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

```c
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

New data object types:
- **Pyramid** object
- **Delay** object

- **pts_delay** is delay of keypoint array objects.
- **pyr_delay** is delay of pyramid objects.

Array of keypoints
Pyramid Data Object

```cpp
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
);
```

```cpp
vx_uint32 width = gui.GetWidth();
vx_uint32 height = gui.GetHeight();
```

<table>
<thead>
<tr>
<th>line</th>
<th>symbol</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>114</td>
<td>//</td>
<td>lk_pyramid_levels - number of pyramid levels for LK optical flow</td>
</tr>
<tr>
<td>115</td>
<td>//</td>
<td>lk_termination - can be VX_TERM_CRITERIA_ITERATIONS or</td>
</tr>
<tr>
<td>116</td>
<td>//</td>
<td>VX_TERM_CRITERIA_EPSILON or</td>
</tr>
<tr>
<td>117</td>
<td>//</td>
<td>VX_TERM_CRITERIA_BOTH</td>
</tr>
<tr>
<td>118</td>
<td>//</td>
<td>lk_termination - error for terminating the algorithm</td>
</tr>
<tr>
<td>119</td>
<td>//</td>
<td>lk_num_iterations - number of iterations</td>
</tr>
<tr>
<td>120</td>
<td>//</td>
<td>lk_use_initial_estimate - turn on/off use of initial estimates</td>
</tr>
<tr>
<td>121</td>
<td>//</td>
<td>lk_window_dimension - size of window on which to perform the algorithm</td>
</tr>
<tr>
<td>122</td>
<td>//</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>//</td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>vx_size</td>
<td>max_keypoint_count = 10000;</td>
</tr>
<tr>
<td>125</td>
<td>vx_float32</td>
<td>horris_strength_thresh = 0.0005f;</td>
</tr>
<tr>
<td>126</td>
<td>vx_float32</td>
<td>horris_min_distance = 5.0f;</td>
</tr>
<tr>
<td>127</td>
<td>vx_float32</td>
<td>horris_k_sensitivity = 0.04f;</td>
</tr>
<tr>
<td>128</td>
<td>vx_int32</td>
<td>horris_gradient_size = 3;</td>
</tr>
<tr>
<td>129</td>
<td>vx_int32</td>
<td>horris_block_size = 3;</td>
</tr>
<tr>
<td>130</td>
<td>vx_int32</td>
<td>lk_pyramid_levels = 6;</td>
</tr>
<tr>
<td>131</td>
<td>vx_float32</td>
<td>lk_pyramid_scale = VX_SCALE_PYRAMID_HALF;</td>
</tr>
</tbody>
</table>
```
Delay Data Object

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

**Example:**

```c
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 ...

After each frame execution:
- **show arrow** from `pts_delay[-1]` to `pts_delay[0]` for each feature keypoint
- `age pyr_delay`
- `age pts_delay`
Tracking can be started from frame#1 ...

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
Harris on frame#0 ...

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

Run Harris graph on frame#0.

Run Tracking graph on all remaining frames...
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