OpenVX is an open, cross-platform, low-level computer vision library for real-time computer vision applications. It offers high performance, portability, and easy-to-use APIs to access hardware acceleration capabilities.

**OpenVX 1.0 API Reference Guide Page 1**

### Graph and Immediate Mode Vision Functions

#### Graph Mode

Graph nodes may be created and linked together, verified by the implementation, and executed as often as needed.

**Immediate Mode**

Immediate mode functions are executed on a context immediately, as if they were single node graphs, with no leaking side-effects.

#### Graph Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vx_node_vtxNode</td>
<td>A graph node is supported.</td>
</tr>
<tr>
<td>vx_graph_vtxGraph, T arg1, ...</td>
<td>The reference to the graph context.</td>
</tr>
</tbody>
</table>

In the functions below, replace `vx` with `vtx` and `vx_node` with `vtx_node` as appropriate.

### Fast Corners

- **vxuFastCorners**
  - Computes a Box filter over a window of the input image.
  - **vx_node_vtxFastCorners**
    - `vx_node` of type `VX_TYPE_KEYPOINT`.
    - `vx_image` input.
    - `vx_image` output.

### Canny Edge Detector

- **vxuCannyEdgeDetector**
  - Provides a Canny edge detection.
  - **vx_node_vtxCannyEdgeDetector**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` output.
    - `vx_image` threshold.
    - `vx_image` gradient size.
    - `vx_image` norm type.

### Channel Combine

- **vxuChannelCombine**
  - Combines multiple images.
  - **vx_node_vtxChannelCombine**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` plane.
    - `vx_image` output.

### Color Convert

- **vxuColorConvert**
  - Converts the format of an image.
  - **vx_node_vtxColorConvert**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` output.

### Pixel Operations

#### Arithmetic Addition

- **vxuAdd**
  - Adds two images.
  - **vx_node_vtxAdd**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` output.

#### Arithmetic Subtraction

- **vxuSubtract**
  - Subtracts one image from another.
  - **vx_node_vtxSubtract**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` output.

#### Bitwise Operations

- **vxuXor**
  - Performs bitwise XOR on two images.
  - **vx_node_vtxXor**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` output.

### Histogram

- **vxuHistogram**
  - Generates a distribution from an image.
  - **vx_node_vtxHistogram**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_distribution` distribution.

### Gaussian Filter

- **vxuGaussianFilter**
  - Computes a Gaussian filter over a window of the input image.
  - **vx_node_vtxGaussianFilter**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` output.

### Harris Corners

- **vxuHarrisCorners**
  - Computes the Harris Corners of an image.
  - **vx_node_vtxHarrisCorners**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` output.
    - `vx_image` mask.
    - `vx_image` output.

### Erode Image

- **vxuErodeImage**
  - Shrinks the white space in an image.
  - **vx_node_vtxErodeImage**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` output.

### Dilate Image

- **vxuDilateImage**
  - Grows the white space in an image.
  - **vx_node_vtxDilateImage**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` output.

### Equalize Histogram

- **vxuEqualizeHist**
  - Normalizes brightness and contrast of an image.
  - **vx_node_vtxEqualizeHist**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` output.

### Magnitude

- **vxuMAG**
  - Computes the magnitude of an image.
  - **vx_node_vtxMAG**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` output.

### Mean and Standard Deviation

- **vxuMeanStdDev**
  - Computes the mean and standard deviation of an image.
  - **vx_node_vtxMeanStdDev**
    - `vx_node` of type `VX_TYPE_IMAGE`.
    - `vx_image` input.
    - `vx_image` output.
    - `vx_image` mean.
    - `vx_image` stddev.
    - `vx_image` output.

---

©2015 Khronos Group - Rev. 0115

www.khronos.org/openvx
objects

Object: Array [3.48]

Grant access to a sub-range of an Array.

+ vx_status vxvStatusArrayRange (vx_array array, vx_size start, vx_size end, vx_size *stride, void **ptr, vx_enum usage)

Add items to the Array.

+ vx_status vxvAddArrayItems (vx_array array, vx_size size, count, void **ptr, vx_size stride)

Commit data back to the Array object.

+ vx_status vxvCommitArrayRange (vx_array array, vx_size start, vx_size end, void *ptr)

Create a reference to an Array object.

+ vx_array vxvCreateArray (vx_context context, vx_enum item_type, vx_size size, void *attribute)

Create opaque reference to virtual Array; no direct user access.

+ vx_array vxvCreateVirtualArray (vx_array graph, vx_enum item_type, vx_size size)

Query the Array for some specific information.

+ vx_status vxvQueryContext (vx_array array, void *attribute, void *ptr, vx_size size)

Release a reference to an Array object.

+ vx_status vxvReleaseArray (vx_array *arr)

Truncate an Array (remove items from the end).

+ vx_status vxvTruncateArray (vx_array array, vx_size new_num_items)

Object: Area [3.49]

Create a context.

+ vx_context vxvCreateContext ()

Retrieve the context from any reference from within a context.

+ vx_context vxvGetContext (vx_reference reference)

Query the context for some specific information.

+ vx_status vxvQueryContext (vx_context context, vx_enum attribute, void *ptr, vx_size size)

Release the OpenVX object context.

+ vx_status vxvReleaseContext (vx_context context)

Set an attribute on the context.

+ vx_status vxvSetContextAttribute (vx_context context, vx_enum attribute, void *ptr, vx_size size)

Object: Array (Advanced) [3.60]

Registers user-defined structures to the context.

+ vx_enum vxvRegisterUserStruct (vx_context context, vx_size size)

Object: Convolution [3.49]

Get the convolution data (copy).

+ vx_status vxvAccessConvolutionCoefficients (vx_convolution xconv, vx_size new_items, void *attribute)

Set the convolution data (copy).

+ vx_status vxvCommitConvolutionCoefficients (vx_convolution conv, vx_size new_items, void *attribute)

Create a reference to a convolution matrix object.

+ vx_convolution vxvCreateConvolution (vx_context context, vx_size new_samples, void *attribute)

Queries an attribute on the convolution matrix object.

+ vx_status vxvQueryConvolution (vx_convolution conv, vx_enum attribute, void *ptr, vx_size size)

Release a reference to a convolution matrix.

+ vx_status vxvReleaseConvolution (vx_convolution *conv)

Set attributes on the convolution object.

+ vx_status vxvSetConvolutionAttribute (vx_convolution conv, vx_enum attribute, void *ptr, vx_size size)

attribute: The attribute to query or modify.

+ columns, rows: The (columns, rows) dimension of the convolution.

+ conv: The pointer to the convolution matrix object to release; or convolution matrix object to set; or the coordinates object to set.

+ size: The size of the container or data to which ptr points.

+ array: The array to place or read the convolution.

+ ptr: The location at which to store the resulting value; or pointer to the value to which set the attribute.

Object: Delay (Advanced) [3.63]

Age the internal delay ring by one.

+ vx_status vxvAgeDelay (vx_delay_delay)

Create a Delay object.

+ vx_status vxvCreateDelay (vx_context context, vx_reference example, vx_size size)

Retrieve a reference from a delay object.

+ vx_reference vxvGetReferenceFromDelay (vx_delay delay, vx_int32 index)

Query a vx_delay object attribute.

+ vx_status vxvQueryDelay (vx_delay delay, vx_enum attribute, void *ptr, vx_size size)

Release a reference to a delay object.

+ vx_status vxvReleaseDelay (vx_delay_delay, vx_size size, void *attribute)

Object: Distribution [3.50]

Get direct access to a Distribution in memory.

+ vx_status vxvAccessDistribution (vx_distrution_distribution, void **ptr, vx_enum usage)

Set the Distribution back to the memory.

+ vx_status vxvCommitDistribution (vx_distrution_distribution, void *ptr)

Create a reference to a 1D Distribution.

+ vx_distribution vxvCreateDistribution (vx_context context, vx_size size, vx_size numBins, vx_size offset, vx_size size_range)

attribute: The attribute to query from vxv_delay_delay, vxv_size size; or from vxv_distrution_distribution, void **ptr, vxv_enum usage).

+ ptr: The location at which to store the resulting value; or pointer to the value to which set the attribute.
Objects (cont.)

Query a Distribution object.

vx_status vxQueryDistribution (vx_distribution distribution, vx_uint32 attribute, void *ptr, vx_size size);

Release a reference to a distribution object.

vx_status vxReleaseDistribution (vx_distribution distribution);

On each attribute, the reference to query, from vx_distribution_attribute_e.

color: The reference to the overall context.

dimensions: The reference to the implementation context.

Object: Graph

Create a reference to an image object that has a singular, vx_image

Create an image from another image given a rectangle.

vx_status vxCreateImage (vx_image img, vx_rectangle_t rect, vx_df_image df);

Compute the size needed to retrieve an image patch.

int: Create a reference to an image object that has a singular, vx_image

Create an image from a single plane.

vx_status vxCreateImageFromHandle (vx_context context, vx_uint32 width, vx_uint32 height, vx_df_image df_color);

Create a reference to an image object that has a singular, vx_image

Create an image from another image given a rectangle.

vx_status vxCreateImageFromROI (vx_image img, vx_rectangle_t rect);

Create a reference to an externally allocated image object.

vx_status vxCreateImageFromHandle (vx_context context, vx_uint32 width, vx_uint32 height, vx_df_image df_color);

Create a reference to an image object that has a singular, vx_image

Create a reference to a uniform image in all pixels.

vx_status vxCreateUniformImage (vx_context context, vx_uint32 width, vx_uint32 height, vx_df_image df_color, void *value);

Create opaque reference to image buffer with no direct user access.

vx_status vxQueryImageObject (vx_image image, vx_uint32 attribute, void *ptr, vx_size size);

access: The reference to the implementation context.

data_type: The type of data stored in the LUT.

luts: The LUT to query or modify; or the pointer to the LUT to release.

data: The size of the container to which ptr points.

usage: Declare the intended usage of the pointer. From vx_accessor_e.

ptr: The location at which to store the resulting value.

access: The reference to the implementation context.

data_type: The type of data stored in the LUT.

luts: The LUT to query or modify; or the pointer to the LUT to release.

data: The size of the container to which ptr points.

usage: Declare the intended usage of the pointer. From vx_accessor_e.

ptr: The location at which to store the resulting value.

Object: Matrix

[3.53]

{[Set, Get] the matrix data (copy).

vx_status vxAccessMatrix (vx_matrix mat, void *array);

Create a reference to a matrix object.

vx_status vxCreateMatrix (vx_context context, vx_matrix_type_t type, vx_size_columns, vx_size_rows);

Query an attribute on the matrix object.

vx_status vxQueryMatrix (vx_matrix mat, vx_enum_attribute attribute, void *ptr, vx_size size);

Release a reference to a matrix object.

vx_status vxReleaseMatrix (vx_matrix *mat);

array: Array to read the matrix, or in which to place the matrix.

attribute: The reference to the attribute value.

color: The reference to the overall context.

Object: Node

[3.47]

Query a parameter to determine its meta-information.

vx_status vxQueryParameter (vx_node node, vx_size count, vx_parameter *value);

Set attributes of a node before Graph Validation.

vx_status vxSetNodeAttribute (vx_node node, vx_enum_attribute attribute, void *ptr, vx_size size);

related nodes: The reference to the node to query, to the node to set, or the pointer to the reference of the node to release.

data: The size of the container to which ptr points.

ptr: The location at which to store the resulting value.

Object: Node (Advanced)

[3.61]

Defines the advanced features of the Node Interface.

vx_node vxCreateGenericNode (vx_graph graph, vx_kernel kernel);

graph: The reference to the graph in which this node exists.

kernel: The kernel reference to associate with this new node.

Object: Parameter (Advanced)

[3.65]

Retrieve a vx parameter from a vx kernel.

vx_status vxGetKernelParameterByIndex (vx_kernel kernel, vx_uint32 index);

Retrieve a vx parameter from a vx node.

vx_status vxGetParameterByIndex (vx_node node, vx_uint32 index);

Query a parameter to determine its meta-information.

vx_status vxQueryParameter (vx_node node, vx_parameter *value);

Set the specified parameter data for a kernel on the node.

vx_status vxSetParameterByIndex (vx_node node, vx_uint32 index, vx_reference value);

Associate a parameter reference and a data reference with a kernel on a node.

vx_status vxSetParameterByReference (vx_parameter parameter, vx_reference value);

attribute: The attribute to query from vx_parameter_e.

index: The index of the parameter.

kernel: The reference to the kernel.

data: The node that contains the kernel.

param: The pointer to the parameter.

(Continued on next page)
Object: Pyramid [3.54]

Create a reference to a pyramid object.

\[ \text{vx_pyramid}(\text{vx_graph}, \text{pyr}, \text{context}) \]

- `vx_pyramid`: The pointer to the object to hint at.
- `vx_graph`: The overall context.
- `pyr`: The string to use to match the kernel.
- `context`: The reference to the overall context.

The type of threshold to create.

\[ \text{vx_threshold}(\text{vx_threshold \_type}, \text{vx \_enum \_data \_type}) \]

- `vx_threshold`: The reference from which to get the scalar.
- `vx \_threshold \_type`: The type of the threshold.
- `vx \_enum \_data \_type`: The data type of the threshold(s).

- `size`: The size of the data pointed to by `ptr`.
- `thresh`: Threshold object to set or query, or a pointer to a threshold object to release.

- `context`: The reference to the overall context.
- `dir`: Direction of the parameter, from `vx_direction_e`.
- `data_type`: Type of parameter, from `vx_type_e`.
- `numParams`: The number of parameters for this kernel.
- `validates`: The {input, output} parameters to this kernel.
- `state`: State of the parameter, from `vx_parameter \_state_e`.
- `name`: The short name of the module to load.
- `module`: The short name of the module to load.

- `numParams`: The number of parameters for this kernel.
- `parameter`: The pointer to the value to which the qualifier is set.

- `ref`: The reference to the kernel.
- `kernel`: The reference to the overall context.
- `dir`: Direction of the parameter, from `vx_direction_e`.
- `data_type`: Type of parameter, from `vx_type_e`.
- `numParams`: The number of parameters for this kernel.
- `validates`: The {input, output} parameters to this kernel.
- `state`: State of the parameter, from `vx_parameter \_state_e`.
- `name`: The short name of the module to load.
- `module`: The short name of the module to load.

- `ref`: The reference to the kernel.
- `kernel`: The reference to the overall context.
- `dir`: Direction of the parameter, from `vx_direction_e`.
- `data_type`: Type of parameter, from `vx_type_e`.
- `numParams`: The number of parameters for this kernel.
- `validates`: The {input, output} parameters to this kernel.
- `state`: State of the parameter, from `vx_parameter \_state_e`.
- `name`: The short name of the module to load.
- `module`: The short name of the module to load.

- `ref`: The reference to the kernel.
- `kernel`: The reference to the overall context.
- `dir`: Direction of the parameter, from `vx_direction_e`.
- `data_type`: Type of parameter, from `vx_type_e`.
- `numParams`: The number of parameters for this kernel.
- `validates`: The {input, output} parameters to this kernel.
- `state`: State of the parameter, from `vx_parameter \_state_e`.
- `name`: The short name of the module to load.
- `module`: The short name of the module to load.

- `ref`: The reference to the kernel.
- `kernel`: The reference to the overall context.
- `dir`: Direction of the parameter, from `vx_direction_e`.
- `data_type`: Type of parameter, from `vx_type_e`.
- `numParams`: The number of parameters for this kernel.
- `validates`: The {input, output} parameters to this kernel.
- `state`: State of the parameter, from `vx_parameter \_state_e`.
- `name`: The short name of the module to load.
- `module`: The short name of the module to load.

- `ref`: The reference to the kernel.
- `kernel`: The reference to the overall context.
- `dir`: Direction of the parameter, from `vx_direction_e`.
- `data_type`: Type of parameter, from `vx_type_e`.
- `numParams`: The number of parameters for this kernel.
- `validates`: The {input, output} parameters to this kernel.
- `state`: State of the parameter, from `vx_parameter \_state_e`.
- `name`: The short name of the module to load.
- `module`: The short name of the module to load.

- `ref`: The reference to the kernel.
- `kernel`: The reference to the overall context.
- `dir`: Direction of the parameter, from `vx_direction_e`.
- `data_type`: Type of parameter, from `vx_type_e`.
- `numParams`: The number of parameters for this kernel.
- `validates`: The {input, output} parameters to this kernel.
- `state`: State of the parameter, from `vx_parameter \_state_e`.
- `name`: The short name of the module to load.
- `module`: The short name of the module to load.

- `ref`: The reference to the kernel.
- `kernel`: The reference to the overall context.
- `dir`: Direction of the parameter, from `vx_direction_e`.
- `data_type`: Type of parameter, from `vx_type_e`.
- `numParams`: The number of parameters for this kernel.
- `validates`: The {input, output} parameters to this kernel.
- `state`: State of the parameter, from `vx_parameter \_state_e`.
- `name`: The short name of the module to load.
- `module`: The short name of the module to load.