Optimization and Acceleration for OpenCV-based Embedded Vision Applications

Bo Wu, *Ph.D.*

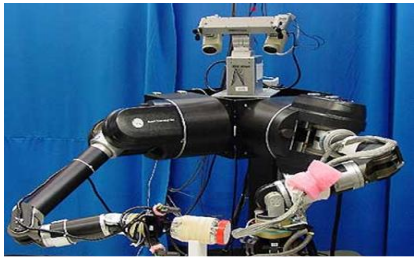
Technical Marketing Manager

Agenda

- OpenCV for Embedded Vision Algorithm Development
- Vision Algorithm Optimization on a Processor
- Vision Algorithm Acceleration using ASIPs
- A Case Study: Canny Edge Detection
- Summary

Vision Processing Goes Embedded

Advanced Driver Assistance Systems



Industrial Automation

Augmented reality

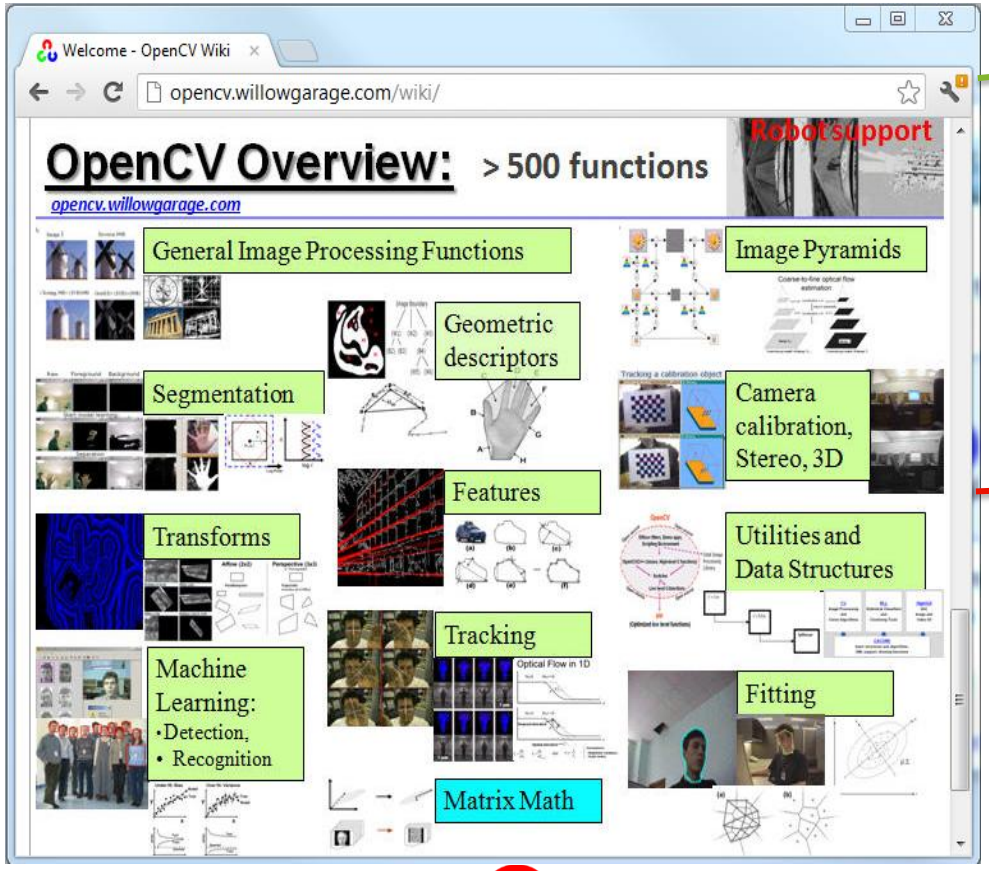
Gesture Control

Unmanned Autonomous Vehicles

Requirements

- Sophisticated and Diverse DSP algorithms
- Specialized I/O: cameras, lighting, stereo
- High Performance, Low Power and Programmability!

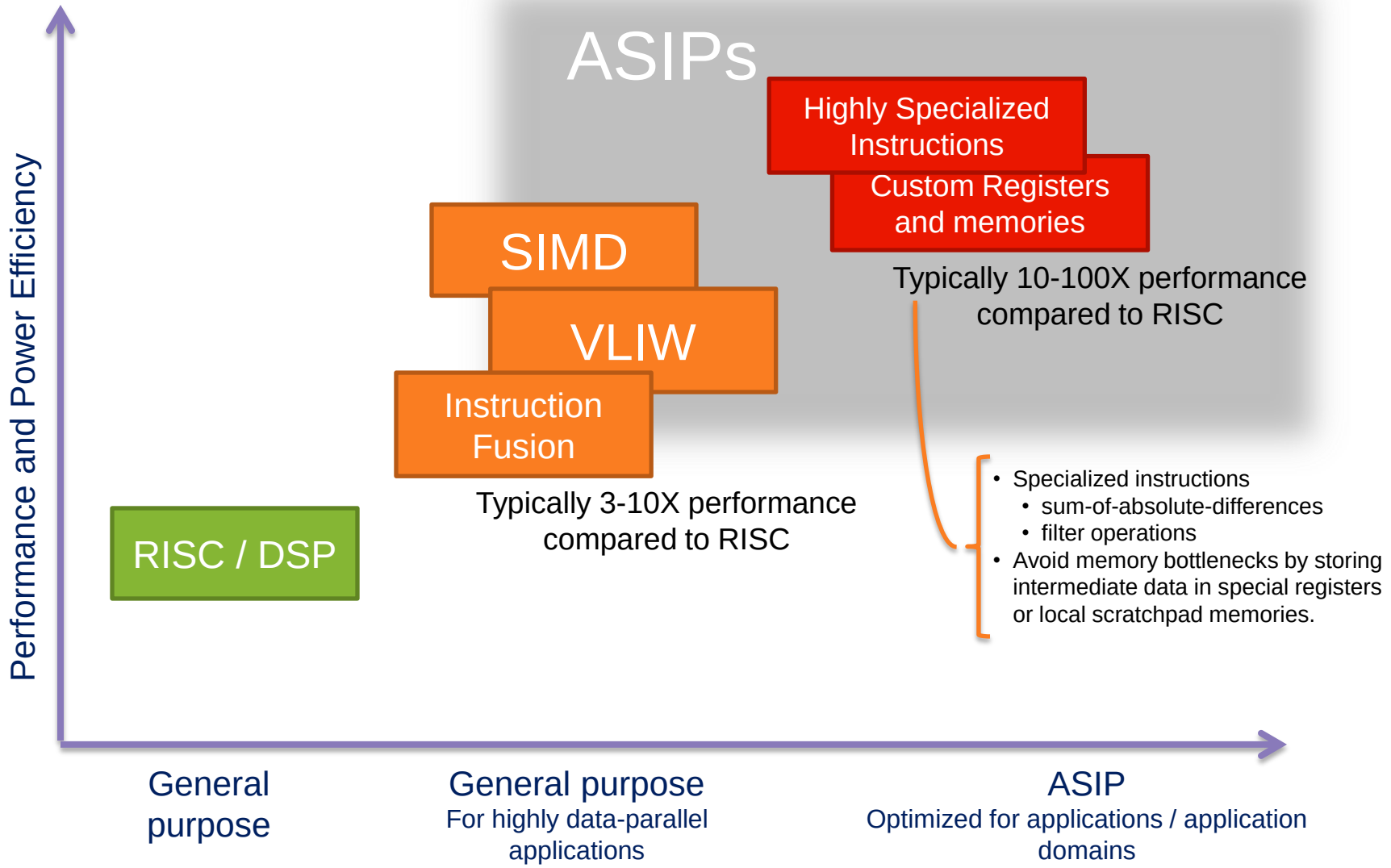
OpenCV for Vision Algorithms



- Very rich library with >500 algorithms for Computer Vision
- BSD-style open source license
- Used widely for computer vision applications

- Difficult to implement in Embedded Applications
  - Requires C++ compiler
  - Optimized for IA32 + high-end GPU, not for embedded RISC or DSP
  - Requires Floating Point
- Difficult to meet performance and power requirements

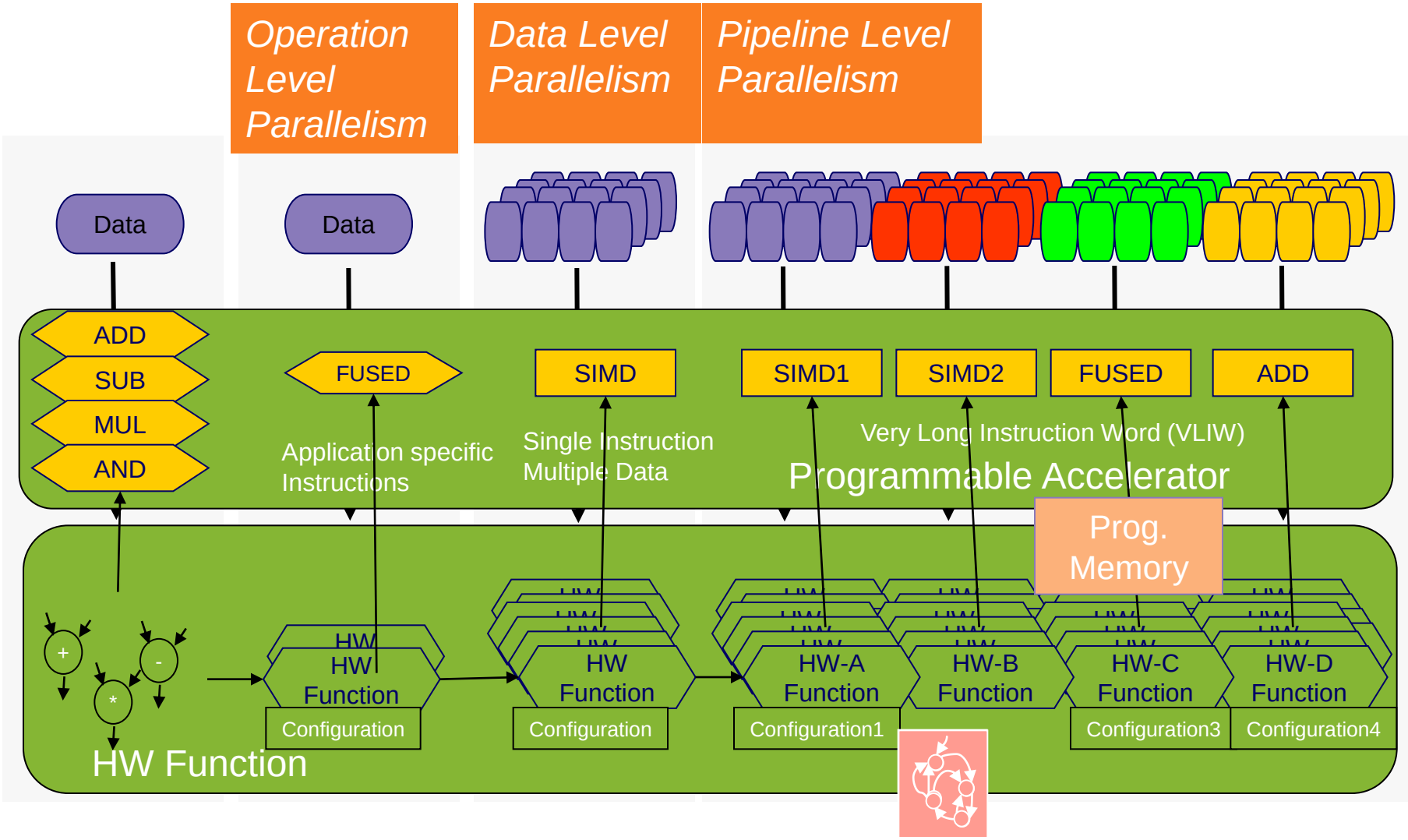
Vision Applications on Processors



Vision Algorithm-level Optimizations

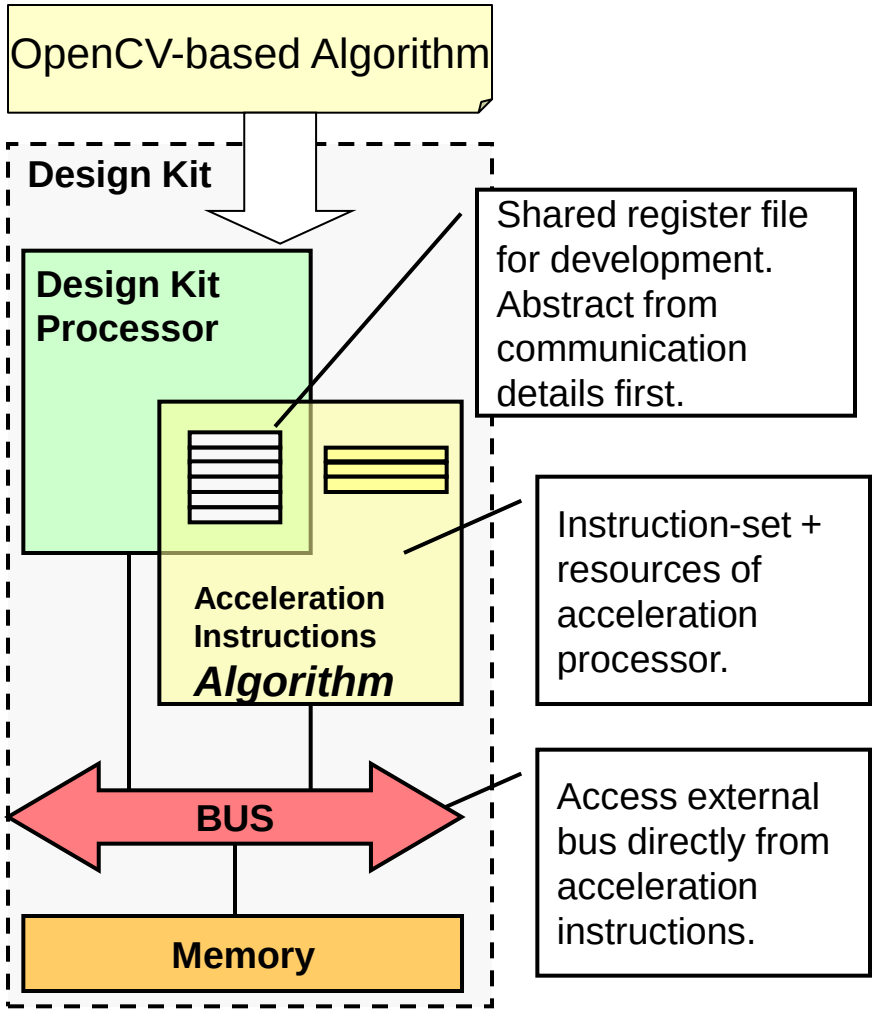
- Pick and choose the algorithmic elements to fit the characteristics of a given application
  - Understand the fundamental of algorithms
  - Understand the requirement of the applications
  - Match the algorithms to applications
- Program level optimization, e.g.,
  - Replace 2D separable filter with two 1D filters
  - Adapt floating point operations to fixed point operations
  - Loop unrolling / folding
  - Memory alignment and reduction

Vision Acceleration Utilizing Parallelism

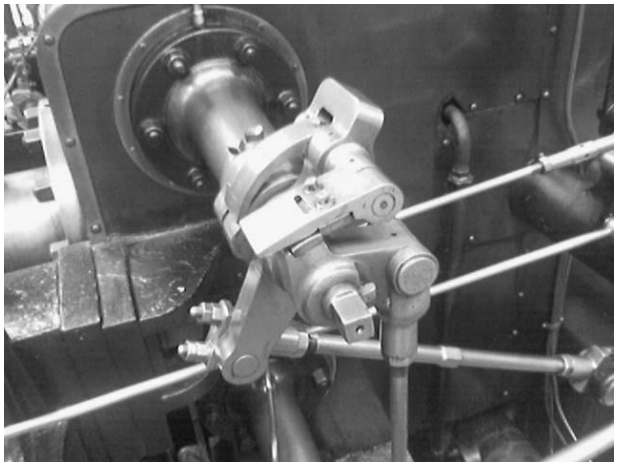


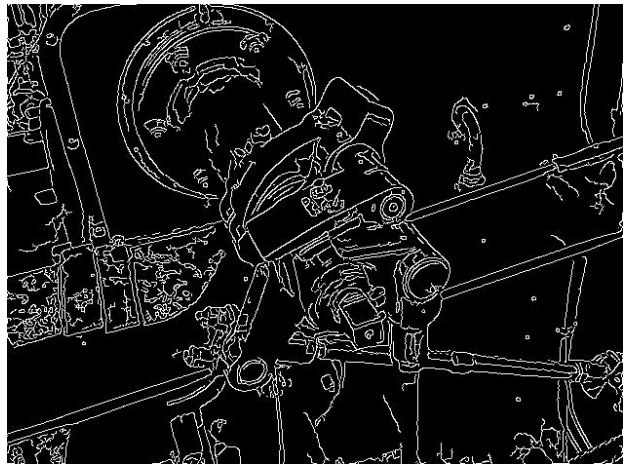
Vision Acceleration Flow using ASIP

- Compile, run and analyze OpenCV based algorithm on a embedded Processor
- Move time critical code from software into hardware by
  - Add necessary registers or local memories
  - Add acceleration instructions
  - Call new instructions from the software
  - Run and test the modified algorithm



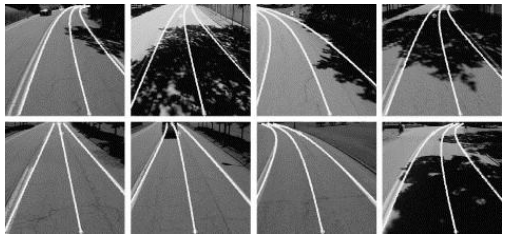
A Case Study: Canny Edge Detection



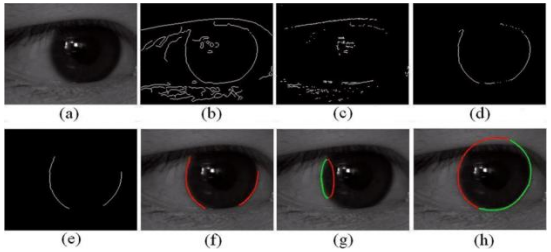


Canny Edge Detection: Robust Edge Detection in Greyscale Image

Example Applications in Vision:

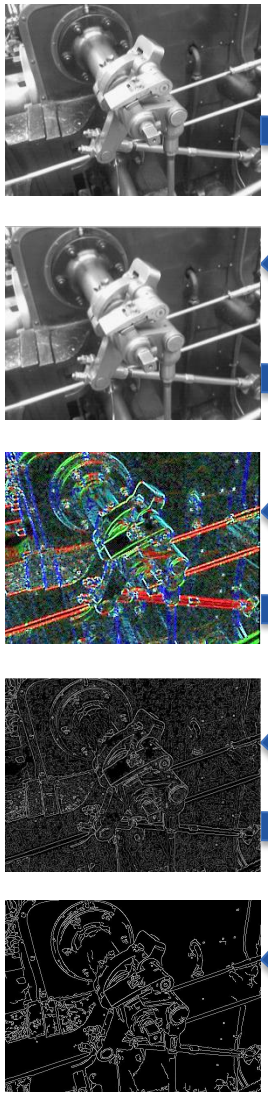


Lane detection



Iris-scan – iris detection

Canny Edge Detection: the Algorithm



Gaussian Blur

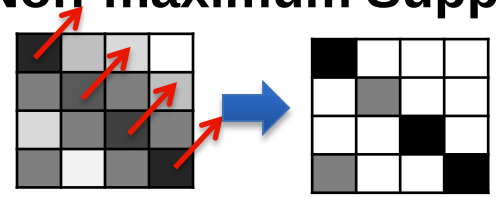
Apply two-dimensional Gauss filter: $\frac{1}{\quad}\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$

Gradient Computation

Apply Sobel Operators: $\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$ and $\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$

and compute magnitude and sector:

Non-maximum Suppression

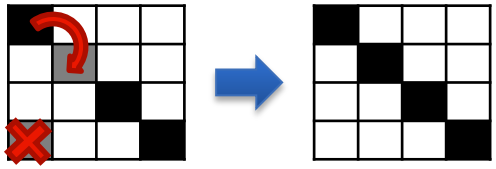


Only retain pixels with highest gradient.

Mark: - “Maybe is edge” ■

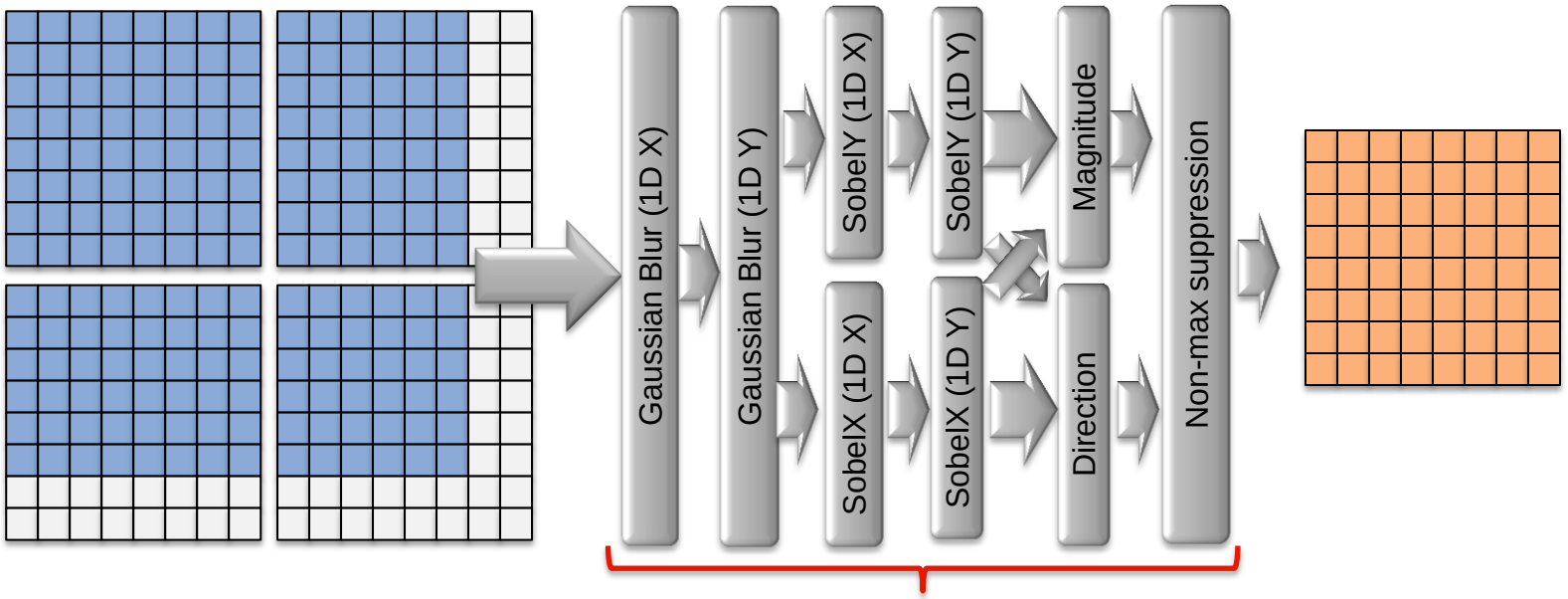
- “Certainly is edge” ■

Thresholding



Trace along the ■-pixels staring from ■-pixels

ASIP: Pipelined Execution of Canny Edge Detection



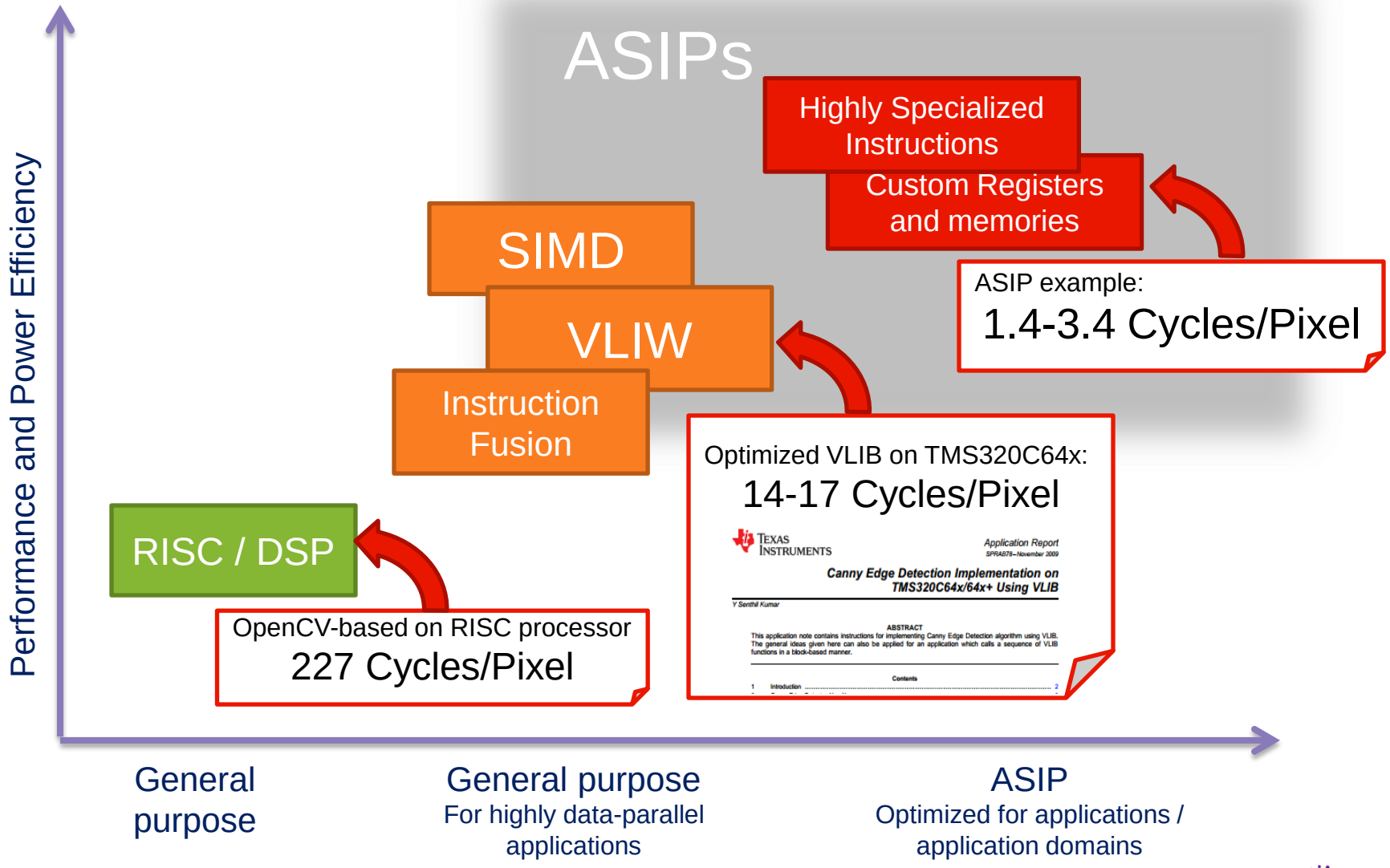
One **Canny** Instruction activates 9 operations in 6 stages.

Assembly code for the ASIP:

```
loadPixels8 p0, [r1], r3          Load 8-lines of pixel data at address r1 with stride r3 in p0
loadPixels8 p1, [r1,8], r3        Load 8-lines of pixel data at address r1 + 8  with stride r3 in p1
loadPixels6 p4, [r2], r3          Load 2-lines of pixel data at address r2 with stride r3 in p4
loadPixels6 p5, [r2,8], r3        Load 2-lines of pixel data at address r2 + 9  with stride r3 in p4

CannyNonMax p3, ( p0, p1, p4, p5 )  Performs all on 16x16 image in p0,p1,p4,p5 and store in p3
```

Results for Canny Edge Detection



Summary

- OpenCV library is a good starting point for embedded vision algorithm development
- Further optimization and acceleration are required for implementation of the vision algorithm on an embedded processor
- Application specific instruction-set processors (ASIPs) can achieve the balance between performance, complexity, and power
- Structured Design Methodology and Tools for ASIPs simplify the design effort