Avoiding Copies with Images Created from Handle and ROI

Efficient IO

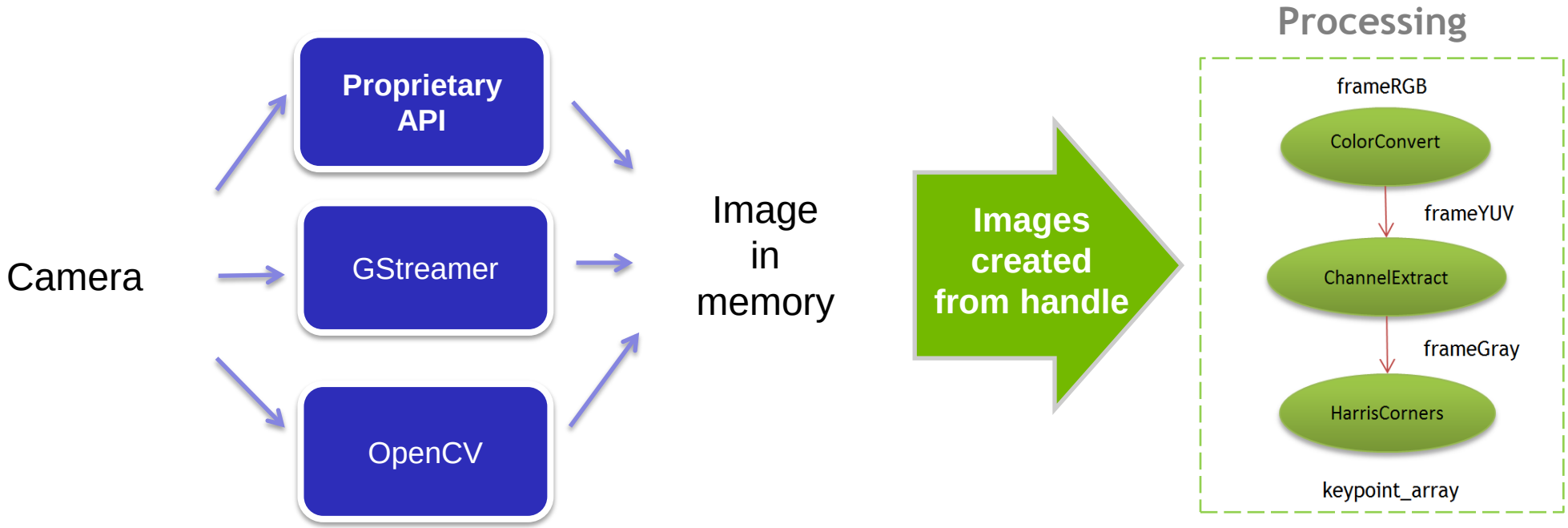
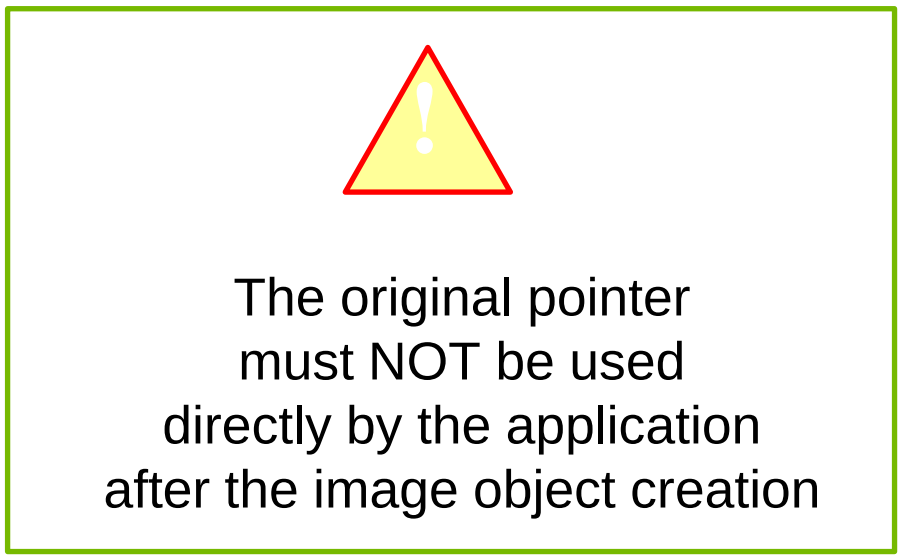


Image Created From Handle

Principle



The original pointer
must NOT be used
directly by the application
after the image object creation

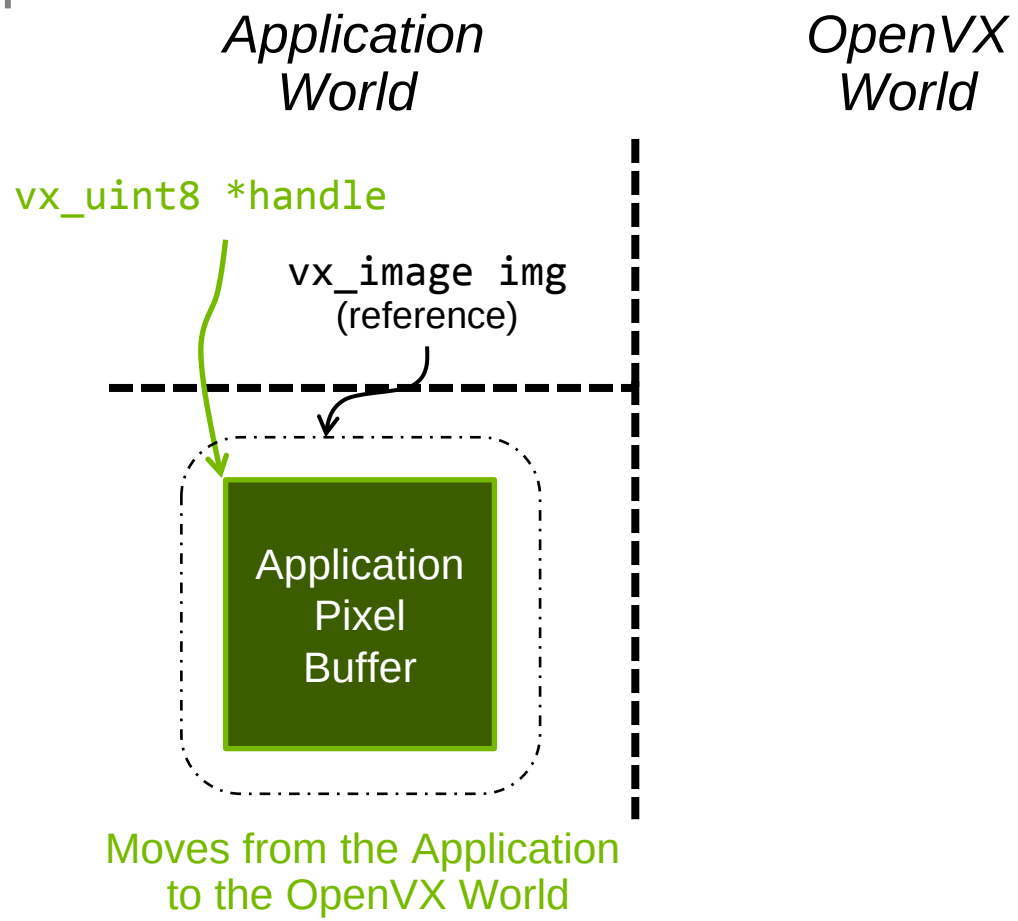


Image Created From Handle

Creation API

Memory type : VX_IMPORT_TYPE_HOST (can be extended)

```
vx_image img = vxCreateImageFromHandle(
                    context, VX_DF_IMAGE_RGB,
                    &addr[0],  // Plane layouts
                    &ptrs[0],  // Plane handles
                    VX_IMPORT_TYPE_HOST
                );
```

Useful for both input and output images

Image Created From Handle

Access with Standard Access/Commit

Image object access like any other image object :
access/commit function in map or copy mode

Mapped at its original address / memory layout

Ownership of memory returns back to the application at image destruction

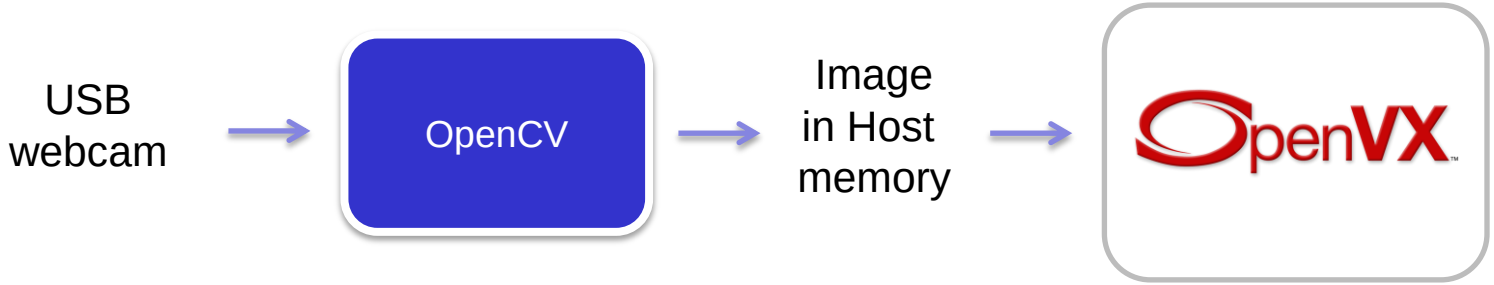
OpenVX 1.1 additions :

```
vx_status vxSwapImageHandle ( vx_image image, void const new_ptrs[],
                              void prev_ptrs[], vx_size num_planes )
```

Image Created From Handle Example

Example: OpenCV Interop

Import a Webcam image into OpenVX directly from the Host memory



OpenCV Interop Example

Import a webcam image

```
// Create a Video Capture from OpenCV
cv::VideoCapture inputVideo;
inputVideo.open( 0 ); // Grab data from the default webcam

// VideoCapture always returns a BGR image, transform it into RGB
cv::Mat cv_bgr, cv_rgb;
inputVideo.read( cv_bgr );
cv::cvtColor( cv_bgr, cv_rgb, cv::COLOR_BGR2RGB );

// Import into OpenVX
vx_imagepatch_addressing_t addr;
  addr.dim_x = cv_rgb.cols;
  addr.dim_y = cv_rgb.rows;
  addr.stride_x = 3*sizeof( vx_uint8 );
  addr.stride_y = cv_rgb.step;
void *ptrs[] = { cv_rgb.data };

vx_image vx_rgb = vxCreateImageFromHandle( context,
  VX_DF_IMAGE_RGB, &addr, ptrs, VX_IMPORT_TYPE_HOST );
```

OpenCV Interop Example

Refresh an Image

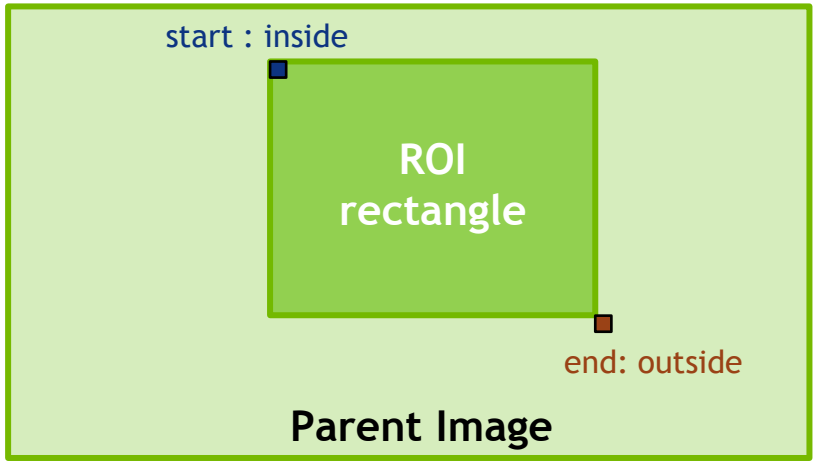
```
// Mapping an image created from handle will map at the
// exact same address and with the same memory layout
void *base = NULL; // NULL means ‘map’
vx_imagepatch_addressing_t addr;
vx_rectangle_t rect = { 0u, 0u, cv_rgb.cols, cv_rgb.rows };
vxAccessImagePatch( vx_rgb, &rect, 0, &addr, &base, VX_WRITE_ONLY );

// Refresh the OpenCV image
inputVideo.read( cv_src_bgr );
cv::cvtColor( cv_src_bgr, cv_src_rgb, cv::COLOR_BGR2RGB );

// Commit back changes
vxCommitImagePatch( vx_rgb, &rect, 0, &src_addr, base );
```

Image ROI

Rectangular sub-image

The same format as the parent image

Share pixels with the parent image (same memory)



struct vx_rectangle_t		
vx_uint32	start_x	The Start X coordinate.
vx_uint32	start_y	The Start Y coordinate.
vx_uint32	end_x	The End X coordinate.
vx_uint32	end_y	The End Y coordinate.

Focus: Images

ROI Example: Stereo Images

```
vx_rectangle_t left_rect = { 0, 0, width,  height };
vx_image leftROI = vxCreateImageFromROI( inputRGB, &left_rect );

vx_rectangle_t right_rect = { width, 0, 2*width, height };
vx_image rightROI = vxCreateImageFromROI( inputRGB, &right_rect );
```

Input images from the Middlebury stereo dataset (http://vision.middlebury.edu/stereo/data)