxImportClassifierModel() 5
- vxReleaseClassifierModel() 5
- 2.2 Scan Classifier 6
- 2.2.1 Detailed Description 6
- 2.2.2 Function Documentation 6
- vxScanClassifierNode() 6

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:

- Frank Brill - Cadence Design Systems
- Tomer Schwartz - Intel
- Thierry Lepley - Cadence Design Systems
- Radha Giduthuri - AMD
- Jesse Villarreal - TI
- Victor Eruhimov - Itseez3D
- 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.

Typedefs

- typedef struct _vx_classifier_model * [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.

Enumerations

- enum [vx_classifier_model_format_e](#) { [VX_CLASSIFIER_MODEL_UNDEFINED](#) = ((([VX_ID_KHRONOS](#)) << 20) | ([VX_ENUM_CLASSIFIER_MODEL](#) << 12)) + 0x0 }

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.

Functions

- [vx_classifier_model](#) [vxImportClassifierModel](#) (vx_context context, vx_enum format, vx_uint8 *ptr, vx_size length)

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_status [vxReleaseClassifierModel](#) ([vx_classifier_model](#) *model)

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.

2.1.1 Detailed Description

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.

2.1.2 Enumeration Type Documentation

[vx_classifier_model_format_e](#)

enum [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.

Enumerator

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

Definition at line 80 of file [vx_khr_class.h](#).

2.1.3 Function Documentation

vxImportClassifierModel()

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

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.

Parameters

in	<i>context</i>	Reference to the context where to create the ClassifierModel.
in	<i>format</i>	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	<i>ptr</i>	A memory pointer to the binary format.
in	<i>length</i>	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()

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

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.

Parameters

in	<i>model</i>	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.

Functions

- 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)

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

2.2.1 Detailed Description

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.

2.2.2 Function Documentation

vxScanClassifierNode()

```
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 )
```

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

Parameters

in	<i>graph</i>	The reference to the graph
in	<i>input.feature_map</i>	The Feature-map, example is the output of vxHOGFeaturesNode .
in	<i>model</i>	The pre-trained model loaded. Loaded using vxImportClassifierModel
in	<i>scan_window_width</i>	Width of the scan window
in	<i>scan_window_height</i>	Height of the scan window
in	<i>step_x</i>	Horizontal step-size (along x-axis)
in	<i>step_y</i>	Vertical step-size (along y-axis)
out	<i>object_confidences</i>	[Optional] An array of confidences measure, the measure is of type <code>VX_TYPE_UINT16</code> . 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	<i>object_rectangles</i>	An array of object positions, in <code>VX_TYPE_RECTANGLE</code>
out	<i>num_objects</i>	[optional] The number of object detected in a <code>VX_SIZE</code> scalar

Note

The border mode `VX_NODE_BORDER` value `VX_BORDER_UNDEFINED` is supported.

Returns

`vx_node`.

Return values

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

Index

- Group_object_classifier_model, [3](#)
 - vx_classifier_model_format_e, [3](#)
 - vxImportClassifierModel, [5](#)
 - vxReleaseClassifierModel, [5](#)

- Scan Classifier, [6](#)
 - vxScanClassifierNode, [6](#)

- vx_classifier_model_format_e
 - Group_object_classifier_model, [3](#)
- vxImportClassifierModel
 - Group_object_classifier_model, [5](#)
- vxReleaseClassifierModel
 - Group_object_classifier_model, [5](#)
- vxScanClassifierNode
 - Scan Classifier, [6](#)