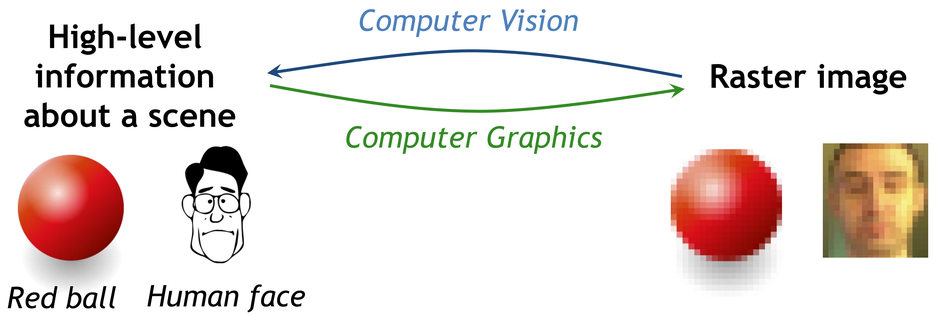
Accelerating Computer Vision with Tegra GPU

Andrey Pavlenko (Itseez), Kirill Kornyakov (Itseez), Colin Tracey (NVIDIA)

Tegra 3 SoC

- 4 ARM cores with NEON SIMD
- Ultra-Low Power GPU
- 12 shader cores
- GLES 2.0, EGL 1.4
- Possibility to share memory between CPU and GPU

Computer Vision on GPU



The same hardware boosts both!

OpenCV

- Open-source computer vision library
- Liberal BSD license
- Going to be a Khronos standard
- 500+ computer vision algorithms
- 3M+ downloads
- Cross-platform: *Windows/Linux/Mac/Android/iOS*

OpenCV on Tegra GPU

Geometric Transformations

- **cv::resize – 7-10x** speedup
  1280x720 RGBA -> 640x480 : 5.3 ms
- **cv::warpAffine – 4-8x** speedup
  1280x720 RGBA : 16.6 ms
- **cv::warpPerspective – 8-15x** speedup
  1280x720 RGBA : 18.9 ms
- **cv::SphericalWarper::warp – 15-18x** speedup
  1280x720 RGBA : 27 ms

Sample Code

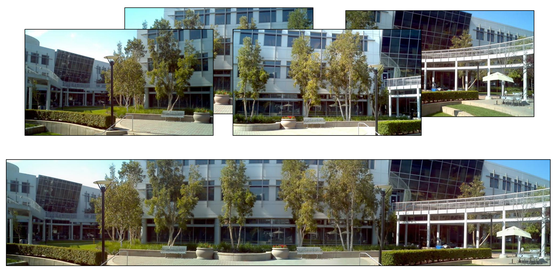
```
// warpAffine fragment shader
varying vec2 uv0;
uniform sampler2D tex0;
uniform float M[6];
uniform vec2 widthHeight;

void main() {
    vec2 uv = uv0 * widthHeight;
    float x = M[0]*uv.x + M[1]*uv.y + M[2];
    float y = M[3]*uv.x + M[4]*uv.y + M[5]);
    gl_FragColor = texture2D(tex0, vec2(x,y) /
         widthHeight);
}
```

Current Limitations

1. Image size <= 2K x 2K pixels
2. Only GLSL, no CUDA on Tegra 3
3. Only RGBA_8888 format is accessible for both GPU & CPU without copy

Computer Vision applications on Tegra GPU

Panorama Stitching



Bottlenecks ported to GPU

- cv::resize
- cv::SphericalWarper::warp

Total app speedup is **1.5-2.0x.**

No C++ code changes, no recompilation is required.
As a result, both applications are working in real time.

Video Stabilization

Bottleneck ported to GPU

- cv::resize

Total app speedup is **2.0x.**