Having Fun with OpenCV

Instructor - Simon Lucey

16-423 - Designing Computer Vision Apps

Carnegie Mellon
THE ROBOTICS INSTITUTE

Today

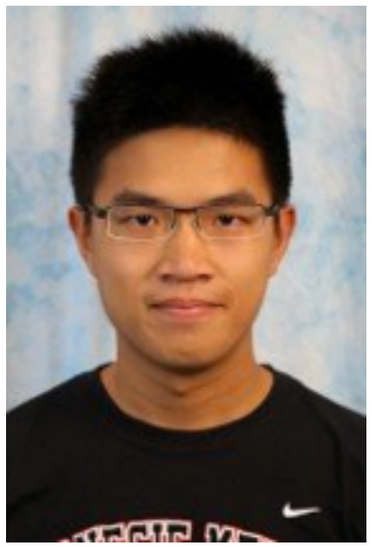
- Course Logistics
- Philosophy to Mobile Computer Vision R&D
- Getting started with OpenCV.

About this Course

- Team

Me

(Instructor)



Chen-Hsuan Lin

(TA)

- Office hours: Tuesday 3:00pm - 4:00pm (or use Piazza)
- Course website has ALL information.
- Questions: Please use Piazza.
- Finding a project partner: Please use Piazza.

Assignments

- There will be 5 assignments - (5 + 10 + 10 + 10 + 10)%
- Each assignment will relate to the topics of the previous lectures, but ALSO take us closer to the task of building our OWN augmented reality app.
- First Assignment will be released on Monday September 8th.
- Assignment is due Friday September 18th. (Tight)
- See course website for full schedule.

Assignments

- Goal is that every assignment takes you a step closer to building your OWN augmented reality app.
- Assignments are designed to take us (step-by-step) towards an augmented reality app.

MidTerm Exam

- Will cover first part of the course.
- Date fixed for November 3rd.
- No substitute date.

- There is no **Final Exam**.

Final Project

- Teams 1-2 (if it is something big we could discuss 3).
- Topic: efficient implementation of CV algorithm on a mobile device.
- Until November 5th,
  - think about a topic
  - find a partner.
- Project Checkpoint November 24th.
  - 2 page latex CVPR style document outlining the goal of the project and background literature.
  - Should also describe why a simplistic application of desktop algorithm would be problematic on a mobile device. How are you going to circumvent it?
  - Or, employ a unique/enhanced sensor on the mobile device (e.g. IMU, high-speed camera, Structure IO depth sensor, etc.).

Background Material

- For the computer vision theory aspect of this course we will be using Simon Prince's new textbook.
- Details on course website, and it is available free online or can buy on Amazon.
- Most other parts of course cannot be found in books.
- I post all slides, and notes in the course on the course website.

Background Material

- If you are completely new to OpenCV and Xcode you should consider getting this book too (link to Amazon.)
- Good beginners guide to using OpenCV in Xcode, so you can build up additional experience during the course.

Resources

- You will need access to a MAC.
- If you do not have a MAC, do not panic CMU has ample MAC clusters on campus.
- See:- https://www.cmu.edu/computing/clusters/facilities/index.html
- We have iPADs for everyone in the class so that is cool (yay!!!) so everyone should have an iOS device.

If you have a MAC

- Every student should have been automatically registered for the Apple's Academic Developer Program (please contact us if that is not the case).
- Please download the Xcode 7 Beta release (you need to be an Apple Developer to download).
- Please ensure your MAC has the latest version of Yosemite.
- Please ensure your iOS device has the latest version 8.4.
- This will make life easy for you (less headaches for me).

Class Participation

- I'll start on time.
- It is important to attend.
  - I will use part slides, part tutorial, part on board.
- Do ask questions.
- Come to office hours or use Piazza.

Sep 1	Why is Computer Vision on a Mobile Device Different?
Sep 3	**Having fun with OpenCV!**
Sep 8	**Using Xcode with OpenCV** Assignment 0 out
Sep 10	**Pin Hole Cameras and Warp Functions**
Sep 15	**Homographies & the Fundamental Matrix**
Sep 17	**BLAS, LAPACK & the Armadillo Library** Assignment 1 out Assignment 0 due (on Fri Sep 18)
Sep 22	**Edge Detection & Segmentation**
Sep 24	**Hough Transform, ICP & Snakes**
Sep 29	**Interest Point Detectors & RANSAC**
Oct 1	**Accessing the GPU & the GPUImage Library** Assignment 2 out Assignment 1 due (on Fri Oct 2)

see 16423.courses.cs.cmu.edu

Oct 6	**The Lucas Kanade Algorithm:- Part 1**
Oct 8	**The Lucas Kanade Algorithm:- Part 2**
Oct 13	**Object Detection & Tracking - Exhaustive Search**
Oct 15	**Correlation Filters for Fast Detection & Tracking** Assignment 3 out Assignment 2 due (on Fri Oct 16)
Oct 20	**Why we need Features/Descriptors in our Detectors?**
Oct 22	**Computationally Efficient Features & the VLFeat Library**
Oct 27	**Object Detection & Tracking - Gradient Search**
Oct 29	**IMU and High-Speed Camera on your Mobile** Assignment 4 out Assignment 3 due (on Fri Oct 30)
Nov 3	**Mid-Term**
Nov 5	**Guest Lecture (Using SWIFT in iOS - TBD)** Project Proposal due (on Fri Nov 6)

see 16423.courses.cs.cmu.edu

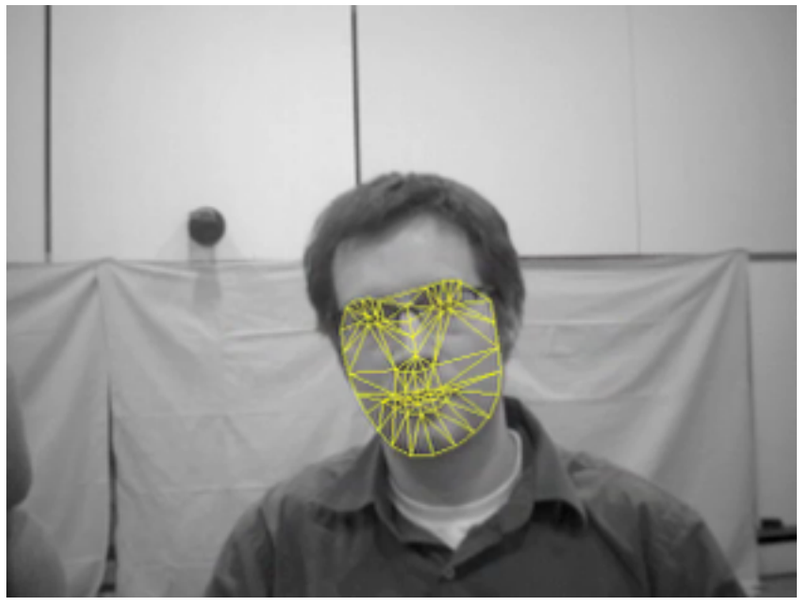
Nov 10	**Deformable Parts Models**
Nov 12	**Optical Flow, SIFT Flow, and Deep Flow**
Nov 17	**Deep Networks in Vision**
Nov 19	**Using Deep Networks on a Mobile Device** Assignment 4 due (on Fri Nov 20)
Nov 24	**Guest Lecture (Visual SLAM - TBD)** Project Checkpoint due (on Nov 24)
Dec 1	**Random Forests**
Dec 3	**Depth Cameras on a Mobile Device**
Dec 8	**Wrap Up Lecture**
Dec 10	**Final Project Presentations** Final Project due (on Fri Dec 11)

see 16423.courses.cs.cmu.edu

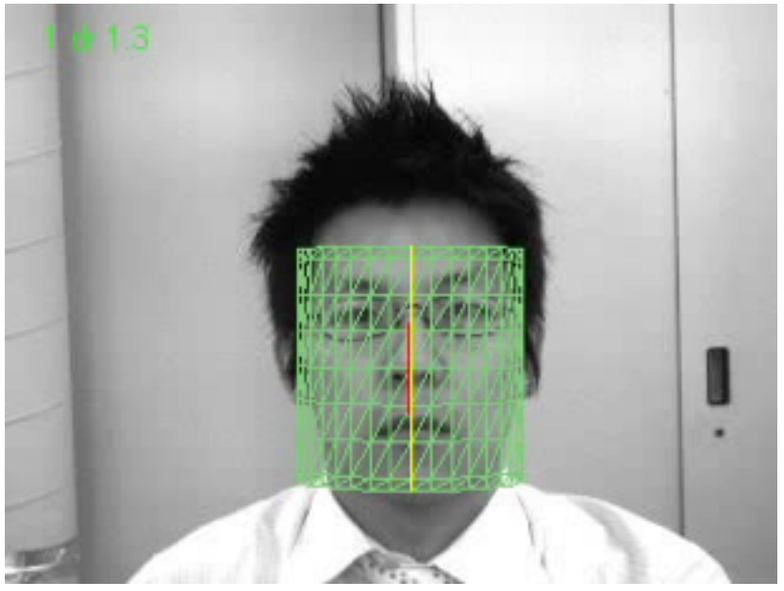
Today

- Course Logistics
- Philosophy to Mobile Computer Vision R&D
- Getting started with OpenCV.

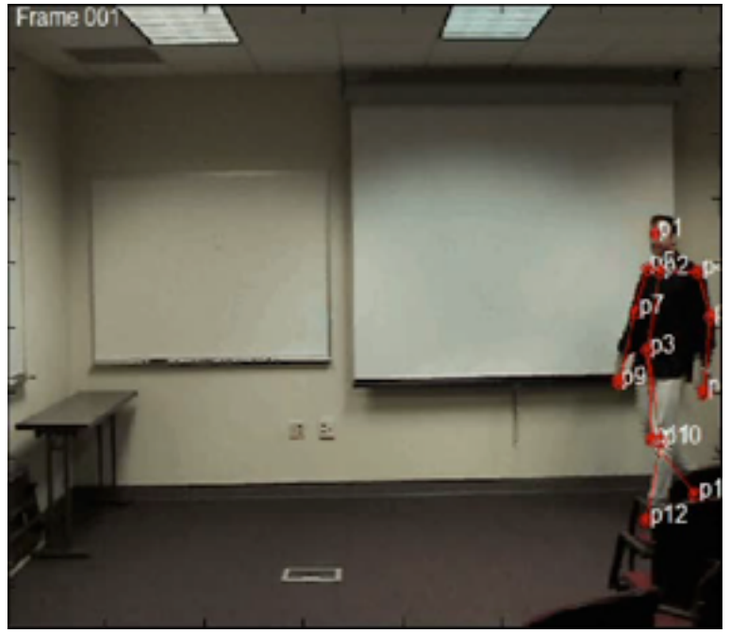
Applications of Computer Vision



“Face Recognition”



“Pose Estimation”



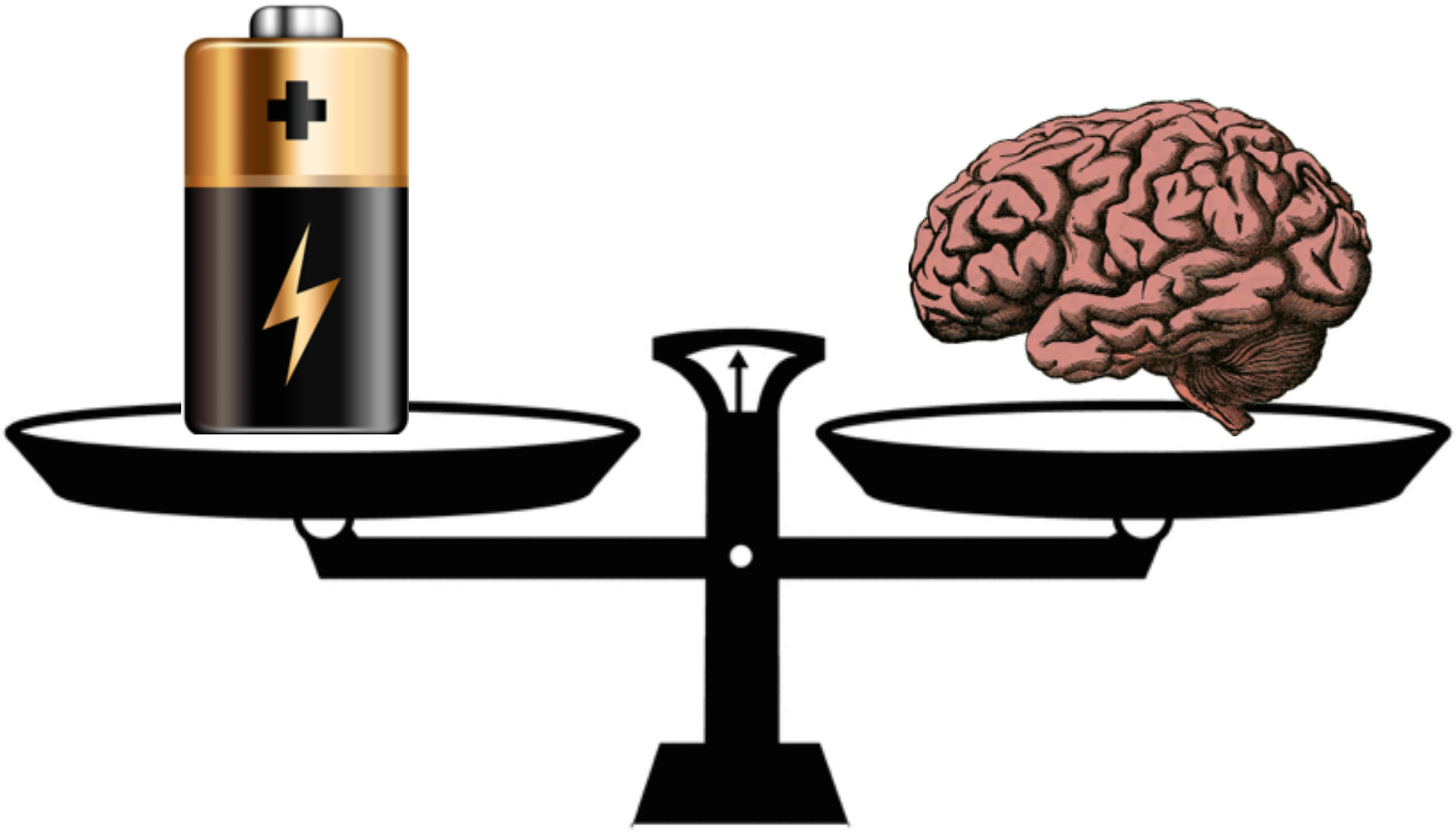
“Body Tracking”

“Speech Reading”

“Palm Recognition”

“Car Tracking”

Balancing Power versus Perception



Algorithm

Software

Architecture

SOC Hardware

Algorithm

Software

Architecture

SOC Hardware

Correlation Filters with Limited Boundaries

Hamed Kiani Galoogahi
Istituto Italiano di Tecnologia
Genova, Italy
hamed.kiani@iit.it

Terence Sim
National University of Singapore
Singapore
tsim@comp.nus.edu.sg

Simon Lucey
Carnegie Mellon University
Pittsburgh, USA
slucey@cs.cmu.edu

Abstract

Correlation filters take advantage of specific properties in the Fourier domain allowing them to be estimated efficiently: $\mathcal{O}(ND \log D)$ in the frequency domain, versus $\mathcal{O}(D^3 + ND^2)$ spatially where D is signal length, and N is the number of signals. Recent extensions to correlation filters, such as MOSSE, have reignited interest of their use in the vision community due to their robustness and attractive computational properties. In this paper we demonstrate, however, that this computational efficiency comes at a cost. Specifically we demonstrate that only $\frac{1}{D}$ proportion of shifted examples are unaffected by boundary effects which has a dramatic impact on detection/tracking performance. In this paper, we propose a novel approach to correlation filter estimation that: (i) takes advantage of inherent computational redundancies in the frequency domain, (ii) dramatically reduces boundary effects, and (iii) is able to implicitly exploit all possible patches densely extracted from training examples during learning process. Impressive object tracking and detection results are presented in terms of both accuracy and computational efficiency.

1. Introduction

Correlation between two signals is a standard approach to feature detection/matching. Correlation touches nearly every facet of computer vision from pattern detection to object tracking. Correlation is rarely performed naively in the spatial domain. Instead, the fast Fourier transform (FFT) affords the efficient application of correlating a desired template/filter with a signal.

Correlation filters, developed initially in the seminal work of Hester and Casasent [15], are a method for learning a template/filter in the frequency domain that rose to some prominence in the 80s and 90s. Although many variants have been proposed [15, 18, 20, 19], the approach's central tenet is to learn a filter, that when correlated with a set of training signals, gives a desired response, e.g. Figure 1 (b). Like correlation, one of the central advantages of the approach is that it attempts to learn the filter in the frequency domain due to the efficiency of correlation in that domain.

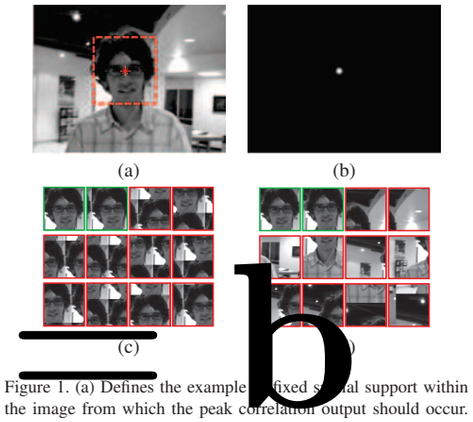


Figure 1. (a) Defines the example fixed spatial support within the image from which the peak correlation output should occur. (b) The desired output response, based on (a), of the correlation filter when applied to the entire image. (c) A subset of patch examples used in a canonical correlation filter where green denotes a non-zero correlation output, and red denotes a zero correlation output in direct accordance with (b). (d) A subset of patch examples used in our proposed correlation filter. Note that our proposed approach uses all possible patches stemming from different parts of the image, whereas the canonical correlation filter simply employs circular shifted versions of the same single patch. The central dilemma in this paper is how to perform (d) efficiently in the Fourier domain. The two last patches of (d) show that $\frac{D-1}{T}$ patches near the image border are affected by circular shift in our method which can be greatly diminished by choosing $D << T$, where D and T indicate the length of the vectorized face patch in (a) and the whole image in (a), respectively.

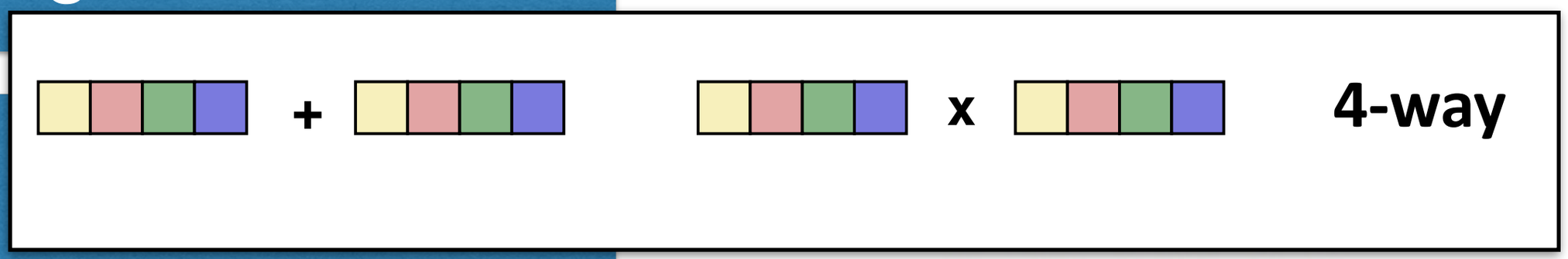
Interest in correlation filters has been reignited in the vision world through the recent work of Bolme et al. [5] on Minimum Output Sum of Squared Error (MOSSE) correlation filters for object detection and tracking. Bolme et al.'s work was able to circumvent some of the classical problems

Algorithm

Software

```
// 5. Now apply some OpenCV operations
cv::Mat gray; cv::cvtColor(cvImage, gray,
    CV_RGBA2GRAY); // Convert to grayscale
cv::GaussianBlur(gray, gray, cv::Size(5,5), 1.2, 1.2); //
    Apply Gaussian blur
cv::Mat edges; cv::Canny(gray, edges, 0, 50); // Estimate
    edge map using Canny edge detector
```

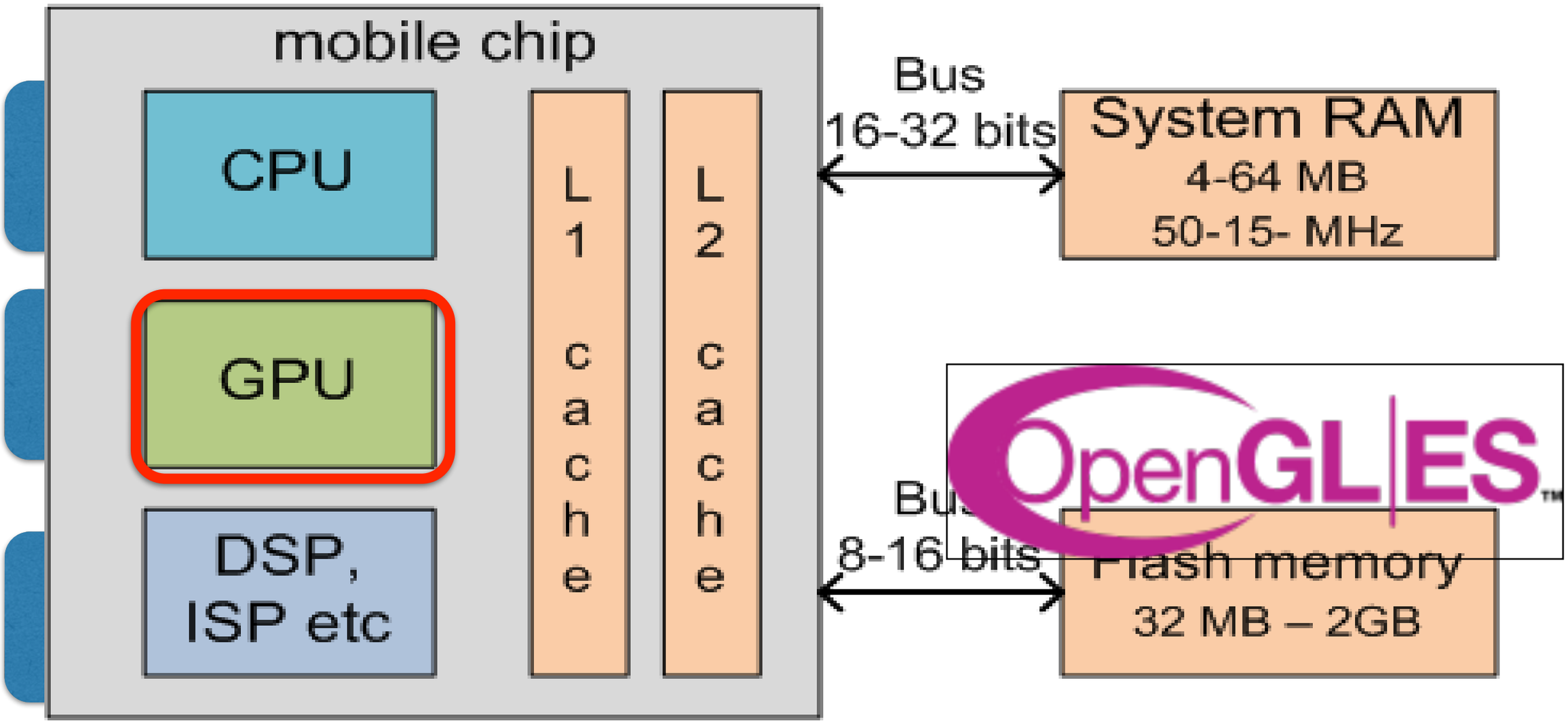
Algorithm



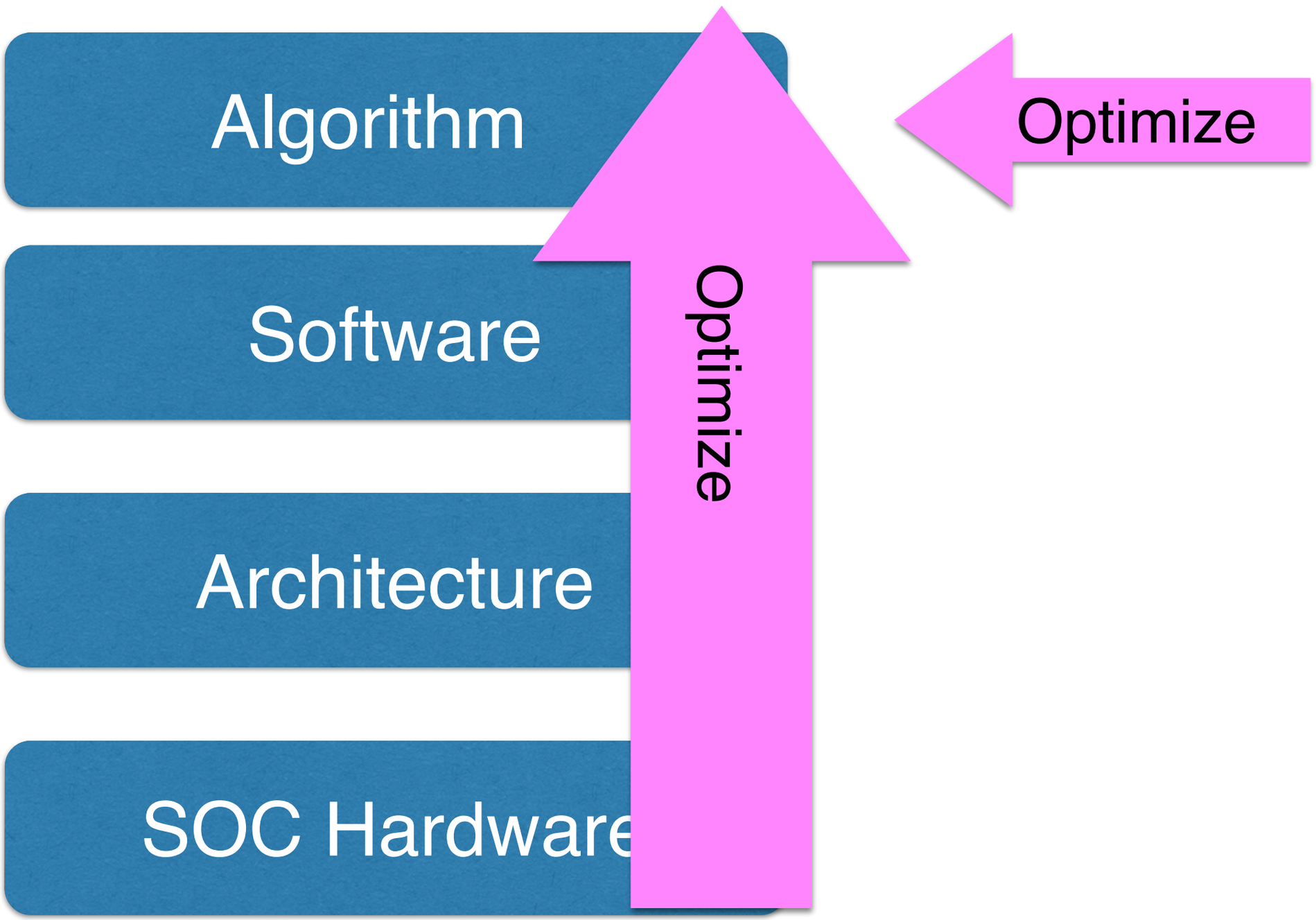
SIMD (Single Instruction, Multiple Data)

Architecture

SOC Hardware



SOC Hardware


Algorithm
Software
Architecture
SOC Hardware
Optimize
Optimize

MATLAB

OpenCV

Some Insights for Mobile CV

- Very difficult to write the fastest code.
  - When you are prototyping an idea you should not worry about this, **but**
  - You have to be **aware** of where bottle necks can occur.
  - This is what you will learn in this course.
- Highest performance in general is non-portable.
  - If you want to get the most out of your system it is good to go deep.
  - However, options like OpenCV are good when you need to build something quickly that works.
- To build good computer vision apps you need to know them algorithmically.
  - Simply knowing how to write fast code is not enough.
  - You need to also understand computer vision algorithmically.
  - OpenCV can be dangerous here.

Some insights taken from Markus Püschel's lectures on "How to Write fast Numerical Code".

Today

- Course Logistics
- Philosophy to Mobile Computer Vision R&D
- Getting started with OpenCV.

What is OpenCV??

- An open source BSD licensed computer vision library.
  - Patent-encumbered code isolated into “non-free” module.
    - SIFT, SURF, some of the Face Detectors, etc.
- Available on all major platforms
  - Android, iOS, Linux, Mac OS X, Windows
- Written primarily in C++
  - Bindings available for Python, Java, even MATLAB (in 3.0).
- Well documented at http://docs.opencv.org
- Source available at https://github.com/Itseez/opencv

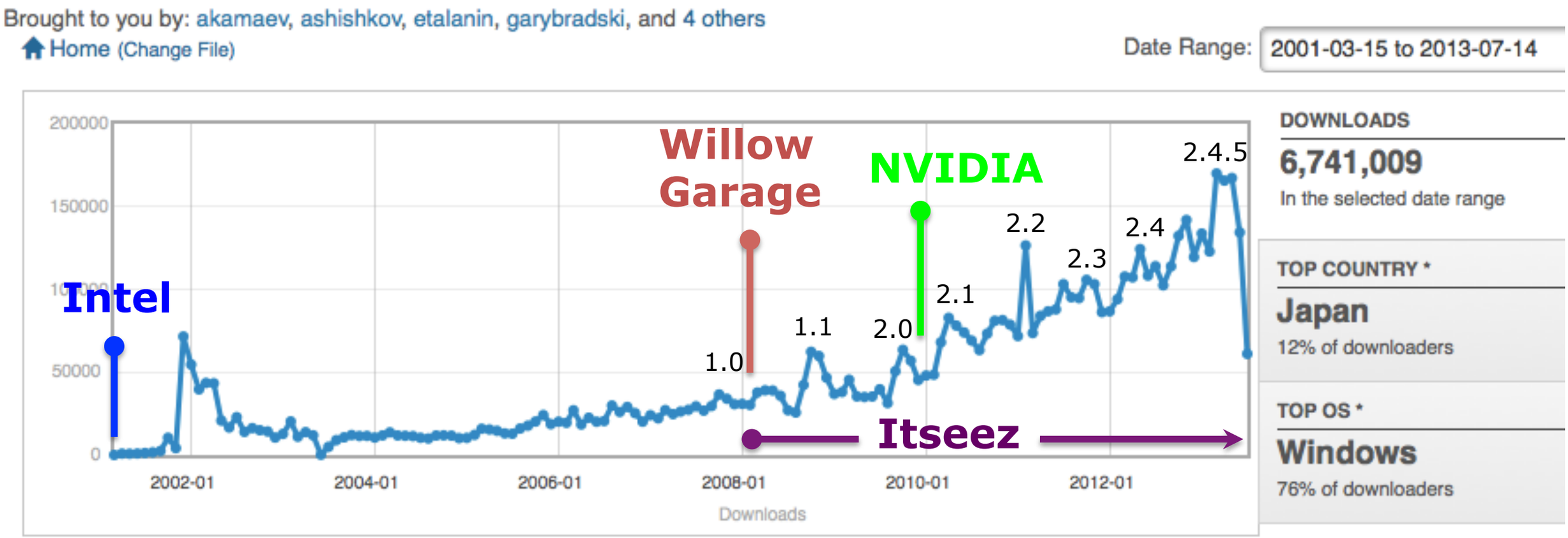
History of OpenCV

- OpenCV started by Intel Research in 1998.
- Goals originally were:-
  - Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
  - Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
  - Advance vision-based commercial applications by making portable, performance-optimized code available for free—with a license that did not require to be open or free themselves.
- Originally released at CVPR 2000.

OpenCV then and now…..

- Version 1.0 was released in 2006.
- In 2008 obtained corporate support from Willow Garage (Robotics Company).
- OpenCV 2 was released in 2009.
  - Included major changes for C++ (mostly C beforehand).
- In 2012 support for OpenCV was taken over by a non-profit foundation OpenCV.org.
- OpenCV 3 was released in 2014.
  - Seems to be under corporate support from Itseez.
  - More on these changes soon.

OpenCV then and now….



Taken from OpenCV 3.0 latest news and the roadmap.

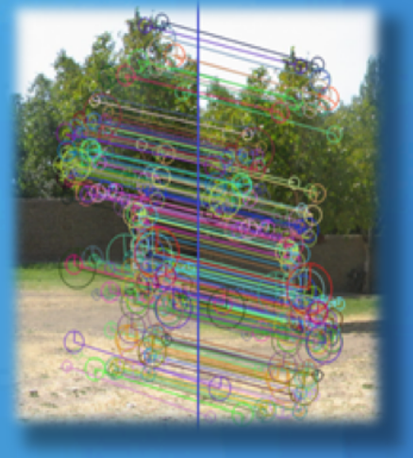
What can OpenCV do?

Image Processing

Filters

Transformations

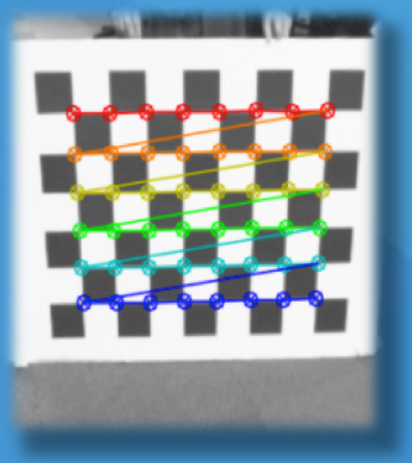
Edges, contours



Robust features

Segmentation

Video, Stereo, 3D



Calibration

Pose estimation

Optical Flow

Detection and recognition

Depth

itseez

Taken from OpenCV 3.0 latest news and the roadmap.

OpenCV 3.0

Migration is relatively smooth from 2.4

- Mostly cleanings
  - Refined C++ API
  - Use **cv::Algorithm** everywhere
- API changes
  - C API will be marked as deprecated
  - Old Python API will be deprecated
  - Monstrous modules will be split into micromodules – Extra modules

OpenCV 3.0

- Sufficiently improved CUDA and OpenCL modules
  - Mobile CUDA support
  - Universal OpenCL binaries (CPU, GPU)
- Hardware Abstraction Layer (HAL)
  - IPP, FastCV-like low-level API to accelerate OpenCV on different HW.
- Open-source NEON optimizations
  - iOS, Android, Embedded.
  - Latest NEWS - 40 NEON optimized functions in 3.0.
- Check out the transition guide.

Which version will be using?

- OpenCV 3.0 is brand new, and is well worth a look and play.
- Most vision tutorials are still in OpenCV 2.4.X.
- OpenCV 2.4.X is still the de facto library for computer vision and image processing.
- Will remain like this until 3.0 matures.

Caution!

- Danger with OpenCV is that it allows you to do a lot with very little understanding for what is going on.
- It is also assumed that you know C++ going forward.

Key OpenCV Classes

`Point_`	Template 2D point class
`Point3_`	Template 3D point class
`Size_`	Template size (width, height) class
`Vec`	Template short vector class
`Matx`	Template small matrix class
`Scalar`	4-element vector
`Rect`	Rectangle
`Range`	Integer value range
`Mat`	2D or multi-dimensional dense array (can be used to store matrices, images, histograms, feature descriptors, voxel volumes etc.)
`SparseMat`	Multi-dimensional sparse array
`Ptr`	Template smart pointer class

(Taken from "OpenCV 2.4 Cheat Sheet")

Matrix Basics

Create a matrix

```
Mat image(240, 320, CV_8UC3);
```

[Re]allocate a pre-declared matrix

```
image.create(480, 640, CV_8UC3);
```

Create a matrix initialized with a constant

```
Mat A33(3, 3, CV_32F, Scalar(5));
Mat B33(3, 3, CV_32F); B33 = Scalar(5);
Mat C33 = Mat::ones(3, 3, CV_32F)*5.;
Mat D33 = Mat::zeros(3, 3, CV_32F) + 5.;
```

Create a matrix initialized with specified values

```
double a = CV_PI/3;
Mat A22 = (Mat_<float>(2, 2) «
   cos(a), -sin(a), sin(a), cos(a));
float B22data[] = {cos(a), -sin(a), sin(a), cos(a)};
Mat B22 = Mat(2, 2, CV_32F, B22data).clone();
```

(Taken from "OpenCV 2.4 Cheat Sheet")

OpenCV 2.4 Cheat Sheet (C++)

The OpenCV C++ reference manual is here: *http://docs.opencv.org*. *Use* ***Quick Search*** *to find descriptions of the particular functions and classes*

Key OpenCV Classes

Point_	Template 2D point class
Point3_	Template 3D point class
Size_	Template size (width, height) class
Vec	Template short vector class
Matx	Template small matrix class
Scalar	4-element vector
Rect	Rectangle
Range	Integer value range
Mat	2D or multi-dimensional dense array (can be used to store matrices, images, histograms, feature descriptors, voxel volumes etc.)
SparseMat	Multi-dimensional sparse array
Ptr	Template smart pointer class

Matrix Basics

Create a matrix

```
Mat image(240, 320, CV_8UC3);
```

[Re]allocate a pre-declared matrix

```
image.create(480, 640, CV_8UC3);
```

Create a matrix initialized with a constant

```
Mat A33(3, 3, CV_32F, Scalar(5));
Mat B33(3, 3, CV_32F); B33 = Scalar(5);
Mat C33 = Mat::ones(3, 3, CV_32F)*5.;
Mat D33 = Mat::zeros(3, 3, CV_32F) + 5.;
```

Create a matrix initialized with specified values

```
double a = CV_PI/3;
Mat A22 = (Mat_<float>(2, 2) «
  cos(a), -sin(a), sin(a), cos(a));
float B22data[] = {cos(a), -sin(a), sin(a), cos(a)};
Mat B22 = Mat(2, 2, CV_32F, B22data).clone();
```

Initialize a random matrix

```
randu(image, Scalar(0), Scalar(256)); // uniform dist
randn(image, Scalar(128), Scalar(10)); // Gaussian dist
```

Convert matrix to/from other structures (without copying the data)

```
Mat image_alias = image;
float* Idata=new float[480*640*3];
Mat I(480, 640, CV_32FC3, Idata);
vector<Point> iptvec(10);
Mat iP(iptvec); // iP – 10x1 CV_32SC2 matrix
IplImage* oldC0 = cvCreateImage(cvSize(320,240),16,1);
Mat newC = cvarrToMat(oldC0);
IplImage oldC1 = newC; CvMat oldC2 = newC;
```

... (with copying the data)

```
Mat newC2 = cvarrToMat(oldC0).clone();
vector<Point2f> ptvec = Mat_<Point2f>(iP);
```

Access matrix elements

```
A33.at<float>(i,j) = A33.at<float>(j,i)+1;
Mat dyImage(image.size(), image.type());
for(int y = 1; y < image.rows-1; y++) {
  Vec3b* prevRow = image.ptr<Vec3b>(y-1);
  Vec3b* nextRow = image.ptr<Vec3b>(y+1);
  for(int x = 0; x < image.cols; x++)
    for(int c = 0; c < 3; c++)
     dyImage.at<Vec3b>(y,x)[c] =
     saturate_cast<uchar>(
     nextRow[x][c] - prevRow[x][c]);
}
Mat_<Vec3b>::iterator it = image.begin<Vec3b>(),
  itEnd = image.end<Vec3b>();
for(; it != itEnd; ++it)
  (*it)[1] ^= 255;
```

Matrix Manipulations: Copying, Shuffling, Part Access

src.copyTo(dst)	Copy matrix to another one
src.convertTo(dst,type,scale,shift)	Scale and convert to another datatype
m.clone()	Make deep copy of a matrix
m.reshape(nch,nrows)	Change matrix dimensions and/or number of channels without copying data
m.row(i), m.col(i)	Take a matrix row/column
m.rowRange(Range(i1,i2)) m.colRange(Range(j1,j2))	Take a matrix row/column span
m.diag(i)	Take a matrix diagonal
m(Range(i1,i2),Range(j1,j2)), m(roi)	Take a submatrix
m.repeat(ny,nx)	Make a bigger matrix from a smaller one
flip(src,dst,dir)	Reverse the order of matrix rows and/or columns
split(...)	Split multi-channel matrix into separate channels
merge(...)	Make a multi-channel matrix out of the separate channels
mixChannels(...)	Generalized form of split() and merge()
randShuffle(...)	Randomly shuffle matrix elements

Example 1. Smooth image ROI in-place

```
Mat imgroi = image(Rect(10, 20, 100, 100));
GaussianBlur(imgroi, imgroi, Size(5, 5), 1.2, 1.2);
```

Example 2. Somewhere in a linear algebra algorithm

```
m.row(i) += m.row(j)*alpha;
```

Example 3. Copy image ROI to another image with conversion

```
Rect r(1, 1, 10, 20);
Mat dstroi = dst(Rect(0,10,r.width,r.height));
src(r).convertTo(dstroi, dstroi.type(), 1, 0);
```

Simple Matrix Operations

OpenCV implements most common arithmetical, logical and other matrix operations, such as

- add(), subtract(), multiply(), divide(), absdiff(), bitwise_and(), bitwise_or(), bitwise_xor(), max(), min(), compare()
  – correspondingly, addition, subtraction, element-wise multiplication ... comparison of two matrices or a matrix and a scalar.

  Example. Alpha compositing function:

```
void alphaCompose(const Mat& rgba1,
    const Mat& rgba2, Mat& rgba_dest)
{
    Mat a1(rgba1.size(), rgba1.type()), ra1;
    Mat a2(rgba2.size(), rgba2.type());
    int mixch[]={3, 0, 3, 1, 3, 2, 3, 3};
    mixChannels(&rgba1, 1, &a1, 1, mixch, 4);
    mixChannels(&rgba2, 1, &a2, 1, mixch, 4);
    subtract(Scalar::all(255), a1, ra1);
    bitwise_or(a1, Scalar(0,0,0,255), a1);
    bitwise_or(a2, Scalar(0,0,0,255), a2);
    multiply(a2, ra1, a2, 1./255);
    multiply(a1, rgba1, a1, 1./255);
    multiply(a2, rgba2, a2, 1./255);
    add(a1, a2, rgba_dest);
}
```

- sum(), mean(), meanStdDev(), norm(), countNonZero(), minMaxLoc(),
  – various statistics of matrix elements.
- exp(), log(), pow(), sqrt(), cartToPolar(), polarToCart()
  – the classical math functions.
- scaleAdd(), transpose(), gemm(), invert(), solve(), determinant(), trace(), eigen(), SVD,
  – the algebraic functions + SVD class.
- dft(), idft(), dct(), idct(),
  – discrete Fourier and cosine transformations

For some operations a more convenient algebraic notation can be used, for example:

```
Mat delta = (J.t()*J + lambda*
   Mat::eye(J.cols, J.cols, J.type()))
   .inv(CV_SVD)*(J.t()*err);
```

implements the core of Levenberg-Marquardt optimization algorithm.

Image Processsing

Filtering

filter2D()	Non-separable linear filter
sepFilter2D()	Separable linear filter
boxFilter(), GaussianBlur(), medianBlur(), bilateralFilter()	Smooth the image with one of the linear or non-linear filters
Sobel(), Scharr()	Compute the spatial image derivatives
Laplacian()	compute Laplacian: $\Delta I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$
erode(), dilate()	Morphological operations

(Taken from "OpenCV 2.4 Cheat Sheet")

Playing with the Mat Object Class

- We are now going to have a play with the Mat Object Class.
- On your browser please go to the address,

https://github.com/slucey-cs-cmu-edu/Example_OCV

- Or better yet, if you have git installed (just use brew install git), you can type from the command line.

```
$ git clone https://github.com/slucey-cs-cmu-edu/Example_OCV.git
```

- See if you can run make on the command line to create the `Example_OCV` executable.

Displaying an Image in OpenCV

- On your browser please go to the address,

https://github.com/slucey-cs-cmu-edu/Show_Lena

- Or again, you can type from the command line.

```
$ git clone https://github.com/slucey-cs-cmu-edu/Show_Lena.git
```

- Question: what happens if you set the imread flag to 0?

Detecting a Face in OpenCV

- On your browser please go to the address,

https://github.com/slucey-cs-cmu-edu/Detect_Lena

- Or again, you can type from the command line.

```
$ git clone https://github.com/slucey-cs-cmu-edu/Detect_Lena.git
```

- Questions: why do you need to clone the Mat image when displaying?

Next Lecture

- Using OpenCV in Xcode.
- Using the Camera in Xcode.
- Checking performance.