Itseez

- Russian R&D company focused on Computer Vision
- Primary OpenCV contributor and maintainer
- Khronos group member, participates in OpenVL development
Agenda

- Mobile computer vision
- OpenCV4Android
- OpenCV4Tegra
- Development tips
- Practice notes
OpenCV overview

- General Image Processing Functions
- Geometric descriptors
- Image Pyramids
- Camera calibration, Stereo, 3D
- Segmentation
- Features
- Utilities and Data Structures
- Transforms
- Tracking
- Machine Learning:
  - Detection,
  - Recognition
- Fitting
- Matrix Math
Mobile Computer Vision

• Why is it interesting?
Camera phones

- Smartphone is the most popular photo camera
Computational photography

- Face detection
- Panorama stitching
- HDR
- Instagram
Augmented reality
More mobile CV

• Video stabilization

• Gesture interface

• Face recognition

• What’s next?
Your turn!

• *Use mobile OpenCV to make your killer app!*
Android

- Leading mobile OS
Android Development

• Java is the only front door
  – JDK, Android SDK, Eclipse+ADT

• Native C/C++ code via JNI
  – Android NDK

• Target devices variety
  – HW: CPU features, GPU, cameras, screen resolutions
  – OS: Android 2.x - 4.x

• Performance / responsiveness

• Environment setup tutorial at docs.opencv.org
OpenCV4Android

- 2 years old, sponsored by NVIDIA
- 4 official releases, 80K+ downloads
- Active community
  - Google group (1100 members, 700 threads)
  - Q&A site (850 members, 700 questions)
- Nightly builds and tests (Buildbot)
OpenCV4Android

• C++ API
• Java API
• Samples
• Tutorials
• Tests

• Visit opencv.org/android for details
Android app
Android CV app
For Java developers

• Most popular C++ API wrapped to Java
  – Java API very close to C++
  – Low overhead
  – Generated automatically

• Set of JUnit tests, Android samples and tutorials

• Familiar with Android?
  Start using OpenCV via Java API!
For native developers:

- Almost full C/C++ API
  - Native camera

- ARM (v5, v7a), x86, MIPS (since 2.4.3)

- Multi-threading enabled with Intel TBB

- Have desktop C++ code with OpenCV? Compile it for Android and call via JNI!
Limitations

• Some functions (e.g. GUI) are N/A

• Video decoding/encoding isn’t supported (coming soon)

• Native camera is incompatible with some devices (TI, Qualcomm)
Getting OpenCV4Android

• SDK packages
  – SourceForge
  – NVIDIA Tegra Android Dev Pack 2.0 (soon)

• Build from sources (usually not needed!)
  – co-developers
  – customized OpenCV
CV app distribution

Deprecated way

• Include OpenCV libs to app package (APK)

• Libs for all platforms are needed (or per-platform app packages)

• Duplicated for every app
CV app distribution

**OpenCV Manager** (since 2.4.2)

- Separate package (APK) providing OpenCV API
- Shares single OpenCV instance across apps (small app APK size)
- OpenCV updates via Google Play
- Use the best HW-matching libs (HW-accelerated if exists)
Distribution models

Deprecated way

OpenCV Manager
OpenCV performance optimizations

• Multi-threading with Intel TBB
  – Multi-core

• SSE
  – x86

• Closed source (proprietary)
NVIDIA Tegra 3

- Quad-core ARM
  - Cortex-A9 with NEON

- ULP GeForce GPU
  - 12 shader cores
OpenCV4Tegra

• Alternative OpenCV4Android library
• Fully compatible API
• Some functions are optimized
  – with NEON
  – with GPU

• Realtime Computer Vision with OpenCV
OpenCV4Tegra

Figure 6. Performance improvement with NEON on Tegra 3.

Time (ms)

- Canny: 1.6x
- Median Blur: 23x
- Optical Flow: 1.6x
- Color Conversion: 9.5x
- Morphology: 5.4x
- Gaussian Blur: 4.6x
- FAST Detector: 2.6x
- Sobel: 3.1x
- pyrDown: 3.4x
- Image Resize: 7.6x

Comparison between Tegra CPU and Tegra NEON.
OpenCV4Tegra

Figure 7. Performance improvement with GLSL on Tegra 3.

- Median Blur: 2.4x
- Planal Warper: 13x
- warpPerspective: 9.8x
- Cylindrical Warper: 14x
- blur3x3: 5.7x
- warpAffine: 3.3x
OpenCV4Tegra

- Used automatically on compatible devices via Google Play Market if app uses OpenCV Manager
- Included into NVIDIA Tegra Android Dev Pack 2.0 (soon)
Development tips

• Use C++ for complex algorithms
  o develop and debug on desktop
  o call via JNI from Android Java app

• Care of performance / battery power
  o mobile ~10 times slower than desktop

• Debugging and profiling on device
  o logcat
  o gdbserver
  o platform vendors tools
Performance tips

• Use OpenCV Manager
• Algo goes 1\textsuperscript{st}: prefer light-weight approaches
• Work on lower resolution
• Use orientation sensors
• Avoid premature optimization, optimize bottlenecks

Usually performance and power efficiency go together.
Performance tips

• Prefer integer algorithms (ARM)
• Replace doubles with floats (ARM)
• Use multi-threading
• Use memory effectively
  – avoid memory reallocations and copying
  – reuse buffers in pipeline
  – process in-place
  – consider cache
    (data layout and access pattern)
Resources

- [http://opencv.org/android](http://opencv.org/android)
- [OpenCV on SourceForge](http://opencv.org)
- ['OpenCV4Android' google group](https://groups.google.com/a/google.com/group/OpenCV4Android)
- [OpenCV Q&A forum](http://answers.opencv.org)
- [OpenCV online docs](http://opencv.org)
  - Android tutorials,
  - Java API, C++ API,
  - OpenCV Manager
- [OpenCV Dev Zone](http://opencv.org)
- "Realtime Computer Vision with OpenCV"
Practice notes

Setup development environment with the “Introduction into Android Development” tutorial:

- Oracle/Sun JDK (v6)
- Android SDK + platforms (2.x, 3.x, 4.x)
- Android NDK (r8)
- Eclipse (3.7) + ADT + CDT
- Connect your device via USB or configure emulator
- Download and unpack OpenCV4Android SDK 2.4.2
Practice notes

We will go through the following steps:

• Camera preview app without OpenCV
  – Without frames modification
  – With frames modification
    (Tutorial 0)

• Add OpenCV to the camera preview app
  (Tutorial 1)

• Switch from Java camera to Native one
That’s it

Questions?
Good time to ask!

Start using OpenCV for Android now!