*The most trusted source of analysis, advice, and engineering for embedded processing technology and applications*


# *Implementing Vision Capabilities in Embedded Systems*

*Presented at the*
*2011 Embedded Systems Conference Silicon Valley*

Berkeley Design Technology, Inc.
Oakland, California USA
+1 (510) 451-1800

info@BDTI.com
http://www.BDTI.com


© 2011 Berkeley Design Technology, Inc.

# INTRODUCTION

© 2011 Berkeley Design Technology, Inc

# What is BDTI?

BDTI is a group of engineers dedicated to helping the electronics industry effectively use embedded digital signal processing technology

BDTI performs hands-on, independent benchmarking and evaluation of chips, tools, algorithms, and other technologies

BDTI helps system designers implement their products through specialized engineering services

BDTI offers a wealth of free information for engineers


© 2011 Berkeley Design Technology, Inc

# What is Embedded Vision?

- "Embedded vision" refers to embedded systems that extract meaning from visual inputs
  - Embedded vision is distinct from multimedia
- Emerging high-volume embedded vision markets include automotive safety, surveillance, and gaming
  - The Xbox Kinect is the fastest-selling CE device to date: 10 million units in 4 months


$130 including game

$920 installed

$300 + $6/month

© 2011 Berkeley Design Technology, Inc

# Why is Embedded Vision proliferating now?

1. It has the potential to create huge value
   - Applications in consumer, medical, automotive, entertainment, retail, industrial, aerospace, …
2. Increasingly, it will be expected
   - As embedded vision becomes common in gaming, consumer electronics, and automotive equipment, consumers will expect it
3. It's now possible
   - Sufficiently powerful, low-cost, energy-efficient processors are now emerging

© 2011 Berkeley Design Technology, Inc

# But, implementing embedded vision is challenging

- It's a whole-system problem
- There is limited experience in building practical solutions
- Embedded systems are often highly constrained in cost, size, and power consumption
- It's very computationally demanding
  - E.g., a 720p optical flow algorithm, optimized for a modern VLIW DSP architecture, consumed about 200 MHz/frame/second ➔ 5 fps @ 1 GHz
  - Many vision functions will require highly parallel or specialized hardware
  - Algorithms are diverse and dynamic, so fixed-function compute engines are less attractive

© 2011 Berkeley Design Technology, Inc

# Objectives of This Presentation

- Introduce embedded computer vision: applications, algorithms
- Highlight challenges of implementing computer vision in embedded systems
- Provide an overview of processor options and associated trade-offs
- Introduce application development tools and techniques
- Provide pointers to resources for digging deeper

© 2011 Berkeley Design Technology, Inc

# Scope of This Presentation

- Introduction to embedded vision
- Example embedded vision applications
- Example embedded vision algorithms
- Processor types for embedded vision
- Tools and techniques for embedded vision

© 2011 Berkeley Design Technology, Inc

# APPLICATIONS

© 2011 Berkeley Design Technology, Inc

# Applications: Introduction

- Applications of embedded vision are numerous and diverse
- They span almost every major electronic equipment market, including consumer, entertainment, automotive, industrial, security, medical, and aerospace
- In this section we'll briefly look at a few representative low-cost applications
- It can be useful to consider the functionality required as distinct from the system and the market
  - Similar functionality may be useful in a variety of systems targeting different markets
  - E.g., gesture-based user interfaces can be useful in smartphones, point-of-sale terminals, industrial equipment, medical devices

© 2011 Berkeley Design Technology, Inc

# Application: Surveillance

- In the U.S., retail theft alone amounts to ~$40 billion per year
- With growing concerns about safety and security, the use of surveillance cameras has exploded in the past 10 years
- The U.K. has led this trend, and has ~1.85 million cameras installed
  - Approximately one camera for every 35 people
  - ~1.85 million cameras generate ~2.5 x $10^9$ minutes of video daily
- It's impossible to manually monitor all of this video
- Studies in the U.K. generally show no significant reduction in crime where cameras are installed
- "Smart" surveillance cameras use vision techniques to look for specific kinds of events
- Intelligence can be in the camera, in a local server, or in the cloud
- Key challenge: accuracy with diverse environments and requirements

Cernium Archerfish Solo

© 2011 Berkeley Design Technology, Inc

# Application: Automotive Safety

- ~1.2 million people are killed in vehicle accidents annually
- ~65 million new vehicles are produced annually
- Vision-based safety systems aim to reduce accidents by:
  - Warning when closing in too fast on vehicle ahead
  - Warning of a pedestrian or cyclist in path of vehicle
  - Warning of unintentional lane departure
  - Preventing spoofing of drunk-driving prevention systems
  - Alerting driver when drowsiness impacts attention
  - Automatically dimming high-beams
- Most systems are passive: alert the driver
  - A few apply the brakes
- Some systems augment vision with radar
- Key challenge: accuracy across diverse situations (weather, glare, groups of people, …)

Mobileye C2-270

© 2011 Berkeley Design Technology, Inc

# Application: Video Games

- Video games (hardware and software) are a ~$60 billion/year business
- Vision-based control of video games enables new types of games and new types of users
- The Microsoft Kinect is the fastest-selling consumer electronics product ever: ~10 million units sold in first six months
- Price: $130 (includes a game title); bill of materials cost: ~$60
- Kinect is not just a game controller:
  - Can be used as an audio/video system controller
  - Is being used as a low-cost vision development platform
- Key challenge: must be extremely easy to use and very inexpensive

Microsoft Kinect

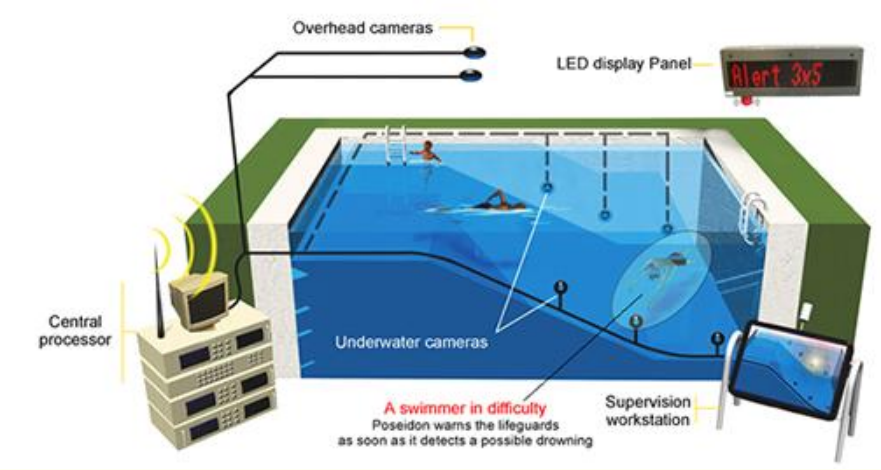

Image courtesy of and © Useit.com

vs.

© 2011 Berkeley Design Technology, Inc

# Application: Swimming Pool Safety

- ~400,000 drowning deaths occur worldwide each year
- In the U.S., drowning is the second-leading cause of accidental death for children 1-14 years old
- 19% of child drowning deaths occur in public pools with certified lifeguards present
- A person drowning is unable to call for help
- The Poseidon system from MG International monitors swimmers and alerts lifeguards to swimmers in distress





Images courtesy of and © MG INTERNATIONAL - POSEIDON

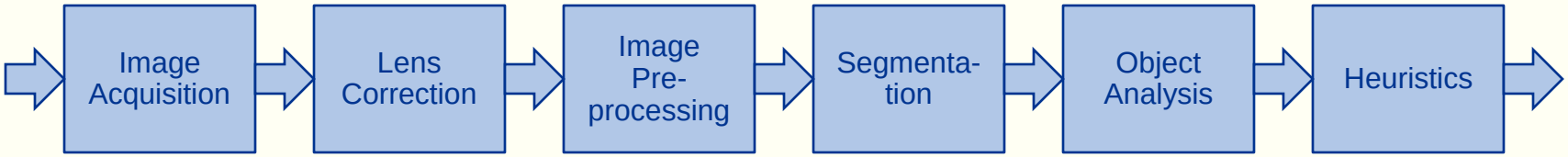
© 2011 Berkeley Design Technology, Inc

# ALGORITHMS

© 2011 Berkeley Design Technology, Inc

# How does Embedded Vision work?

A typical embedded vision pipeline:




Typical total compute load for VGA 30 fps processing:
~3 billion DSP instructions/second

Loads can vary dramatically with pixel rate and algorithm complexity

© 2011 Berkeley Design Technology, Inc

# Lens Distortion Correction
## The Problem

- Lenses (especially inexpensive ones) tend to distort images

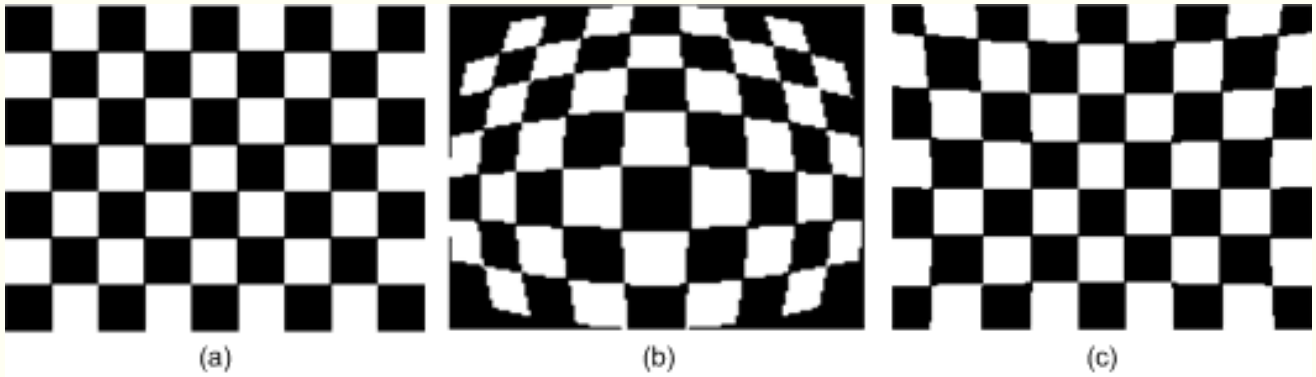



**Figure 2**. Examples of different types of lens distortion. (a) original (b) barrel distortion (c) pincushion distortion

- Straight lines become curves
- Distorted images tend to thwart vision algorithms

Image courtesy of and © Luis Alvarez

Section based on "Lens Distortion Correction" by Shehrzad Qureshi; used with permission.

© 2011 Berkeley Design Technology, Inc

# Lens Distortion Correction
## A Solution

- A typical solution is to use a known test pattern to quantify the lens distortion and generate a set of warping coefficients that enable the distortion to be (approximately) reversed

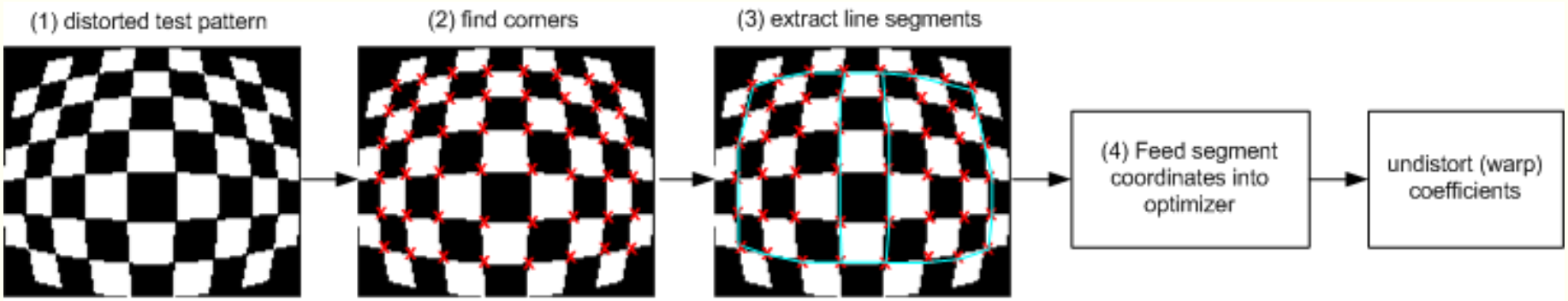



**Figure 3**. Camera calibration procedure

- The good news: the calibration procedure is performed once
- The bad news: the resulting coefficients then must be used to "undistort" (warp) each frame before further processing
- Warping requires interpolating between pixels

© 2011 Berkeley Design Technology, Inc

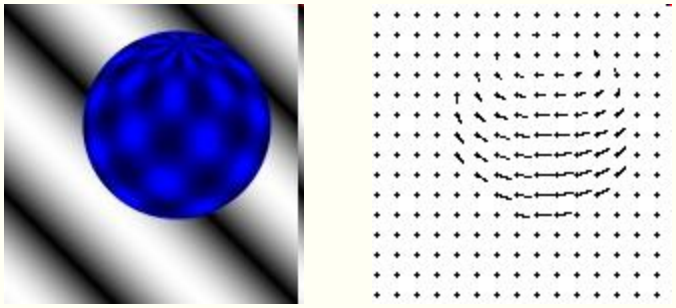
# Lens Distortion Correction
## Challenges and Trade-offs

- Lens distortion is a well-studied phenomenon, and robust distortion correction solutions exist
  - E.g., http://www.ipol.im/pub/algo/ags_algebraic_lens_distortion_estimation/
- Warping is very computationally intensive
  - Each color component of each pixel requires a calculation
    - E.g., 720p 60 fps: 921,600 pixels × 3 color components × 60 fps ➔ 166 million data elements per second
    - If warping each data element requires 10 math operations (e.g., for bilinear interpolation) ➔ 1.66 GOPS
  - However, warping is readily parallelizable
- There is a trade-off between the quality of the distortion correction and the computation load

© 2011 Berkeley Design Technology, Inc

# Dense Optical Flow
## The Problem

- Estimate the pattern of apparent motion of objects, surfaces, and edges in a visual scene
- Typically results in a motion vector for each pixel position in a video frame



Rotating sphere and corresponding optical flow field
(Images from http://of-eval.sourceforge.net/)

Used in vision applications

- To estimate observer and object positions and motion in 3D space
- To estimate image registration for super-resolution and noise reduction algorithms

© 2011 Berkeley Design Technology, Inc

# Dense Optical Flow
## Challenges and Tradeoffs

- Optical flow can't be computed without making some assumptions about the video content (this is known as the *aperture problem*)
- Different algorithms make different assumptions
  - E.g. constant illumination, smooth motion
- Many algorithms exist, roughly divided into the following classes:
  - **Block-based methods** (similar to motion estimation in video compression codecs)
  - **Differential methods** (Lucas-Kanade, Horn-Schunck, Buxton-Buxton, and variations)
  - **Other methods** (Discrete optimization, phase correlation)
- Aliasing can occur
  - E.g. when an object in the scene has a repeating texture pattern, or when motion exceeds algorithmic constraints
- Some algorithms are sensitive to camera noise
- Most algorithms are computationally intensive

© 2011 Berkeley Design Technology, Inc

# Dense Optical Flow
## A Solution

Lucas-Kanade method with image pyramid is a popular solution

- Lucas-Kanade method is a differential method of estimating optical flow; it is simple but has significant limitations
  - Assumes constant illumination and constant motion in a small neighborhood around the pixel-position of interest
  - Limited to very small velocity vectors (less than one pixel per frame)
- Image pyramids extend Lucas-Kanade to support greater motion
  - Original frames are sub-sampled to create several pyramid levels
  - Lucas-Kanade method is used at the top level (lowest resolution) yielding a coarse estimate, but supporting greater motion
  - Lucas-Kanade is used again at lower levels (higher resolution) to refine the optical flow estimate

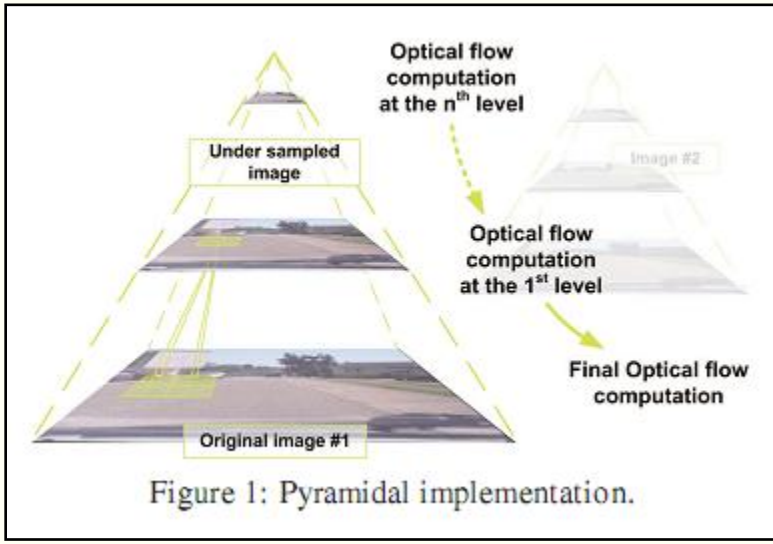


Figure 1: Pyramidal implementation.

Figure courtesy of and © Julien Marzat

© 2011 Berkeley Design Technology, Inc

# Pedestrian Detection
## The Problem

- Surveillance/monitoring applications
  - Unauthorized access to restricted areas
  - People tracking in a given area
- Automotive safety applications
  - Pedestrian detection in the path of the vehicle
  - Must distinguish between pedestrians and other moving objects
- High accuracy is required
  - To avoid missed alarms (under-sensitive detector)
  - To avoid false alarms (over-sensitive detector)
- Real-time processing
  - Low latency is required (quick response is desired)

© 2011 Berkeley Design Technology, Inc

# Pedestrian Detection
## Example

- Detecting pedestrians at a stop sign


© 2011 Berkeley Design Technology, Inc

# Pedestrian Detection
## Challenges and Tradeoffs (1)

- Challenges:
  - Detecting the relative motion of pedestrians against a moving background
  - Pedestrians come in many sizes, shapes, and costumes
  - Pedestrians sometimes move in erratic ways
  - Pedestrians frequently travel in groups
  - The camera's view of a pedestrian may become occluded
  - Computationally intensive: can reach hundreds of GOPS

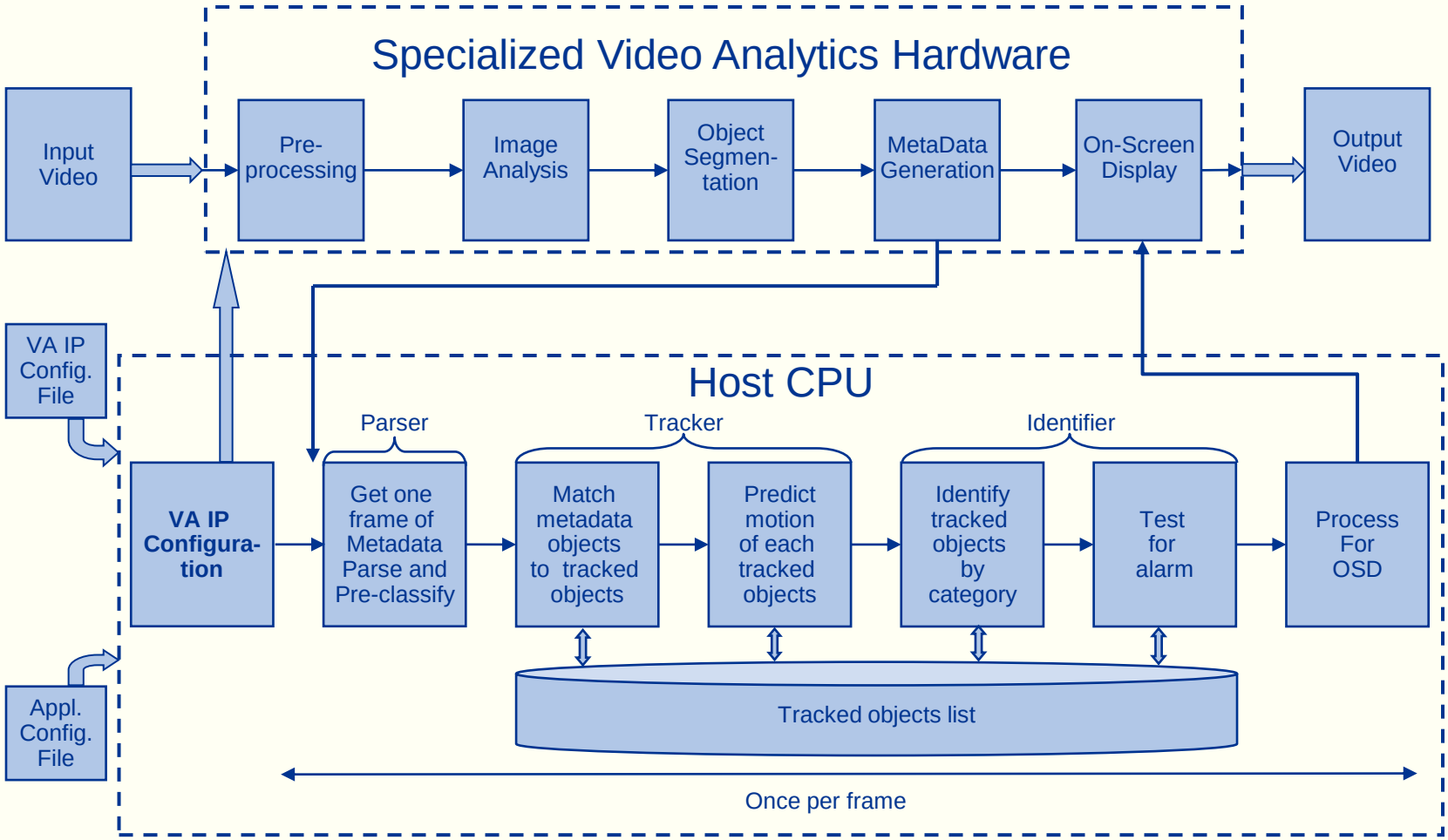
© 2011 Berkeley Design Technology, Inc

# Pedestrian Detection
## Challenges and Trade-offs (2)

- Trade-offs:
  - Fixed camera view; limiting application scope for less computation
  - Detection based on motion of vertical edges rather than colors; limiting identification and tracking capabilities for less computation
  - Object classification based on aspect ratio; identification of individuals rather than groups, thus filtering out non-pedestrian size/shape moving objects
  - Tracking based on confidence level; improved tracking of occluded objects at the expense of detection latency. This also compensates for some of the erratic human behavior such as sudden stops

© 2011 Berkeley Design Technology, Inc

# Pedestrian Detection
## A Solution




© 2011 Berkeley Design Technology, Inc

# PROCESSORS

© 2011 Berkeley Design Technology, Inc

# The Processing Challenge

Embedded vision applications typically require:

- Very high performance
- Programmability
- Low cost
- Energy efficiency

Achieving all of these together is difficult

- Dedicated logic yields high performance at low cost, but with little programmability
- General-purpose CPUs provide programmability, but with weak performance or poor cost-, energy-efficiency

© 2011 Berkeley Design Technology, Inc

# How is Embedded Vision implemented?

Demanding embedded vision applications will most often use a combination of processing elements (similar to wireless baseband chips), e.g.:

- CPU for complex decision-making, network access, user interface, storage management, overall control
- High-performance DSP-oriented processor for real-time, moderate-rate processing with moderately complex algorithms
- Highly parallel engine(s) for pixel-rate processing with simple algorithms

© 2011 Berkeley Design Technology, Inc

# Processor Types for Embedded Vision

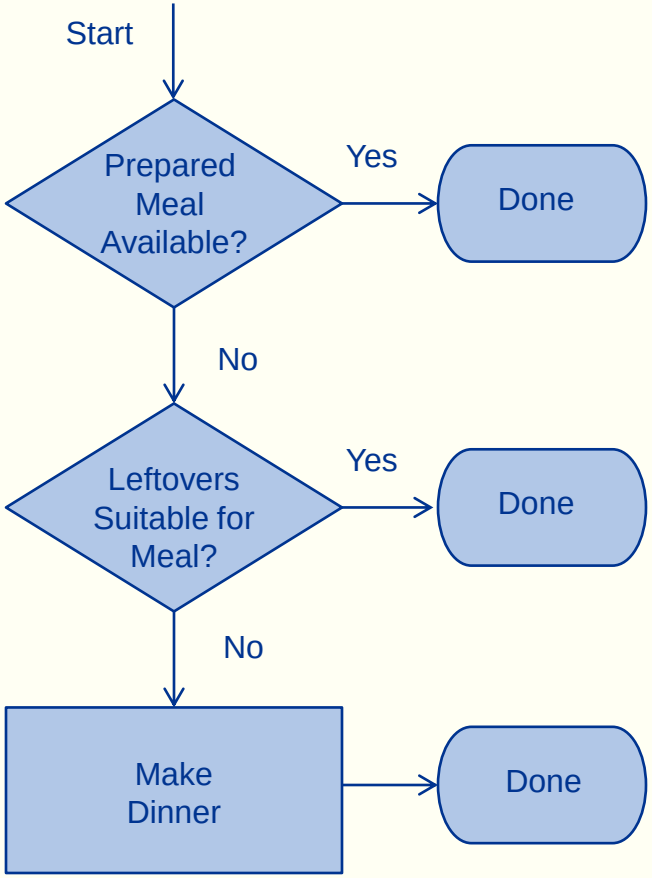
While any processor can in theory be used for embedded vision, the most promising types today are:

- High-performance embedded CPU
- Application-specific standard product (ASSP) + CPU
- Graphics processing unit (GPU) + CPU
- DSP processor + accelerators + CPU
  - Mobile “application processor”
- Field programmable gate array (FPGA) + CPU

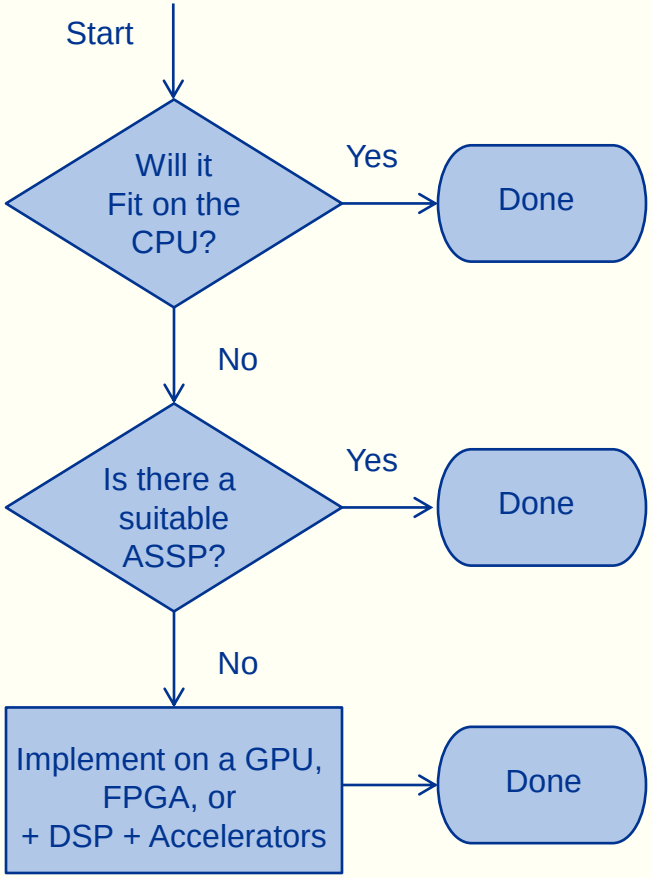
© 2011 Berkeley Design Technology, Inc

# The Path of Least Resistance

## Making Dinner on a Tuesday




## Selecting a Processor for an Embedded Vision Application




© 2011 Berkeley Design Technology, Inc

# High-performance Embedded CPUs

Though challenged with respect to performance and efficiency, unaided high-performance embedded CPUs are attractive for some vision applications

- Vision algorithms are initially developed on PCs with general-purpose CPUs
- CPUs are easiest to use: tools, operating systems, middleware, etc.
- Most systems need a CPU for other tasks

However:

- Performance and/or efficiency is often inadequate
- Memory bandwidth is a common bottleneck

Example: Intel Atom E660T

© 2011 Berkeley Design Technology, Inc

# Application-specific Standard Product + CPU

Application-specific standard products (ASSPs) are specialized, highly integrated chips tailored for specific applications or application sets

- ASSPs may incorporate a CPU, or use a separate CPU chip
- By virtue of specialization, they tend to deliver superior cost- and energy-efficiency
- They usually include strong application-specific software development infrastructure and/or application software

However:

- The specialization may not be right for your particular application
- They may come from small suppliers, which can mean more risk
- They use unique architectures, which can make programming them, and migration to other solutions, more difficult
- Some are not user-programmable

Example: PrimeSense PS1080-A2

© 2011 Berkeley Design Technology, Inc

# Graphics Processing Unit (GPU) + CPU

GPUs, mainly used for 3-d graphics, are increasingly capable of being used for other functions

- Referred to as “general-purpose GPU” or “GPGPU”
  - 👍 Often used for vision algorithm development
  - 👍 Widely available; easy to get started with parallel programming
  - 👍 Well-integrated with CPU (sometimes on one chip)
  - 👎 Typically cannot be purchased as a chip, only as a board, with limited selection of CPUs
  - 👎 Low-cost, low-power GPUs (designed for smart phones, tablets) are not GPGPUs

Example: NVIDIA GT240

© 2011 Berkeley Design Technology, Inc

# DSP Processor + Co-processors + CPU

Digital signal processors ("DSP processors" or "DSPs") are processors specialized for signal processing algorithms

- This makes them more efficient  than CPUs for the kinds of signal processing tasks that are at the heart of vision applications
- DSPs are relatively mature and easy to use compared to other kinds of parallel processors

However:

- DSPs often lack sufficient performance, and aren't as easy to use at CPUs
- Hence, DSPs are often augmented with specialized co-processors and a CPU on the same chip

Example: Texas Instruments DM8168

© 2011 Berkeley Design Technology, Inc

# Mobile “Application Processor”

A mobile “application processor” is a highly integrated system-on-chip, typically designed primarily for smart phones but used for other applications

- Typically comprise a high-performance CPU core and a constellation of specialized co-processors: GPU, VPU, 2-d graphics, image acquisition, etc.
- 👍 Energy efficient
- 👍 Often have strong development support, including low-cost development boards, Linux/Android ports, etc.

However:

- 👎 Specialized co-processors are usually not user-programmable

Example: Freescale i.MX53

© 2011 Berkeley Design Technology, Inc

# Field Programmable Gate Array + CPU

Field programmable gate arrays (FPGAs) are flexible logic chips that can be reconfigured at the gate and block levels

- Enables custom specialization and enormous parallelism
- Enables selection of I/O interfaces and on-chip peripherals

However:

- FPGA design is hardware design, typically done at a low level (RTL) (register transfer level)
- Ease of use improving due to:
  - "IP" block libraries
  - Reference designs
  - Emerging high-level synthesis tools
- Low-performance CPUs can be implemented in the FPGA; higher-performance integrated CPUs coming

Example: Xilinx Spartan-6 LX150T

© 2011 Berkeley Design Technology, Inc

# DEVELOPMENT AND TOOLS

© 2011 Berkeley Design Technology, Inc

# Embedded vision system development challenges

Developing embedded vision systems is challenging

- Vision is a system-level problem: success depends on numerous elements working together, besides the vision algorithms themselves:
  - Lighting, optics, image sensors, image pre-processing, etc.
  - Getting these elements working together requires multi-disciplinary expertise
- There are many computer vision experts who know little about embedded systems, and many embedded system designers who know little about computer vision
  - Many projects die in the chasm between these groups
- There are numerous algorithms available, but picking the best and ensuring that they meet application requirements can be very difficult
- Vision often uses complex, computationally demanding algorithms; implementing these under severe cost, size, and energy constraints requires selecting the right processor for the job
  - Expect to optimize algorithm implementations for the processor

© 2011 Berkeley Design Technology, Inc

# The PC is your friend

- Most embedded vision systems—and virtually all vision algorithms—begin life on a personal computer
- The PC is a fabulous platform for research and prototyping:
  - Ubiquitous
  - Inexpensive
  - Outstanding development infrastructure:
    - Generic software tools, libraries
    - Vision-specific libraries
    - Domain-specific design, simulation tools
    - Example applications
  - Easy to integrate cameras, displays, networks, and other I/O
- One can begin implementing vision applications within a day of unpacking a new PC and webcam
- Parallel acceleration of vision algorithms can be done using GPGPUs

© 2011 Berkeley Design Technology, Inc

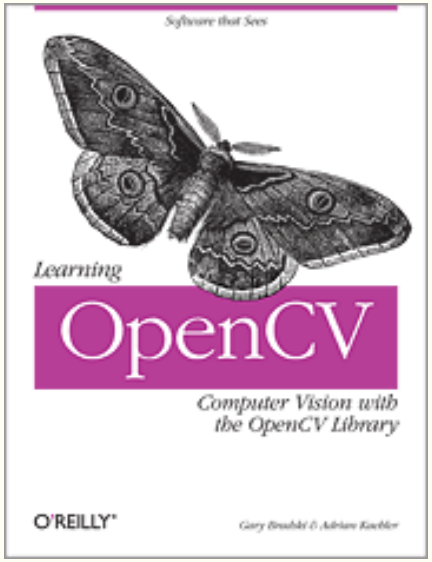
# The PC is your foe

- The PC is not an ideal platform for implementing most embedded vision systems
- Although some applications can embed a PC, many cannot due to cost, size, and power considerations
- PCs lack sufficient performance for many real-time vision applications
  - GPGPUs don't yet address embedded applications
- Many of the same tools and libraries that make it easy to develop vision algorithms and applications on the PC also make it difficult to create efficient embedded implementations
  - Algorithm expert: "Here's my algorithm. It has 99% accuracy"
  - Embedded developer: "How is it coded? How does it perform?"
  - Algorithm expert: "It uses 85 MATLAB functions, 27 OpenCV functions, and double-precision floating-point. It runs at 1/20$^{th}$ of real-time on a 3 GHz quad-core workstation"
  - Embedded developer: "Just shoot me"

© 2011 Berkeley Design Technology, Inc

# OpenCV

- OpenCV is a free, open source computer vision software component library comprising over two thousand algorithms
  - Originally developed by Intel, now maintained by Willow Garage
- The OpenCV library, used along with Bradski and Kahler's book, is a great way to quickly begin experimenting with computer vision
- However:
  - Some OpenCV functions work better than others
  - OpenCV is a library, not a standard
  - OpenCV is not well suited to embedded implementation
- Ports to non-PC platforms have been made, and more are underway, but there's little coherence to these efforts




© 2011 Berkeley Design Technology, Inc

# Some Promising Developments

- Microsoft Kinect
  - The Kinect is becoming very popular for vision development
  - It has been integrated with OpenCV
  - Microsoft will introduce a Kinect SDK for Windows 7
  - It has also been integrated with the Beagle Board embedded development platform
- XIMEA Currera: integrates an embedded PC in a camera
- Several embedded processor vendors have begun to recognize the magnitude of the opportunity for embedded vision
- Smart phones and tablets have the potential to become effective embedded vision platforms
- Application software platforms are emerging for certain EV applications, such as augmented reality and gesture-based UIs


Image courtesy of and © XIMEA

© 2011 Berkeley Design Technology, Inc

# CONCLUSIONS

© 2011 Berkeley Design Technology, Inc

# Conclusions

To date, embedded computer vision has largely been limited to low-profile applications like surveillance and industrial inspection

Thanks to the emergence of high-performance, low-cost, energy efficient programmable processors, this is changing

In the coming years, embedded vision will change our industry

Embedded vision technology will rapidly proliferate into many markets, creating opportunities for chip, equipment, algorithm, and services companies

But implementing embedded vision applications is challenging, and there is limited know-how in industry

© 2011 Berkeley Design Technology, Inc

# Conclusions (cont'd)

Don't go it alone!  Re-use what you can:

- Algorithms
- Cameras
- Software libraries
- Application platforms

Be realistic!

- Recognize that vision is a system-level problem
- Accept that many vision problems are hard; if the application requires perfect vision performance, it may never succeed
- Expect challenges in implementing vision within embedded cost, size, and power budgets

© 2011 Berkeley Design Technology, Inc

# RESOURCES

# Selected Resources

- www.embedded-vision.com (coming in June)
- OpenCV:
  - http://opencv.willowgarage.com/wiki/
  - Bradski and Kaehler, "Learning OpenCV: Computer Vision with the OpenCV Library", O'Reilly, 2008
- MATLAB/Octave:
  - "Machine Vision Toolbox", P.I. Corke, IEEE Robotics and Automation Magazine, 12(4), pp 16-25, November 2005. http://petercorke.com/Machine_Vision_Toolbox.html
  - P. D. Kovesi. "MATLAB and Octave Functions for Computer Vision and Image Processing." Centre for Exploration Targeting, School of Earth and Environment, The University of Western Australia. http://www.csse.uwa.edu.au/~pk/research/matlabfns.
- Visym (beta): http://beta.visym.com/overview

© 2011 Berkeley Design Technology, Inc

# Selected Resources

- "Predator" self-learning object tracking algorithm:
  - Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-Backward Error: Automatic Detection of Tracking Failures," International Conference on Pattern Recognition, 2010, pp. 23-26
  - http://info.ee.surrey.ac.uk/Personal/Z.Kalal/
- Vision on GPUs: GPU4vision project, TU Graz: http://gpu4vision.icg.tugraz.at
- Lens distortion correction:
  - Luis Alvarez, Luis Gomez and J. Rafael Sendra. "Algebraic Lens Distortion Model Estimation." Image Processing On Line, 2010. DOI:10.5201/ipol.2010.ags-alde:

    http://www.ipol.im/pub/algo/ags_algebraic_lens_distortion_estimation/

© 2011 Berkeley Design Technology, Inc

# Resources

The Embedded Vision Alliance is a new industry partnership to facilitate the flow of high-quality information on practical aspects of embedded vision engineering; public launch May 31

BDTI's web site provides a variety of free information on processors used in vision applications.

BDTI's free *InsideDSP* email newsletter covers tools, chips, and other technologies for embedded vision and other DSP applications. Sign up at www.BDTI.com.