Embedded realtime computer vision

Victor Erukhimov

2016

itseez

Emerging mobile/embeddeduse cases

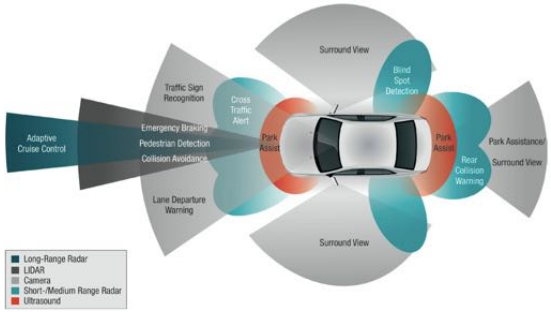
Computational photography

Robotics

Drones

Wearable
augmented reality

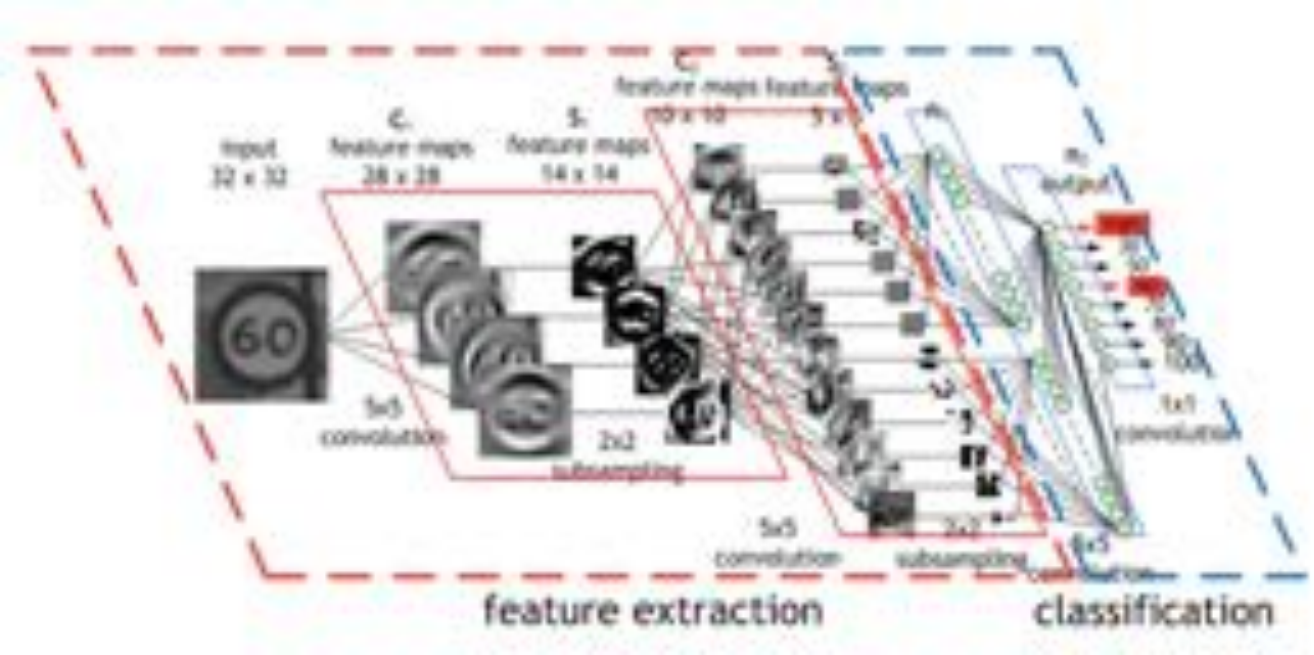
Driver assistance /
driverless cars



Wearable
virtual reality

itseez

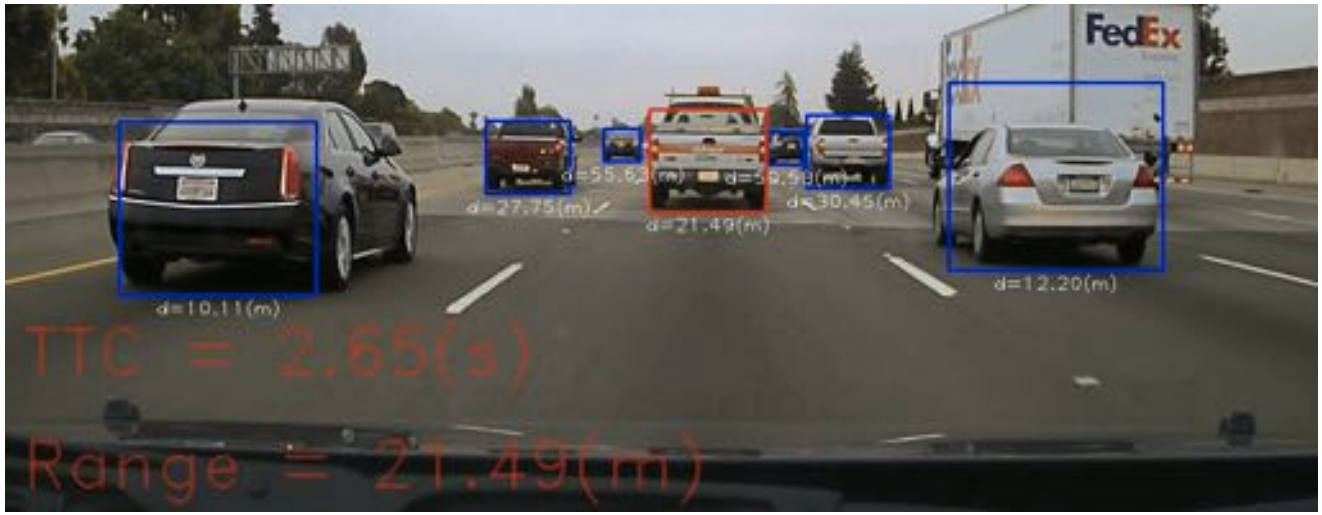
Convolutional neural networks



Detection involves a lot of convolution operations, that cannot be done in realtime on conventional embedded hardware

itseez

Example: driver assistance



Safety standards:

- MISRA C
- ISO 26262

itseez

Example: 3D scanning

Sample people models

itseez

Sample object models

itseez

3D reconstruction on a tablet: full body

Reconstruction on a tablet, 45 seconds (iPad Air 2)

Cloud reconstruction, 10 minutes

itseez

Motivation

2010: OpenCV experimental port to Android

Back camera:

- 5MP static images
- 480p video

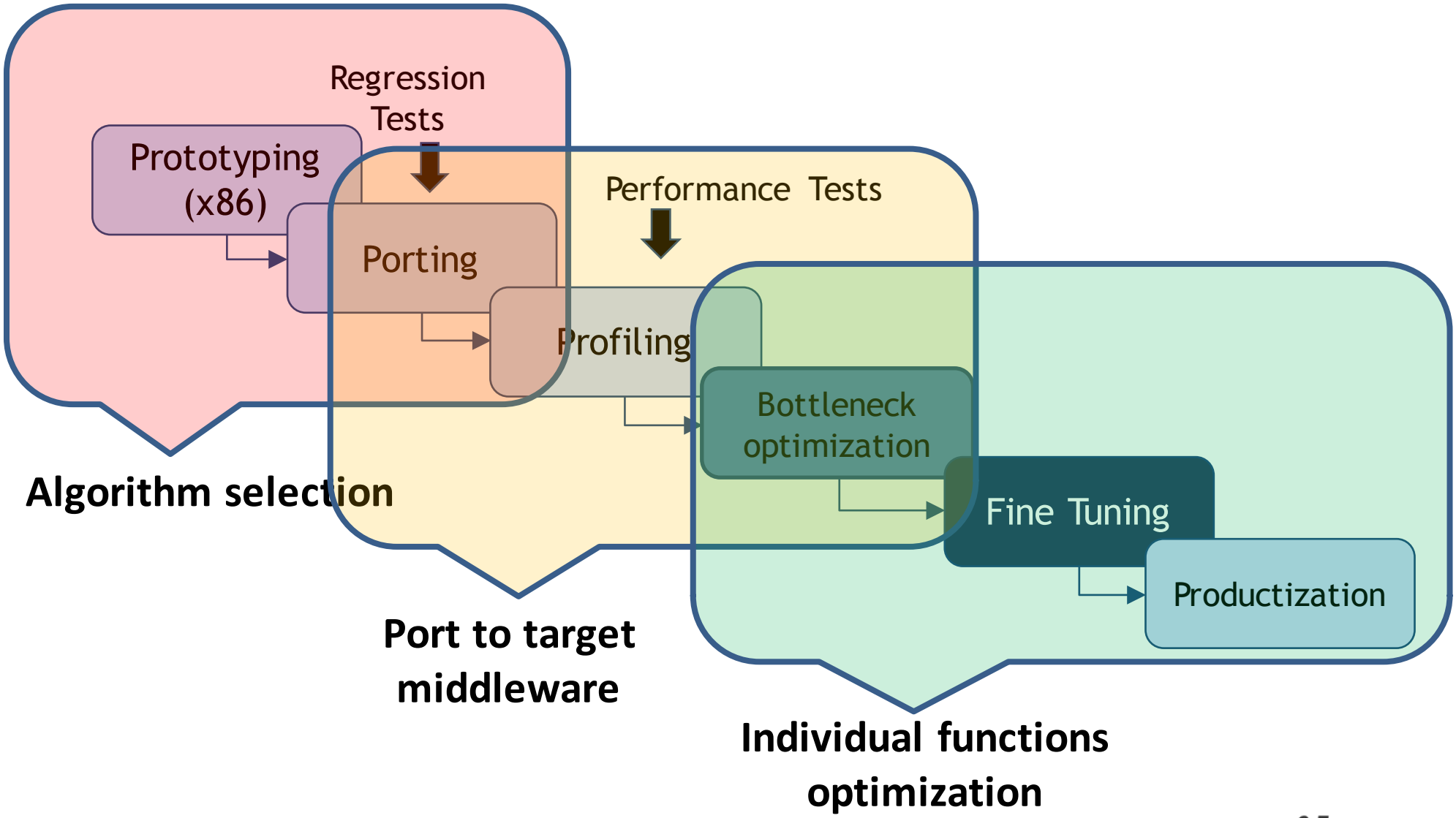
- Even cv::resize was too slow
- Wrappers to move images between android and opencv were not realtime

itseez

Embedded vision challenges

- **Need to run in real-time on embedded/mobile/wearable devices**
- **Intense and power hungry computations**
  - Heavy image processing requires significant memory bandwidth
    - Multiple cores do not help
- **Mobile platforms are even more complicated**
  - Unstable FPS impacts algorithm complexity (e.g. tracking)
  - Hard to predict when/if we are consuming too much
  - Thermal protection, power saving are hard to control and influence
- **Very few computer vision accelerators**

Embedded algorithm implementation



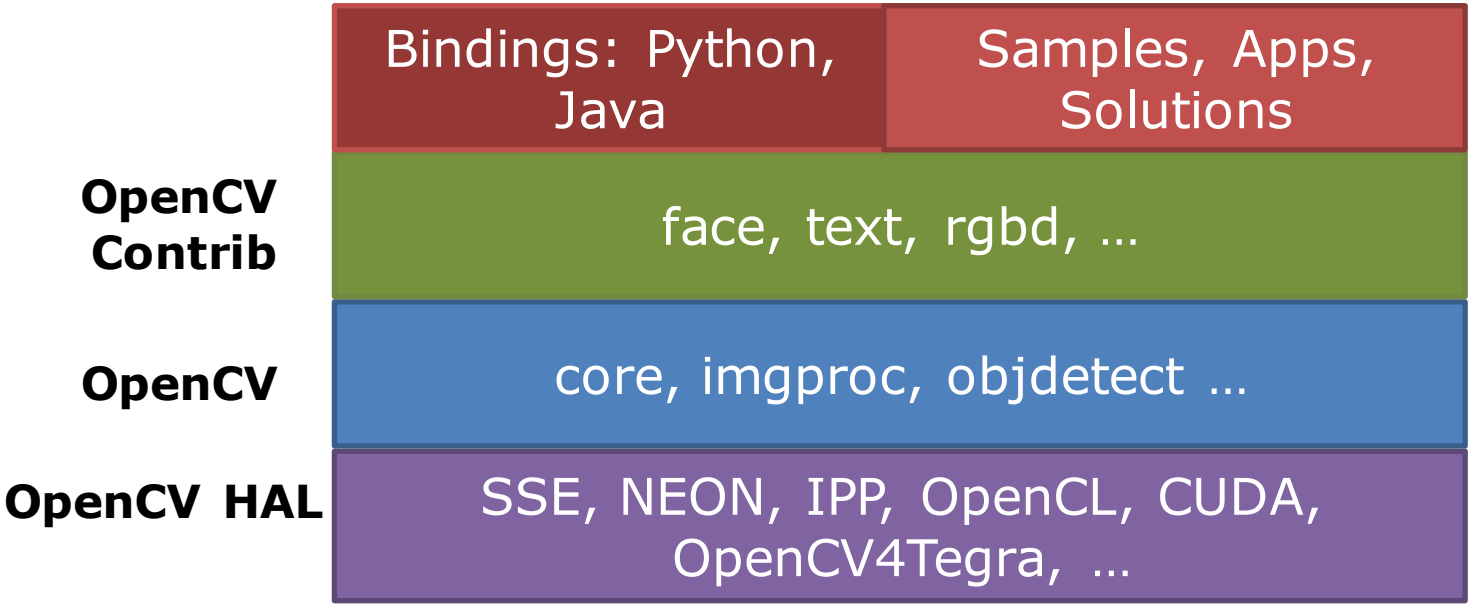
itseez

Available middleware frameworks

- **Proprietary platforms (e.g. FastCV, VisionWorks)**
- **OpenCV**
  - SSE, AVX
  - NEON
  - CUDA
  - OpenCL
  - Proprietary optimizations
- **OpenVX**

OpenCV at glance

- BSD license, **10M** downloads, **500K+** lines of code
- Huge community involvement, automated patch testing and integration process
- Runs everywhere



- Find more at http://opencv.org (user)
- Or http://code.opencv.org (developer)

itseez

OpenCV QA

Contribution/patch workflow:
see OpenCV wiki

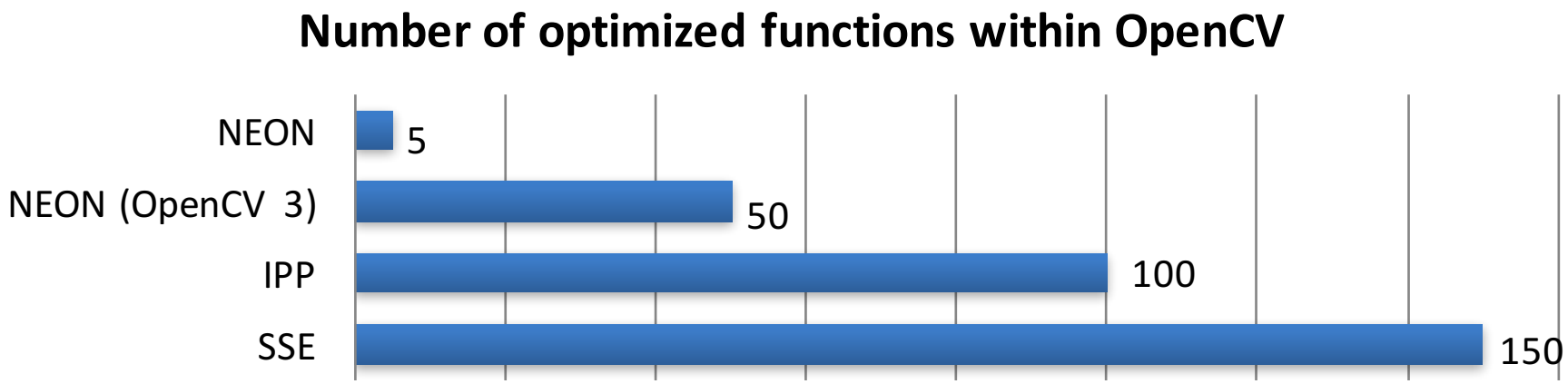
http://github.com/itseez/opencv

http://build.opencv.org: buildbot with 50+ builders

http://pullrequest.opencv.org: tests each patch

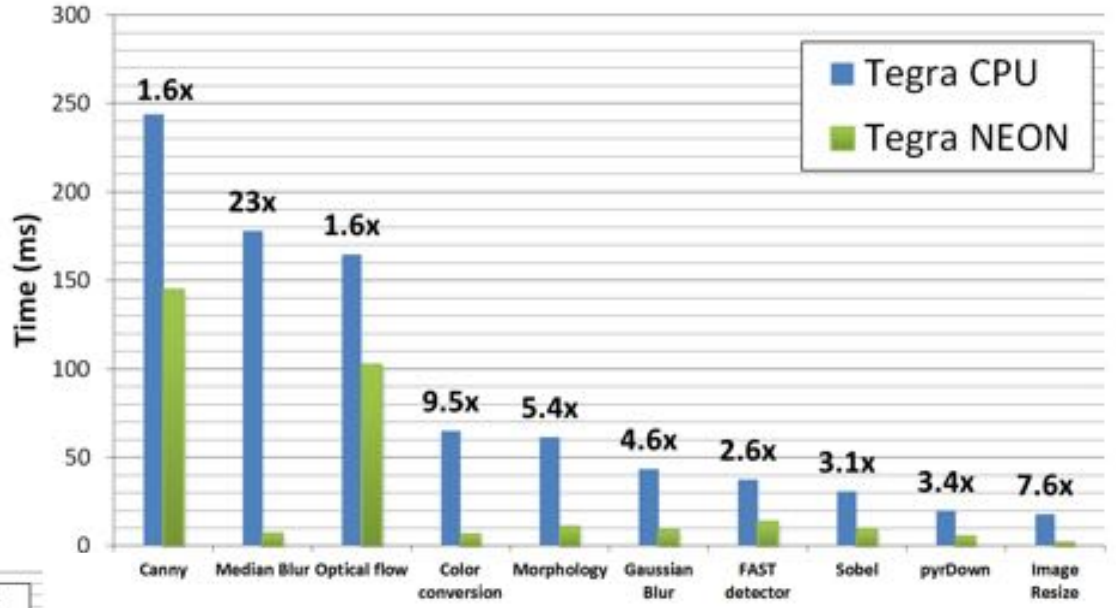
Why my OpenCV code is slow on mobile?

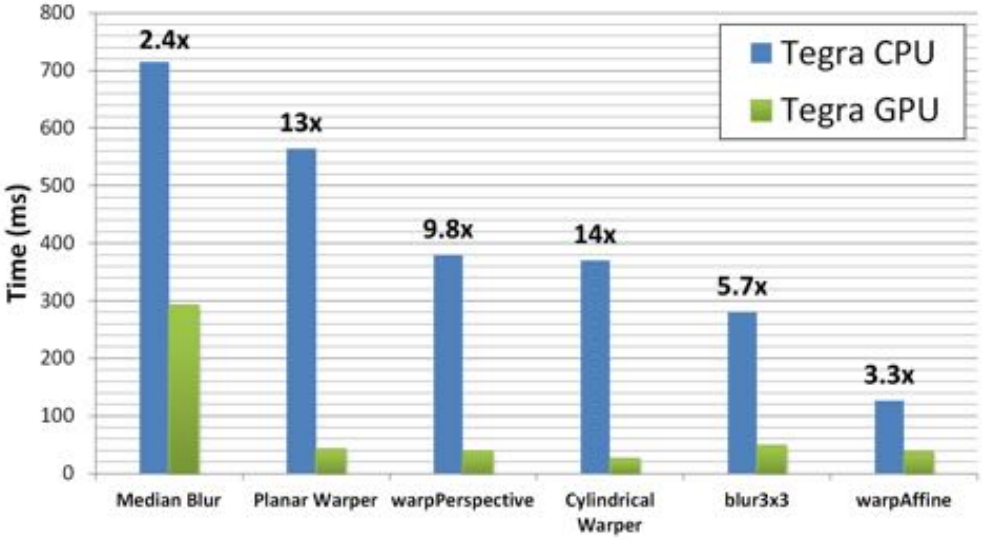
- OpenCV was initially optimized for desktop where it works fast
- ARM optimizations are far behind
- Scalar code does not perform on ARM as good as on x86
- Optimization might help to some extent



itseez

Early ARM/GLSL speedups*

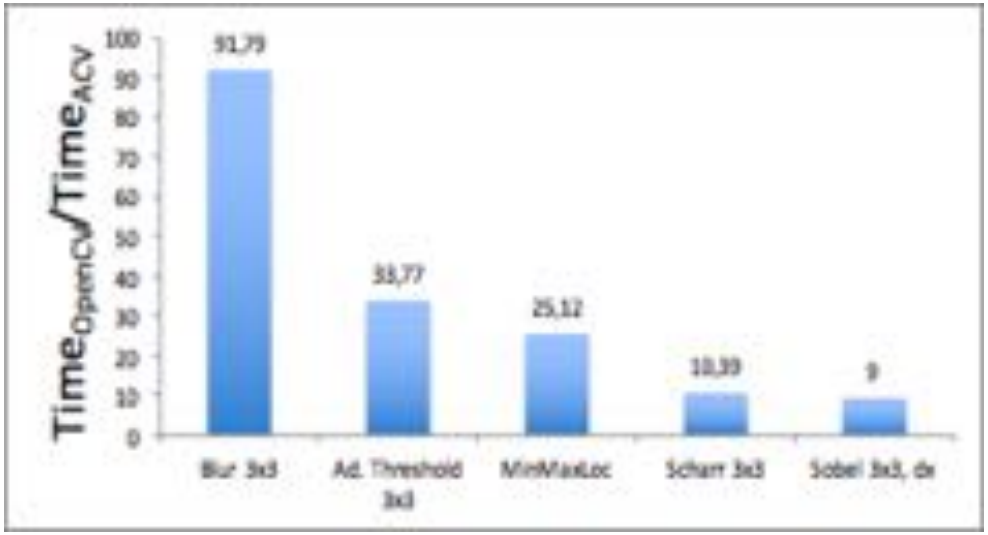


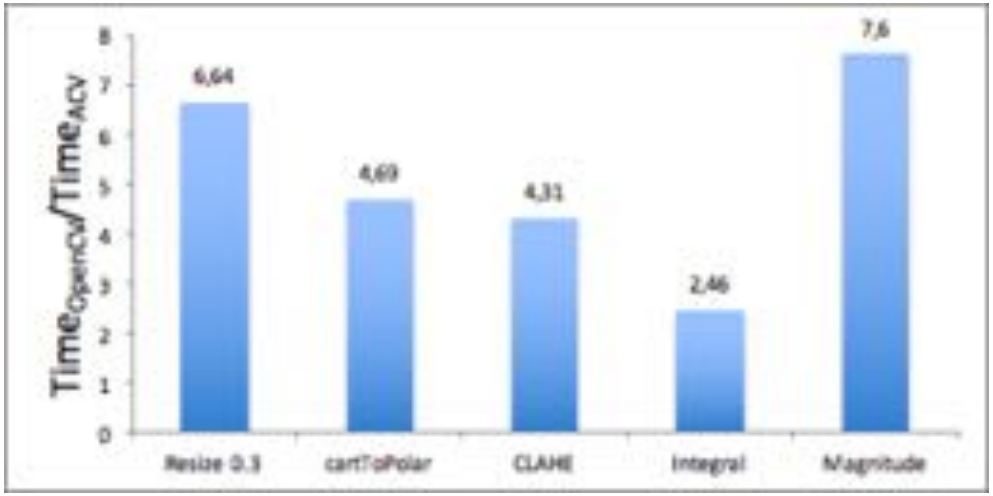


*Kari Puli et al, **Real-time computer vision with OpenCV**,
Communications of the ACM, Volume 55 Issue 6, June 2012, Pages 61-69

itseez

AcceleratedCV (Itseez) speedup against OpenCV





Comparison against public 2.4.9 release of OpenCV (compiled with GCC 4.9, auto vectorization ON, -O3)
Qualcomm Snapdragon 800 SoC (2.15 Ghz, 4 cores), Android
Input image size is 720p

Higher level functions speedups

- **Robotics algorithms**
  - Acceleration up to **2x**
  - Resize, Filtering, CLAHE, blur, LBP cascade, Optical Flow, FAST features
- **ADAS algorithms**
  - Acceleration about **20-50% (1.25x-2x)**
  - Threshold, MinMaxLoc, Resize, Optical Flow, Match Template

itseez

What is wrong with OpenCV as middleware for embedded platforms?

- **Transparent memory model**
  - Users have to optimize memory throughput themselves
- **Does not address heterogeneous platform structure**
  - Users have to manage different accelerators
- **Many optimization opportunities are lost**
  - Choosing optimal accelerator
  - Filter stacking
  - Tiling

Vision Acceleration

Khronos Connects Software to Silicon

Open Consortium creating ROYALTY-FREE, OPEN STANDARD APIs for hardware acceleration

Defining the roadmap for low-level silicon interfaces needed on every platform

Graphics, compute, rich media, vision, sensor and camera processing

Rigorous specifications AND conformance tests for cross-vendor portability

Acceleration APIs
BY the Industry
FOR the Industry

Well over a *BILLION* people use Khronos APIs *Every Day…*

http://accelerateyourworld.org/

Khronos Standards

Visual Computing
- 3D Graphics
- Heterogeneous Parallel Computing

3D Asset Handling
- 3D authoring asset interchange
- 3D asset transmission format with compression

KHRONOS GROUP

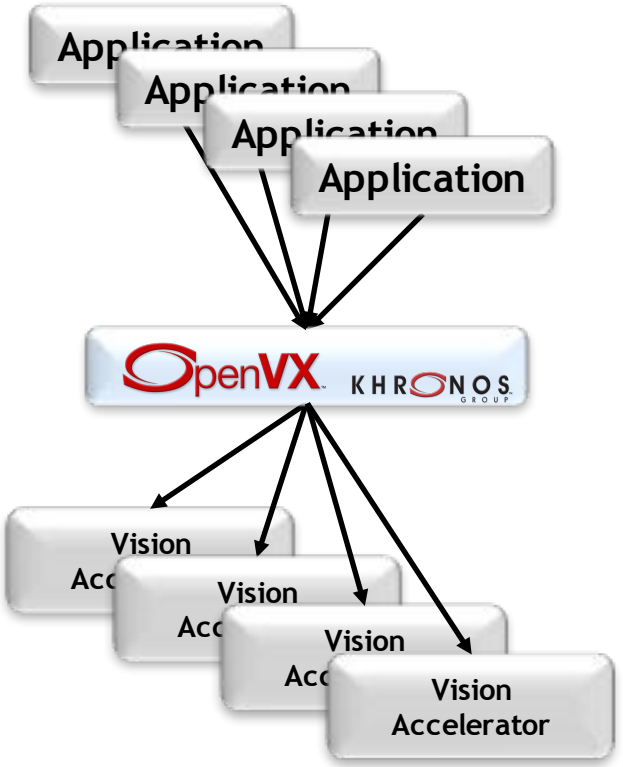
Over 100 companies defining royalty-free APIs to connect software to silicon

Sensor Processing
- Vision Acceleration
- Camera Control
- Sensor Fusion

Acceleration in HTML5
- 3D in browser – no Plug-in
- Heterogeneous computing for JavaScript

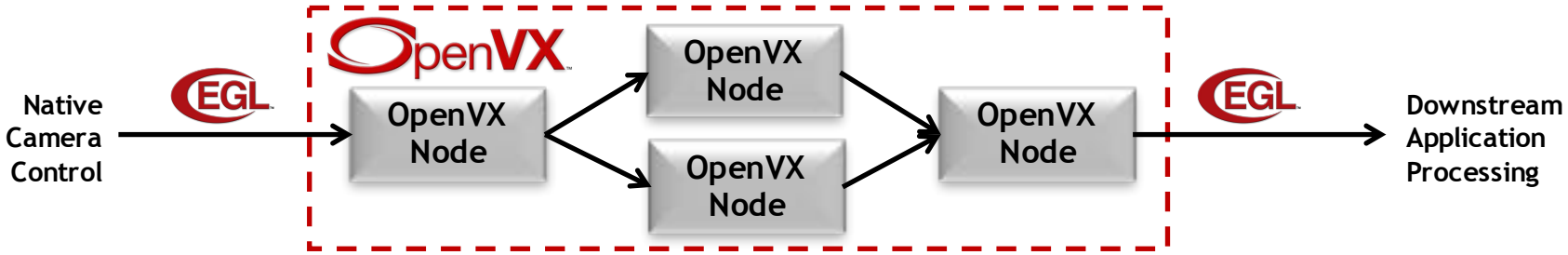
OpenVX - Power Efficient Vision Acceleration

- **Out-of-the-Box vision acceleration framework**
  - Enables low-power, real-time applications
  - Targeted at mobile and embedded platforms
- **Functional Portability**
  - Tightly defined specification
  - Full conformance tests
- **Performance portability across diverse HW**
  - Higher-level abstraction hides hardware details
    - ISPs, Dedicated hardware, DSPs and DSP arrays, GPUs, Multi-core CPUs …
- **Enables low-power, always-on acceleration**
  - Can run solely on dedicated vision hardware
  - Does not require full SOC CPU/GPU complex to be powered on



OpenVX Graphs - The Key to Efficiency

- **Vision processing directed graphs for power and performance efficiency**
  - Each Node can be implemented in software or accelerated hardware
  - Nodes may be fused by the implementation to eliminate memory transfers
  - Processing can be tiled to keep data entirely in local memory/cache
- **VXU Utility Library for access to single nodes**
  - Easy way to start using OpenVX by calling each node independently
- **Opaque memory model for images**
  - Graph manager can move images between host memory and accelerator memory
  - Virtual images (users cannot see data)



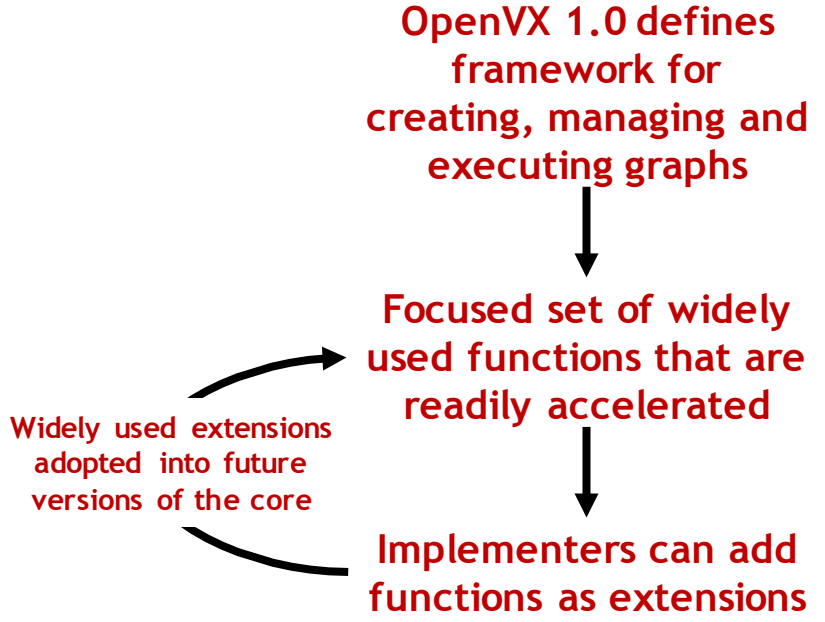
Example OpenVX Graph

OpenVX 1.0 Function Overview

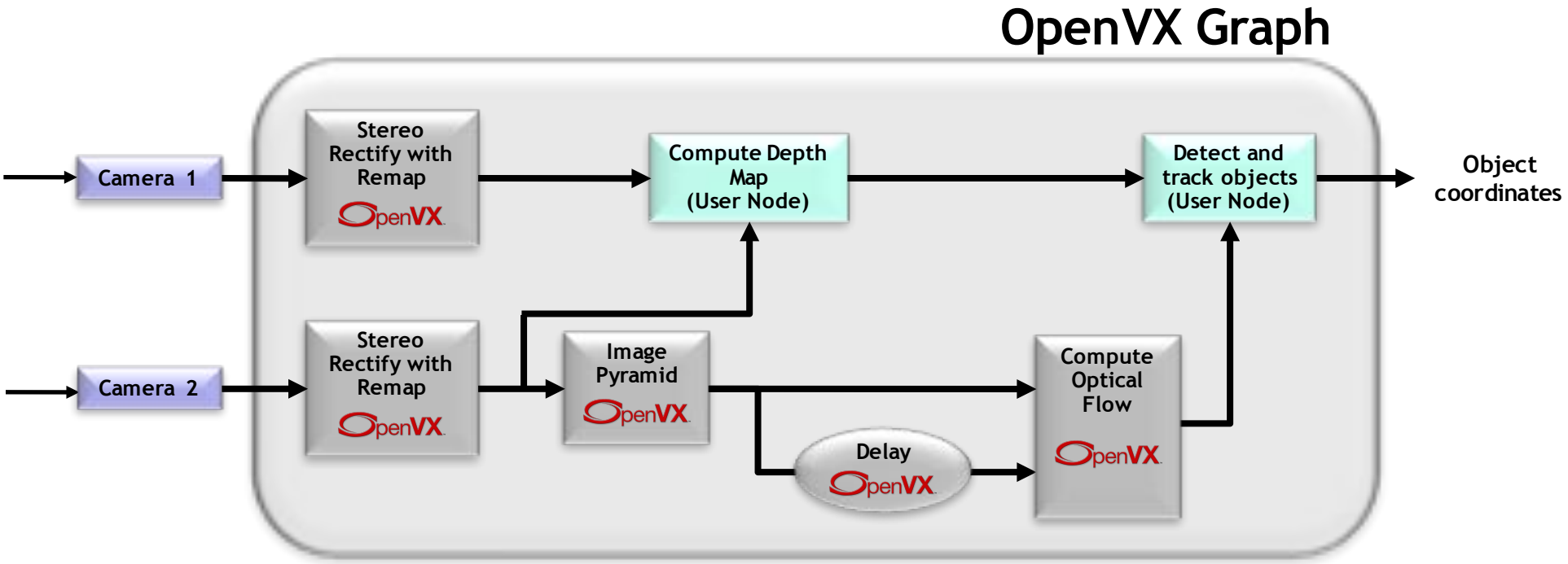
- **Core data structures**
  - Images and Image Pyramids
  - Processing Graphs, Kernels, Parameters
- **Image Processing**
  - Arithmetic, Logical, and statistical operations
  - Multichannel Color and BitDepth Extraction and Conversion
  - 2D Filtering and Morphological operations
  - Image Resizing and Warping
- **Core Computer Vision**
  - Pyramid computation
  - Integral Image computation
- **Feature Extraction and Tracking**
  - Histogram Computation and Equalization
  - Canny Edge Detection
  - Harris and FAST Corner detection
  - Sparse Optical Flow

OpenVX Specification Is Extensible

Khronos maintains extension registry



Example Graph - Stereo Machine Vision



Tiling extension enables user nodes (extensions) to also optimally run in local memory

OpenVX standard

Specification document

- Defines the API
- Available to all in html and pdf from https://www.khronos.org/registry/vx/
- A conformant implementation has to implement all the functions defined by the spec
- Extensions
  - Optional for implementation

Conformance tests

- Are used to enforce functional portability
- Available to adopters ($20K Adopters fee, $15K for members)
- Test suite exercises graph framework and functionality of each OpenVX 1.0 node
- OpenVX group reviews submitted results
- Approved Conformant implementations can use the OpenVX trademark

Sample implementation

- Open source implementation of OpenVX
- Available from https://www.khronos.org/registry/vx/
- Conformant OpenVX implementation
- Was created to help create optimized OpenVX implementations

OpenVX

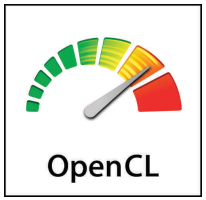
- OpenVX 1.1 specification released May 2016
  - www.khronos.org/openvx
- Full conformance test suite and Adopters Program immediately available
- Multiple companies have announced the release of OpenVX implementation, including Vivante, NVIDIA, Synopsys, Imagination, AMD
  - Many more to come!

OpenVX and OpenCV are Complementary

	OpenCV	OpenVX
Governance	Community driven open source with no formal specification	Formal specification defined and implemented by hardware vendors
Conformance	No conformance tests for consistency and every vendor implements different subset	Full conformance test suite / process creates a reliable acceleration platform
Portability	APIs can vary depending on processor	Hardware abstracted for portability
Scope	Very wide 1000s of imaging and vision functions Multiple camera APIs/interfaces	Tight focus on hardware accelerated functions for mobile vision Use external camera API
Efficiency	Memory-based architecture Each operation reads and writes memory	Graph-based execution Optimizable computation, data transfer
Use Case	Rapid experimentation	Production development & deployment

Khronos APIs for Vision Processing

GPU Compute Shaders (OpenGL 4.X and OpenGL ES 3.1)
Pervasively available on almost any mobile device or OS
Easy integration into graphics apps – no vision/compute API interop needed
Program in GLSL not C
Limited to acceleration on a single GPU



General Purpose Heterogeneous Programming Framework
Flexible, low-level access to any devices with OpenCL compiler
Single programming and run-time framework for CPUs, GPUs, DSPs, hardware
Open standard for any device or OS – being used as backed by many languages and frameworks
Needs full compiler stack and IEEE precision

Out of the Box Vision Framework - Operators and graph framework library
Can run some or all modes on dedicated hardware – no compiler needed
Higher-level abstraction means easier performance portability to diverse hardware
Graph optimization opens up possibility of low-power, always-on vision acceleration
Fixed set of operators – but can be extended

It is possible to use OpenCL or GLSL to build OpenVX Nodes on programmable devices!

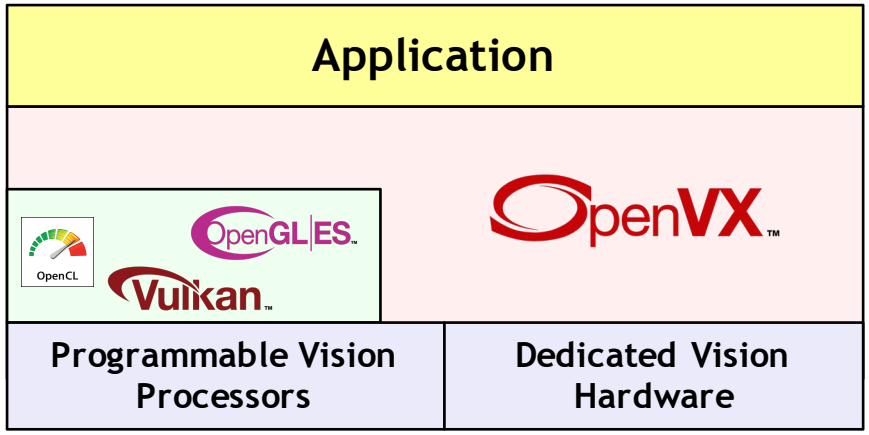
OpenVX and OpenCL are Complementary

		
Use Case	General Heterogeneous programming	Domain targeted vision processing
Ease of Use	General-purpose math libraries with no built-in vision functions	Fully implemented vision operators and framework 'out of the box'
Architecture	Language-based – needs online compilation	Library-based - no online compiler required
Target Hardware	'Exposed' architected memory model – can impact performance portability	Abstracted node and memory model - diverse implementations can be optimized for power and performance
Precision	Full IEEE floating point mandated	Requires minimal floating point or fixed point–optimized for vision operators

Layered Vision Processing Architecture

- **The higher abstraction level of OpenVX protects app from hardware differences**
  - Enables low-power, always-on acceleration – with application portability
- **OpenVX is the Khronos vision API that does not NEED a CPU complex to be turned on during execution**
  - Can use any processor for node execution – including hardware blocks
  - Can be configured to respond to incoming images with no host interaction

Implementers may choose to use OpenCL or OpenGL Compute Shaders to *implement* OpenVX nodes



And then use OpenVX to enable a developer to easily *connect* those nodes into a graph

OpenVX 1.0 extensions

- **Tiling extension: more efficient processing of graphs with user nodes**
  - Provisional spec released
- **XML Schema extension: cross-platform graph saving and loading**
  - Provisional spec released

Summary

- **Khronos is building a trio of interoperating APIs for portable / power-efficient vision and sensor processing**
- **OpenVX 1.0.1 specification is now finalized and released**
  - Full conformance tests and Adopters program immediately available
  - First commercial implementation released
- **Any company is welcome to join Khronos to influence the direction of mobile and embedded vision processing!**
  - $15K annual membership fee for access to all Khronos API working groups
  - Well-defined IP framework protects your IP and conformant implementations
- **More Information**
  - www.khronos.org
  - ntrevett@nvidia.com
  - @neilt3d

KHRONOS GROUP

Summary

- **Exploding amount of computer vision applications running on mobile and embedded devices**
- **There is a need for an efficient software stack**
  - OpenCV will help in the short term
  - OpenVX will simplify the problem in the long run
- **Algorithm should be designed to work on embedded platforms from the very beginning**
  - With functional safety requirements

itseez

Sample models

itseez

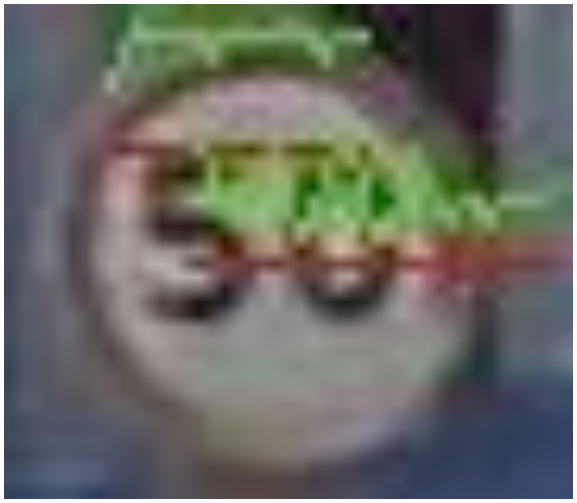
Traffic sign recognition: Existing approaches

- **Color segmentation**
- **Grey-scale segmentation**
- **Shape fitting**
- **Viola-Jones cascade approach**
- **OCR**
  - Text localization
  - Neumann, Matas, CVPR 2012

itseez

Can we detect a scanline that crosses a sign?

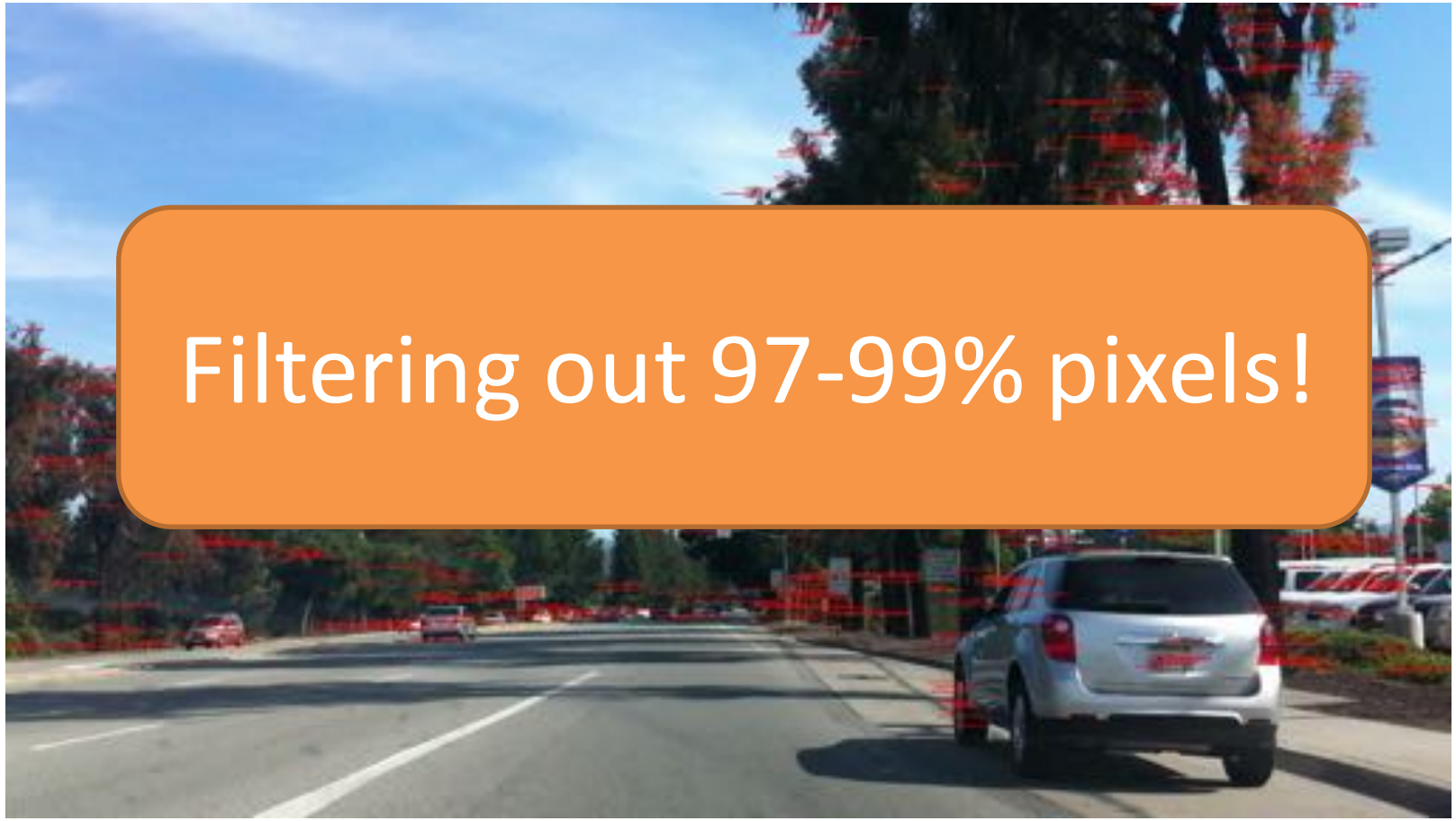


itseez

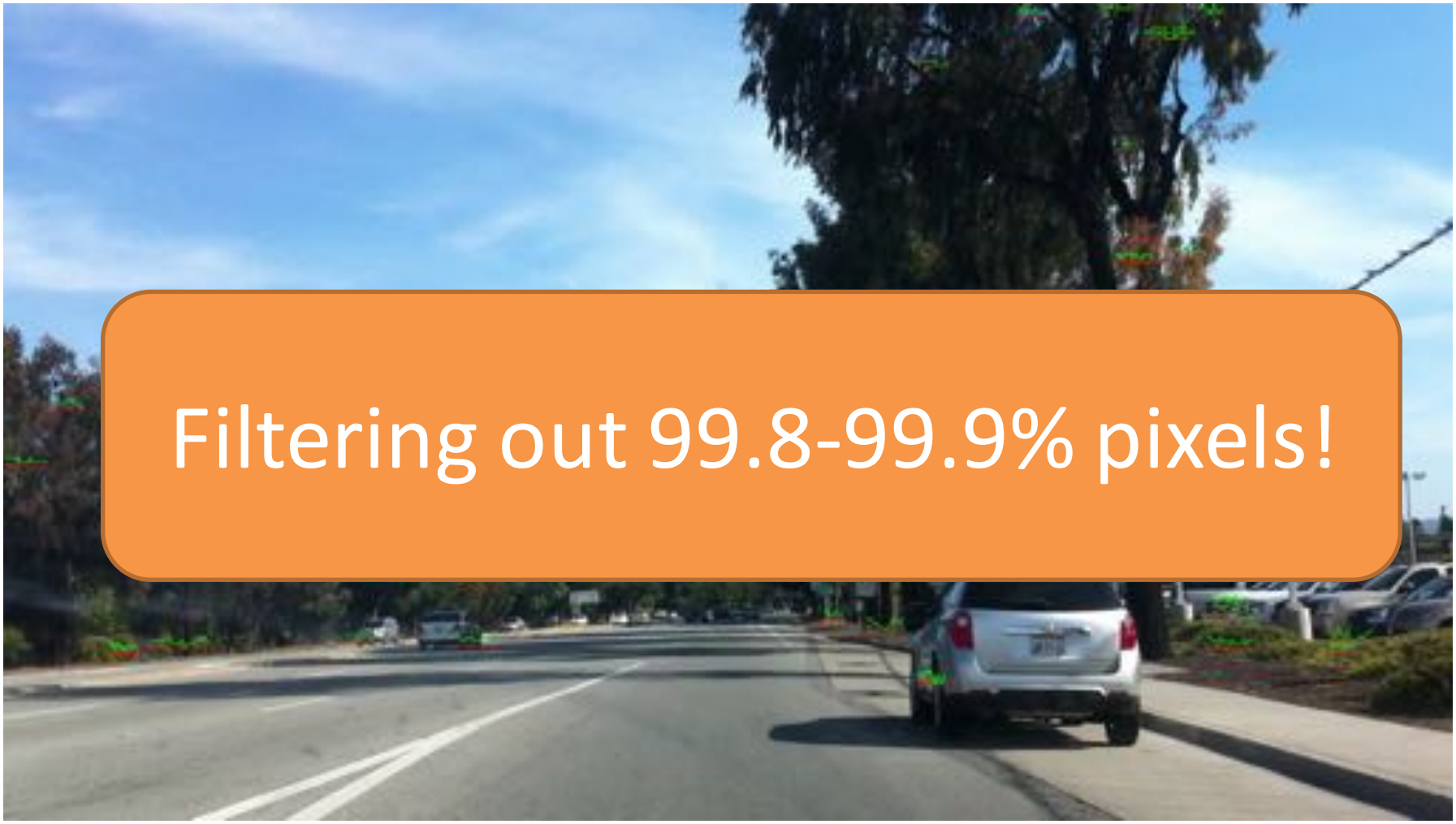
Algorithm: W filter

Detect a W sequence of extremums



itseez

W + Classification filtering



itseez

Now, we have to analyze ROI

itseez