Tutorial Practice Session

Step 2: Graphs

Why graphs?

- **Most APIs (e.g., OpenCV) are based on function calls**
  - a function abstracts an algorithm that processes input data and produces output
  - the function can be optimized independent of all the other functionality

- **An application executes many function calls**
  - the call sequence of the function really defines a graph

- **There are limits to how much functions can be optimized**
  - but if you know that function B is called after A, you might be able to combine them to a new function that is much faster than time(A) + time(B)

- **By building first a graph of the function calls, and only executing later, we open up lots of optimization possibilities**
  - this is also how graphics APIs like OpenGL work

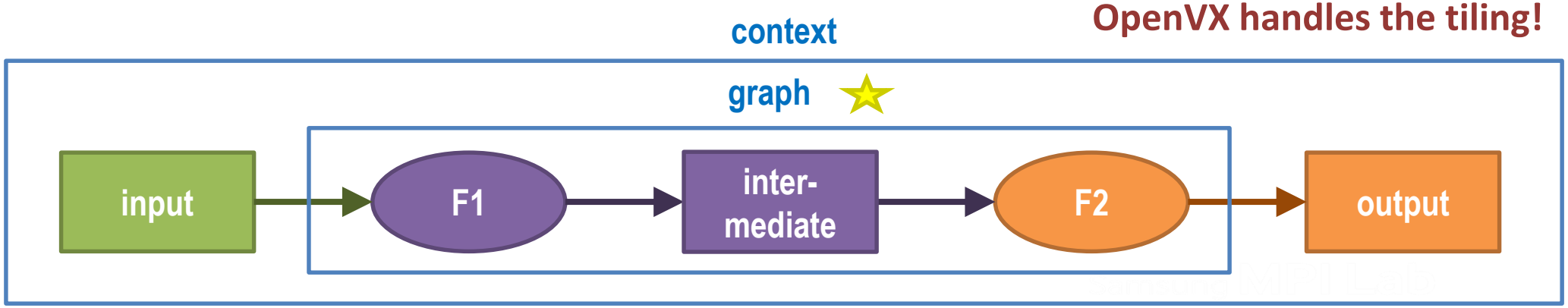
Optimization opportunities

- **Fuse kernels**
  - while accessing the image, do many things at the same time
- **Process the images in tiles**
  - for better locality, memory access coherency
- **Parallelize**
  - with more work that is available, more parallelization opportunities

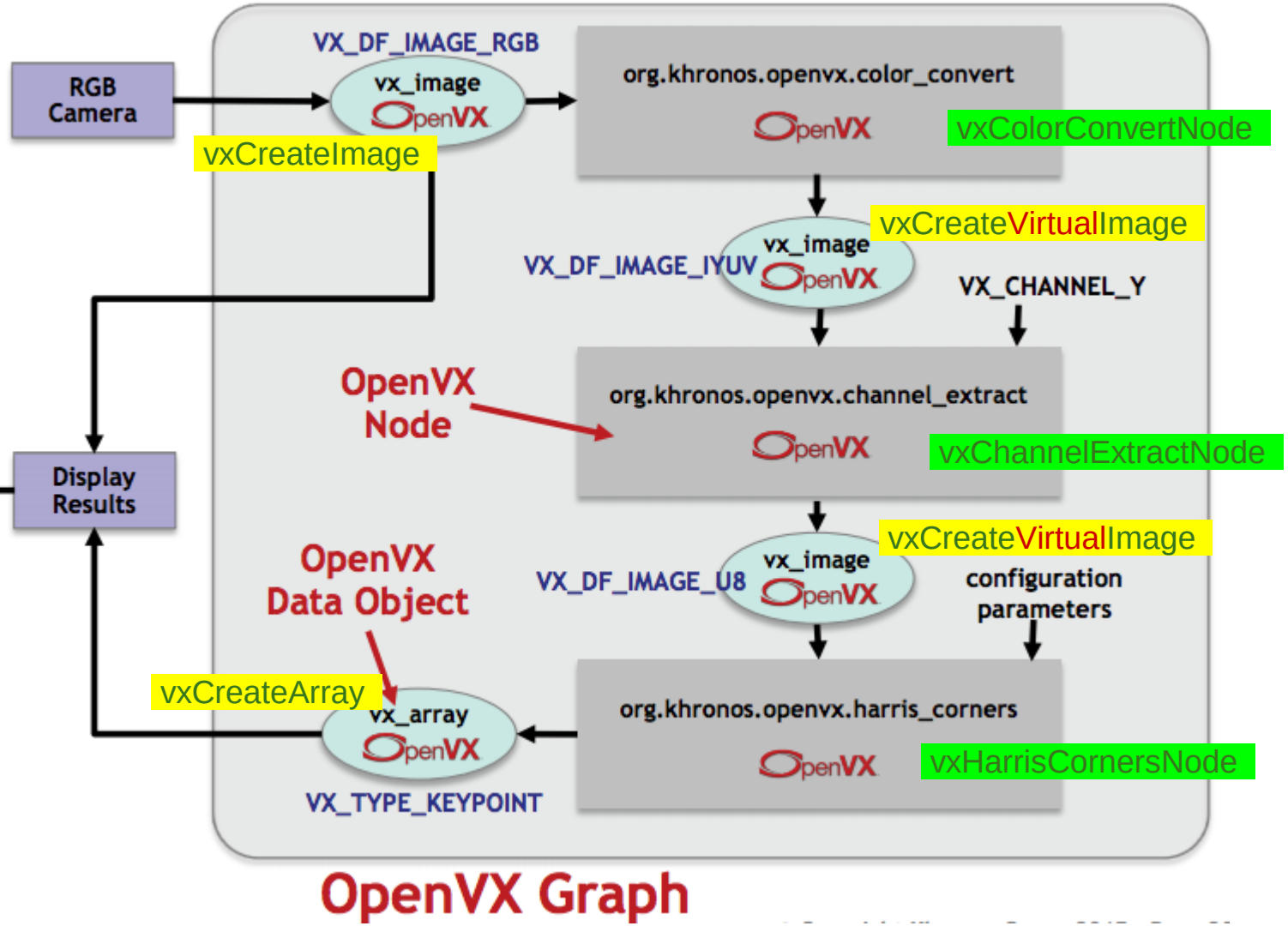
OpenVX Code

```
vx_context context = vxCreateContext();
vx_image input = vxCreateImage( context, 640, 480, VX_DF_IMAGE_U8 );
vx_image output = vxCreateImage( context, 640, 480, VX_DF_IMAGE_U8 );

vx_graph graph = vxCreateGraph( context );
vx_image intermediate = vxCreateVirtualImage( graph, 640, 480, VX_DF_IMAGE_U8 );
vx_node F1 = vxF1Node( graph, input, intermediate );
vx_node F2 = vxF2Node( graph, intermediate, output );

vxVerifyGraph( graph );
vxProcessGraph( graph ); // run in a loop
```

OpenVX handles the tiling!



Example: Keypoint Detector



Example: Feature Tracking Graph

Image capture API

RGB frame

color convert

frameYUV

channel extract

frameGray

gaussian pyramid

pyr_delay

pts_delay

pyr -1

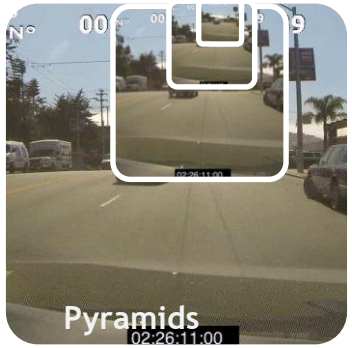
pyr 0

pts -1

pts 0

optical flow pyrLK

Array of keypoints



New data object types:

- **Pyramid** object
- **Delay** object

pts_delay is delay of keypoint array objects.

pyr_delay is delay of pyramid objects.

Display API

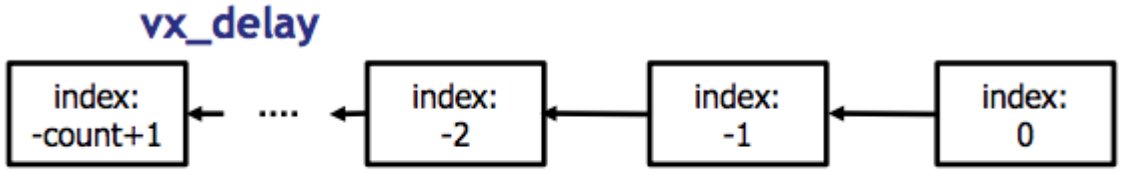
Pyramid Data Object

```
vx_pyramid vxCreatePyramid (
    vx_context context,
    vx_size levels,
    vx_float32 scale, // VX_SCALE_PYRAMID_HALF or VX_SCALE_PYRAMID_ORB
    vx_uint32 width,
    vx_uint32 height,
    vx_df_image format // VX_DF_IMAGE_U8
);
```

exercise2.cpp <Select Symbol>

```
114     //   lk_pyramid_levels       - number of pyramid levels for LK optical flow
115     //   lk_termination          - can be VX_TERM_CRITERIA_ITERATIONS or
116     //                                VX_TERM_CRITERIA_EPSILON or
117     //                                VX_TERM_CRITERIA_BOTH
118     //   lk_epsilon              - error for terminating the algorithm
119     //   lk_num_iterations       - number of iterations
120     //   lk_use_initial_estimate - turn on/off use of initial estimates
121     //   lk_window_dimension     - size of window on which to perform the algorithm
122     vx_uint32  width                  = gui.GetWidth();
123     vx_uint32  height                 = gui.GetHeight();
124     vx_size    max_keypoint_count     = 10000;
125     vx_float32 harris_strength_thresh = 0.0005f;
126     vx_float32 harris_min_distance    = 5.0f;
127     vx_float32 harris_k_sensitivity   = 0.04f;
128     vx_int32   harris_gradient_size   = 3;
129     vx_int32   harris_block_size      = 3;
130     vx_uint32  lk_pyramid_levels      = 6;
131     vx_float32 lk_pyramid_scale       = VX_SCALE_PYRAMID_HALF;
```

Delay Data Object



Delay container types:
vx_image, vx_pyramid, vx_array, vx_scalar,
vx_matrix, vx_distribution, vx_remap, vx_lut, vx_threshold

```
vx_delay vxCreateDelay
 (
    vx_context context,
    vx_reference exemplar,
    vx_size count
 );
```

Example:

```
vx_pyramid exemplar = vxCreatePyramid(context, …);
vx_delay pyr_delay = vxCreateDelay(context, (vx_reference)exemplar, 2);
vxReleasePyramid(&exemplar);

…
vx_pyramid pyr_0 = (vx_pyramid)vxGetReferenceFromDelay(pyr_delay, 0);
vx_pyramid pyr_1 = (vx_pyramid)vxGetReferenceFromDelay(pyr_delay, -1);
…

vxAgeDelay(pyr_delay);
```

Feature Tracking Graph ...

Image capture API

RGB frame

color convert

frameYUV

channel extract

frameGray

gaussian pyramid

pyr_delay

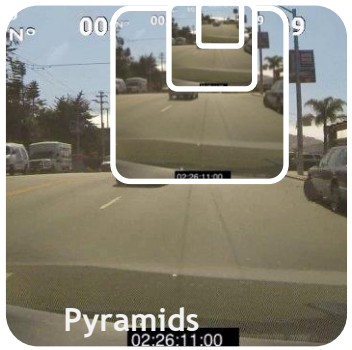
pts_delay

pyr -1 | pyr 0

pts -1 | pts 0

optical flow pyrLK

Array of keypoints



After each frame execution:

- **show arrow from pts_delay[-1] to pts_delay[0] for each feature keypoint**
- **age pyr_delay**
- **age pts_delay**

Display API

Tracking can be started from frame#1 …

Image capture API

RGB frame

color convert

frameYUV

channel extract

frameGray

gaussian pyramid

pyr_delay

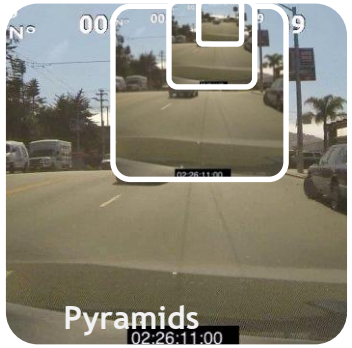
pts_delay

pyr -1 | pyr 0

pts -1 | pts 0

optical flow pyrLK

Array of keypoints



optical flow on frame#N requires:

- **pts_delay[-1] with features in frame#N-1**
- **pyr_delay[-1] with pyramid of frame#N-1**
- **pyr_delay[0] with pyramid of frame#N**

Display API

Harris on frame#0 ...

Image capture API

RGB frame

color convert

frameYUV

channel extract

frameGray

gaussian pyramid

pyr_delay

pts_delay

pyr -1

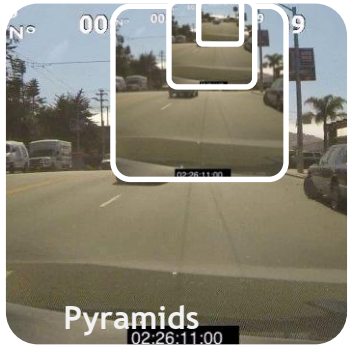
pyr 0

pts -1

pts 0

harris corners

Array of keypoints



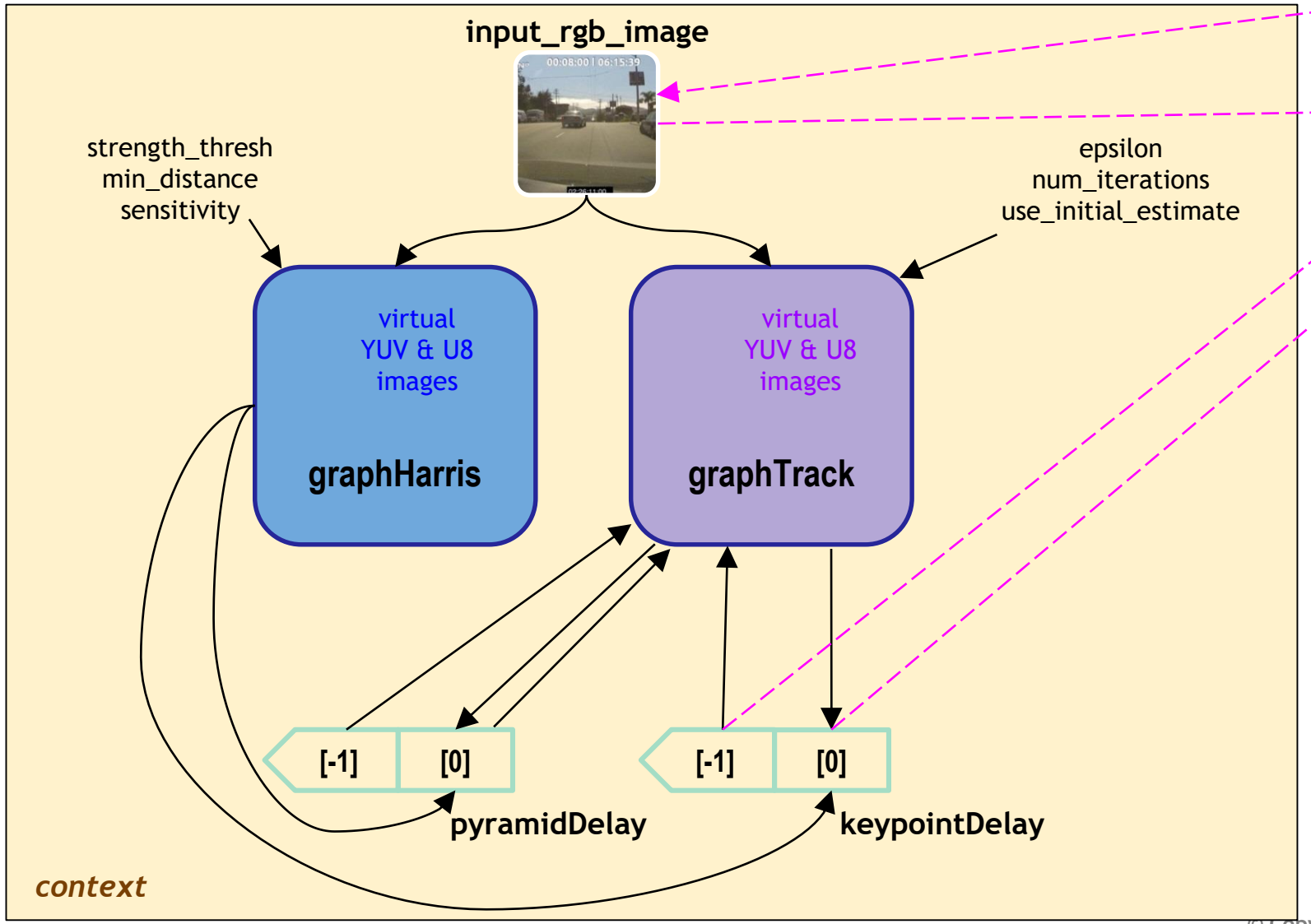
You need to create two graphs:
- **Harris graph**
- **Tracking graph**

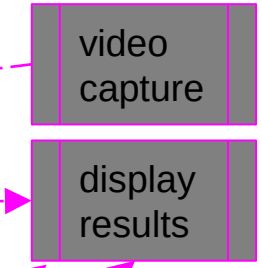
Run Harris graph on frame#0.

Run Tracking graph on all remaining frames...

Display API

Exercise2 in a nut-shell





create all data objects and graphs within a context.

verify graphs.

process each frame:
- read input frame
- process graph:
  harris, if frame#0
  track, otherwise
- display results
- age delay objects