# Leveraging OpenCV 3.0 on Intel® Graphics Technology with OpenCL™

Maxim Shevtsov, Software Architect, Intel Corporation, maxim.y.shevtsov@intel.com

Many companies including Intel contribute optimizations for the OpenCL™ code in OpenCV

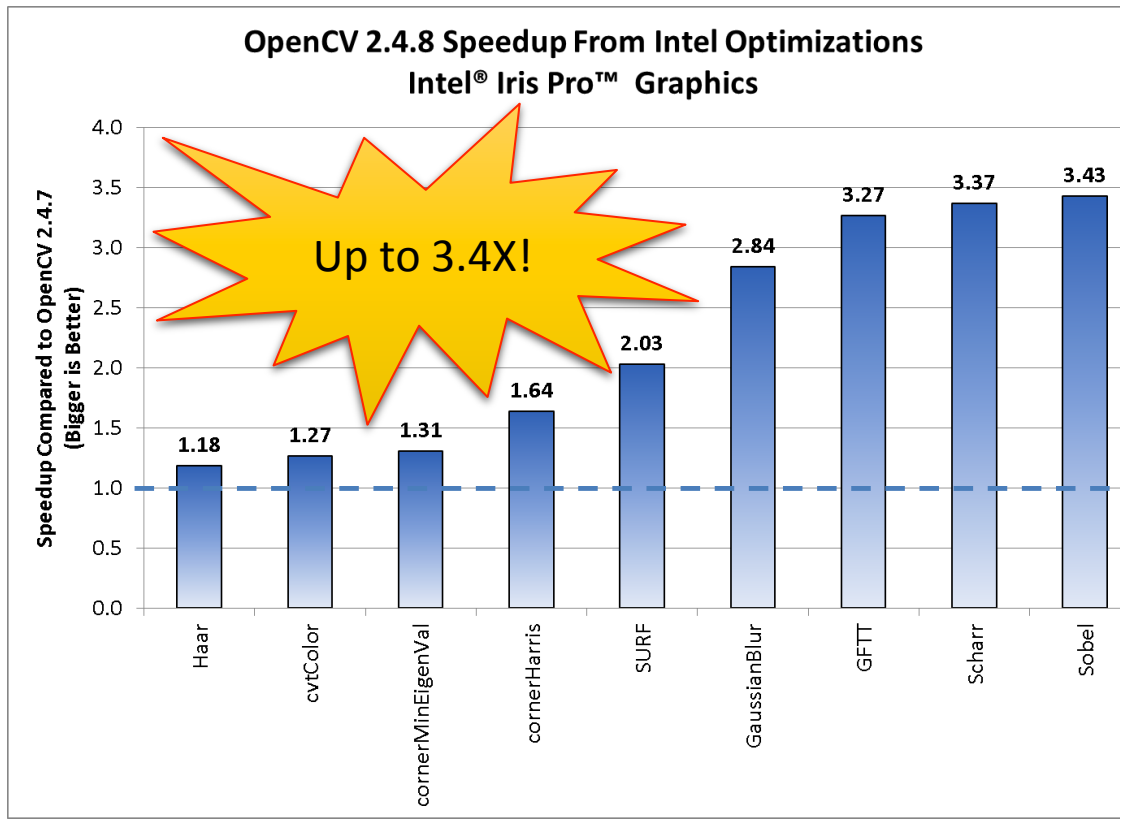
- Initial work for 2.4.8
- Propagated to 3.0

Upcoming OpenCV 3.0 architecture improves support for **GPUs**

- "Transparent API" enables same code to use CPU or **OpenCL**
- Little or no code changes from existing OpenCV code
- On platforms without OpenCL uses efficient CPU fallback
- Improved use of shared physical memory for integrated GPU perf

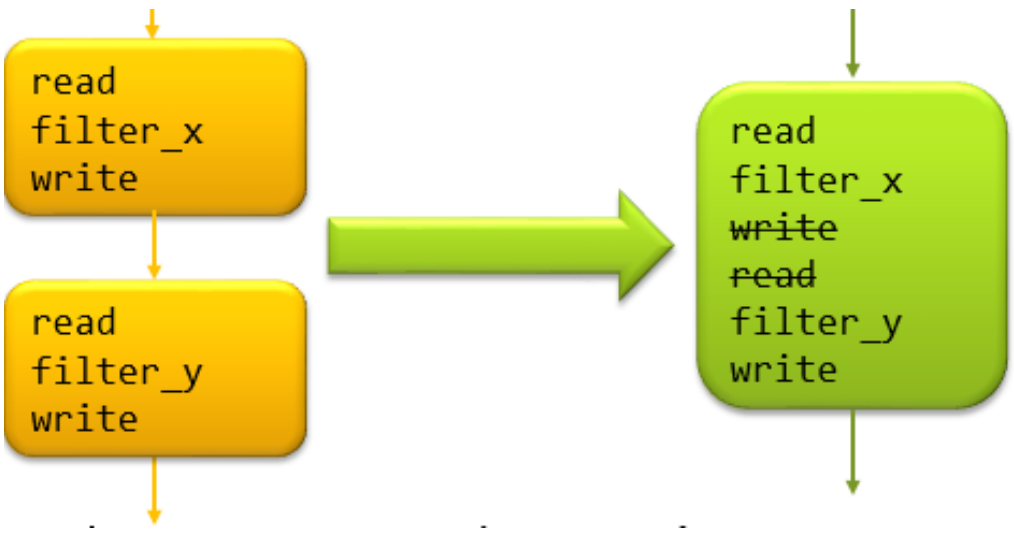
Intel optimizations delivered substantial performance improvements

- Most optimizations are general (improve performance on all GPUs)



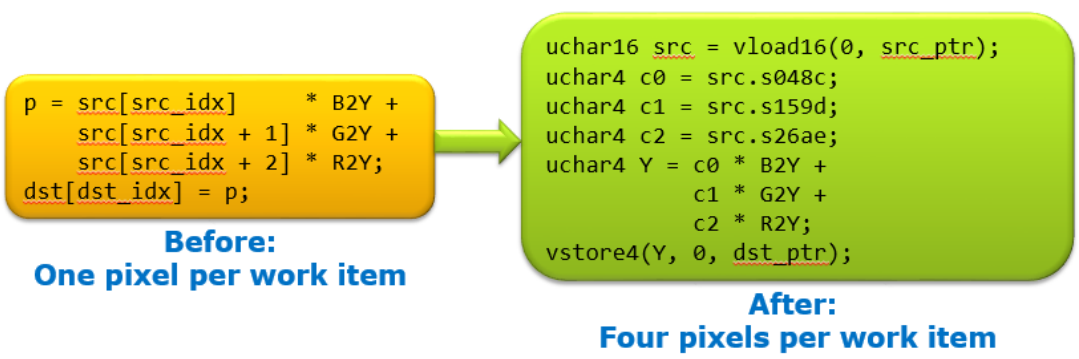

**Optimization #1**: **Reduce memory overdraw with kernel fusion**

- Reduces GPU/CPU round trips and associated overheads
- Examples: separable filters like Sobel/Sharr/Blur and combos (cornerHarris)




**Optimization #2**: **Process multiple pixels per work item**

- Amortizes GPU scheduling
- Amortizes GPU data path
- Example: cvtColor




**Optimizations #3: Host to Device Transfers**

- Host (CPU) and Device (GPU) share the same physical memory
- No transfers for OpenCL buffers (zero copy)!

Take advantage of shared physical memory!

- Allocate system memory aligned to a cache line (64 bytes)
- Allocate an even number of cache lines
- Create buffer with system memory pointer and CL_MEM_USE_HOST_PTR
  - Use clEnqueueMapBuffer() to access data

**Optimizations #4**:

- Avoid redundant resource allocation/dealloc
- Pre-compile and cache kernels

## OpenCV 3.0 is designed to better use of OpenCL™ on Intel® Architecture