Embedded Vision and OpenCV

Gary Bradski

Founder: Industrial Perception, Inc.

President OpenCV.org

Consulting Faculty: Stanford CS.

garybradski@gmail.com

http://www.industrial-perception.com

Outline

1. **What is OpenCV?**
2. The history of OpenCV
3. What is in OpenCV? Why Adopt it?
4. Case studies
5. OpenCV and embedded vision
6. The future of OpenCV

What is OpenCV

- **Open** Source **C**omputer **V**ision Library
- Routines focused on real time image processing and 2D + 3D computer vision.
  - On Linux, Windows, Mac, Android and soon iOS
  - C++, C, Java, Matlab and Python interfaces
- Free for commercial or research use in whole or in part.

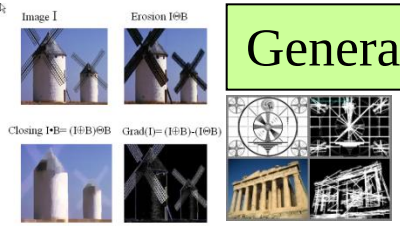
OpenCV Overview:

OpenCV

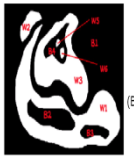
Developer http://code.opencv.org; User: http://opencv.org

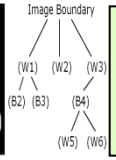
> 2500 algorithms

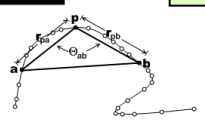
General Image Processing Functions



Geometric descriptors







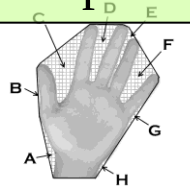
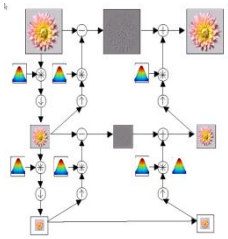
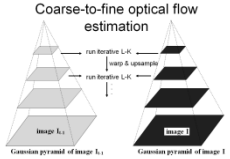
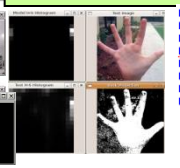


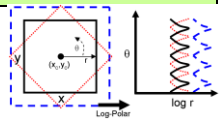
Image Pyramids



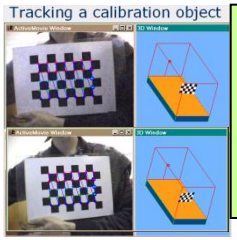


Segmentation

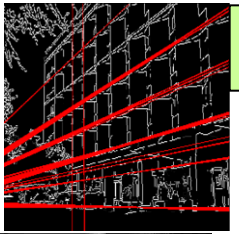


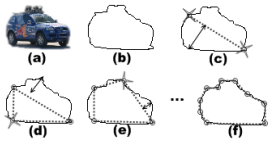


Camera calibration, Stereo, 3D

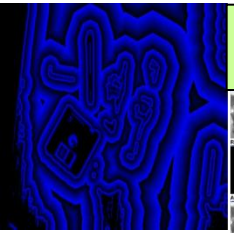


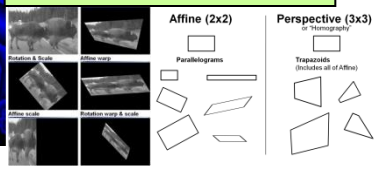
Features



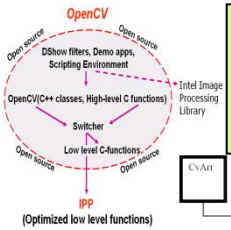


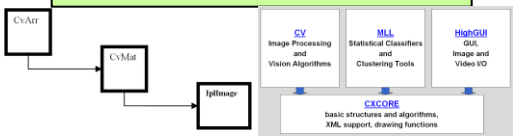
Transforms



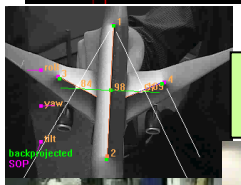


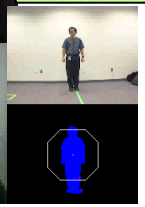
Utilities and Data Structures

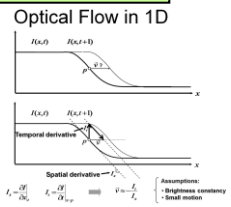




Tracking

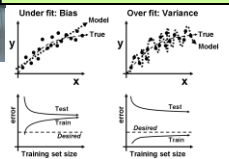




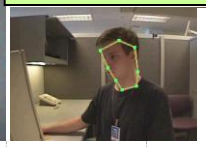


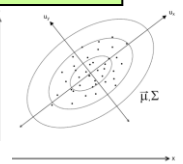
Machine Learning:

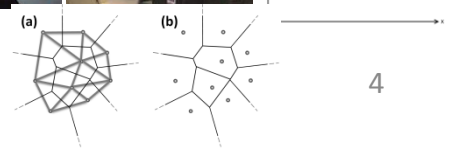
- Detection,
- Recognition



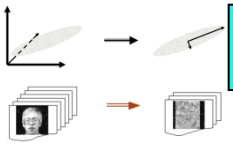
Fitting







Matrix Math

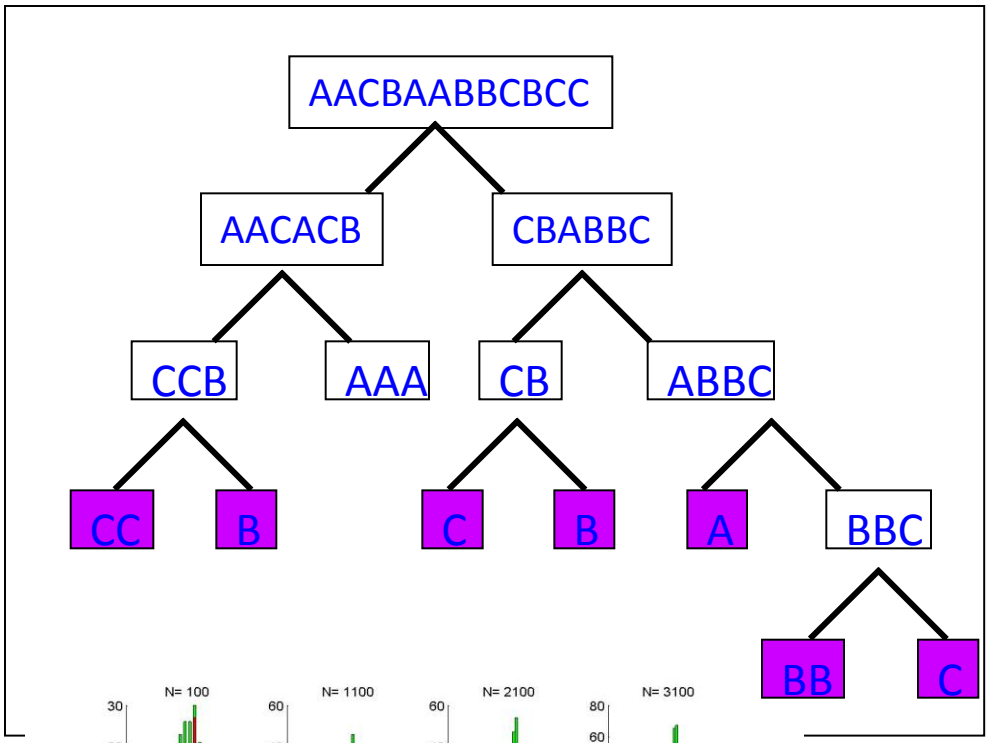


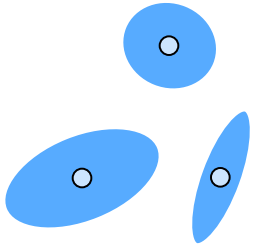
Machine Learning Library (MLL)

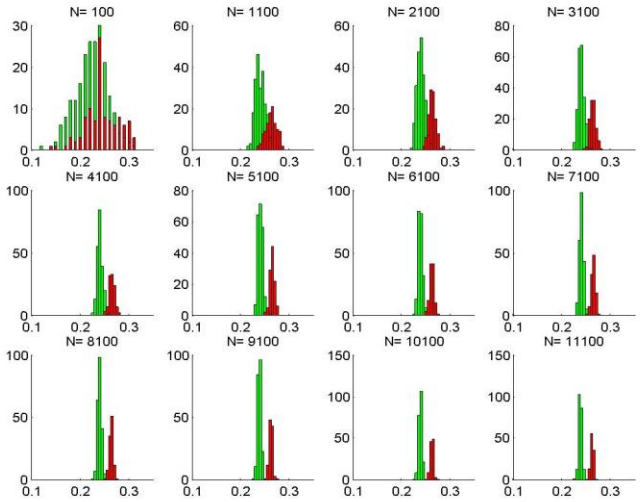
CLASSIFICATION / REGRESSION
Fast Approximate NN (FLANN)
Extremely Random Trees
CART
Naïve Bayes
MLP (Back propagation)
Statistical Boosting, 4 flavors
Random Forests
SVM
Face Detector
(Histogram matching)
(Correlation)

CLUSTERING
K-Means
EM
(Mahalanobis distance)

TUNING/VALIDATION
Cross validation
Bootstrapping
Variable importance
Sampling methods







http://opencv.org

5

OpenCV License

- Based on BSD license
- Free for **commercial** and research use
- Does **not force** your code to be open
- You need not contribute back
  - But you are very to!

Environments, Platforms

- Languages:
  - C++,C#,Python,C,Java
- Processors
  - Intel+MMX+SSE, GPU/Tegra, ARM
- Platforms:




OpenCV Foundation

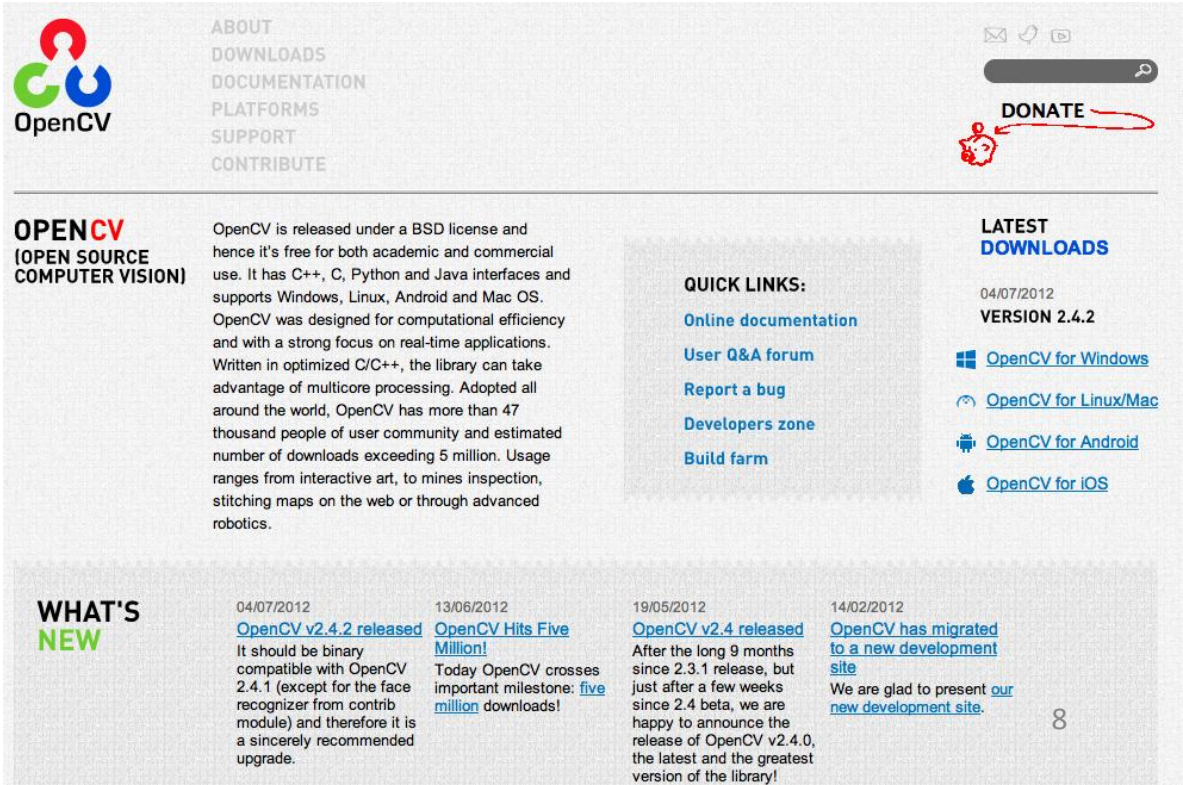
- **Founded this July, 2012**
- **http://opencv.org (user site)** **Documentation: http://docs.opencv.org**
- **http://code.opencv.org (developer site)**
- **Contribute (via Credit, debit or paypal):**
  - **http://tinyurl.com/7eujyo2**

For larger corporate support And/or partnership, contact
Garybradski@gmail.com

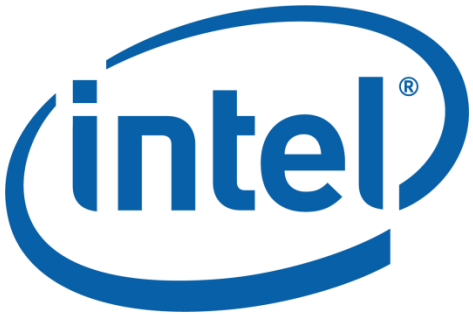
There will be
- Platinum
- Gold
- Silver
- Bronze

levels. May include email and/or phone support; a say in decision process.

Partnership can be very creative, we have consulting teams of developers, workshops, education, contests etc.



OpenCV.org sponsors

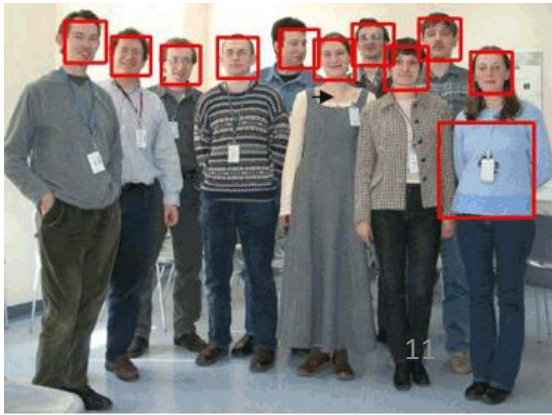


Outline

1. *What is OpenCV?*
2. **The history of OpenCV**
3. What is in OpenCV? Why Adopt it?
4. Case studies
5. OpenCV and embedded vision
6. The future of OpenCV

OpenCV History (motives)

- I (Gary Bradski) started OpenCV in 1999 in Intel's Microprocessor Research Labs because:
  - I was chartered with finding new uses for increasing CPU performance.
  - I wanted to advance the field of computer vision
    - by giving everyone a common infrastructure to stop reinventing the basics;
    - making this infrastructure open and free to prevent future monopolies in computer vision from taking over and then shutting down progress.



OpenCV History (Early resources)

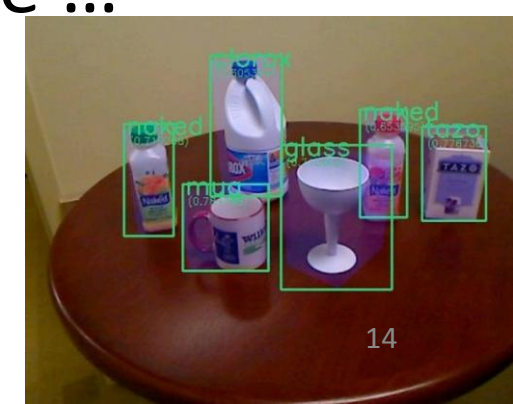
- At Intel, in addition to my local team (Jean-Yves Bouguet, Bob Davies, Ara Nefian), I was able to leverage the software libraries group in Sarov and in Nihzniy Novgorod, Russia. Some early names on that team:
  - Vadim Pisarevsky, principle developer
  - Victor Eruhimov, researcher
  - Sergey Molinov, researcher
  - Alexander Bovyrin, developer
- These were fun times, rapid advances, many research papers.
  - My manager Mark Holler and all the way up the hierarchy (Bob Liang, Fred Pollack, Justin Rattner, and especially Richard Wirt) were very supportive.
  - Omid Moghadam helped market OpenCV

OpenCV History (early years)

- We presented the first release of OpenCV at CVPR 2000
- In 2000, the .boom went .bust strongly impacting Intel revenues.
  - Computer vision and open source became lower priorities.
  - I shifted the teams to work on Intel manufacturing problems, predicting and reducing sources of error.
- We kept OpenCV going quietly on the side
  - Much of the machine learning that is in OpenCV came (in re-written from scratch form) from our manufacturing work period.

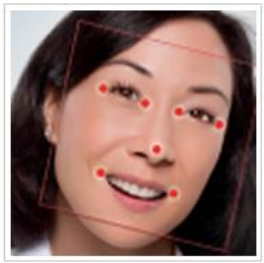
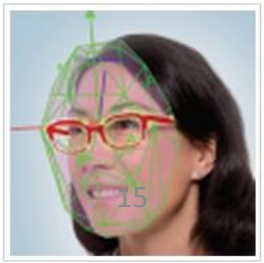
OpenCV History (exits)

- I wanted to organize efforts to get vision into hardware and to look towards robotics. It wasn't supported and I left Intel in 2005 to join VideoSurf (sold to Microsoft 2011).
  - Much of my local team left to Google, NASA and startups.
  - Much of the original Russian team eventually left to form a software contracting group itseez.com in 2008.
- Still, we kept OpenCV going on the side …
  - The years of "wandering in the desert"

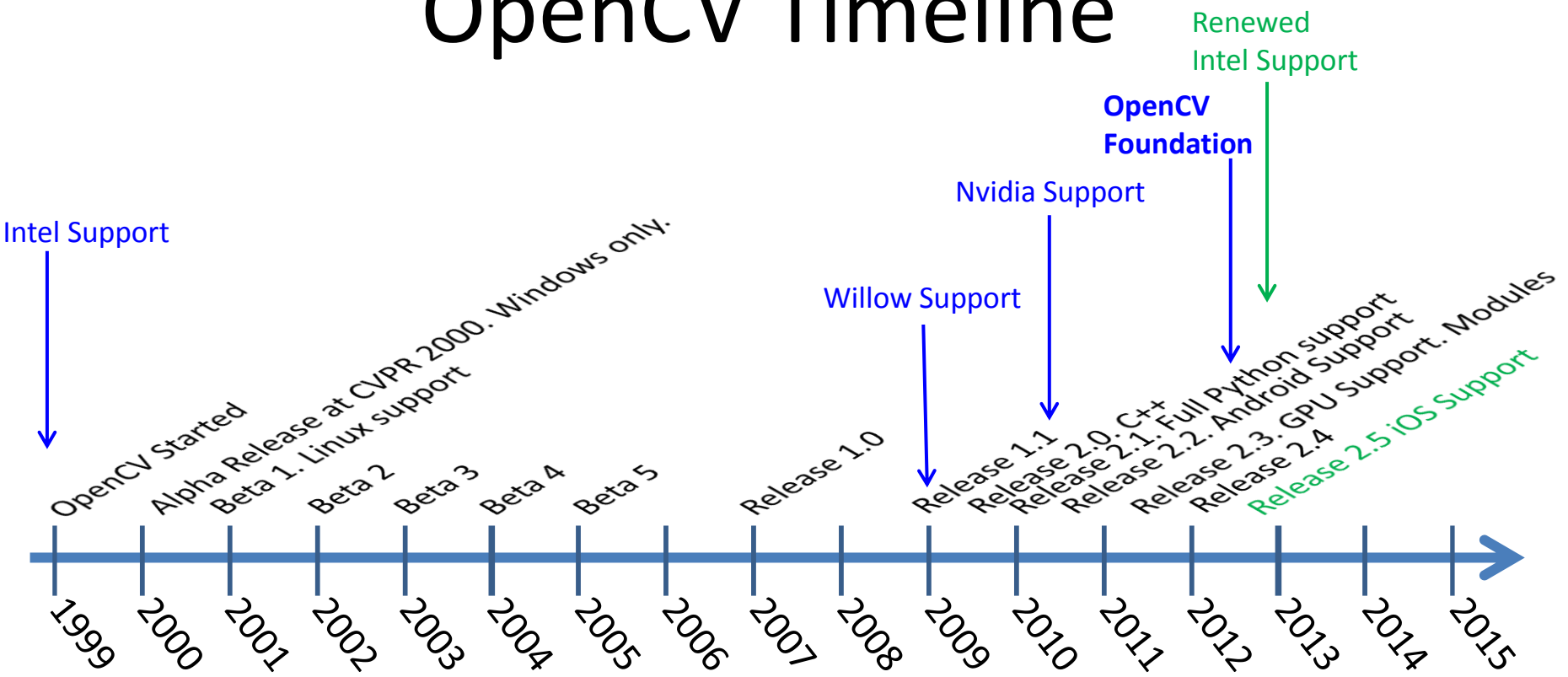


OpenCV History (revival)

- I left Video Surf in 2007 to join Willow Garage, a robotics incubator.
  - Willow Garage began contracting many of the original OpenCV team who were starting Itseez.com to refresh the library in 2008.
    - Book: Learning OpenCV, G. Bradski and A. Kaehler came out in 2008.
    - Willow Garage is still **the** key supporter of OpenCV!
  - Google started key support to OpenCV via Google Summer of Code.
- OpenCV became a non-profit foundation 2012
  - New Intel support seems likely
    - They are investing in vision again
    - http://www.gizmag.com/intel-ultrabook-4th-generation/24116/

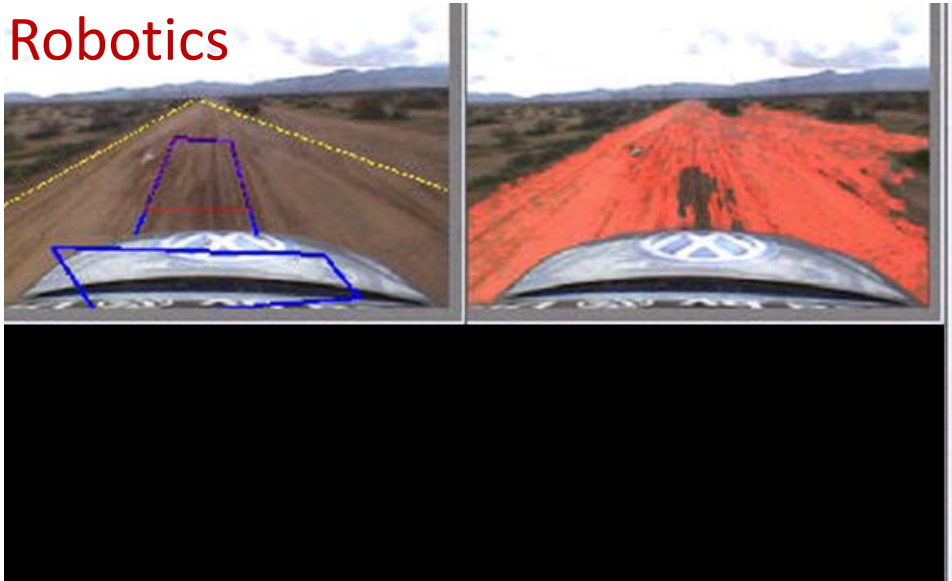
OpenCV Timeline

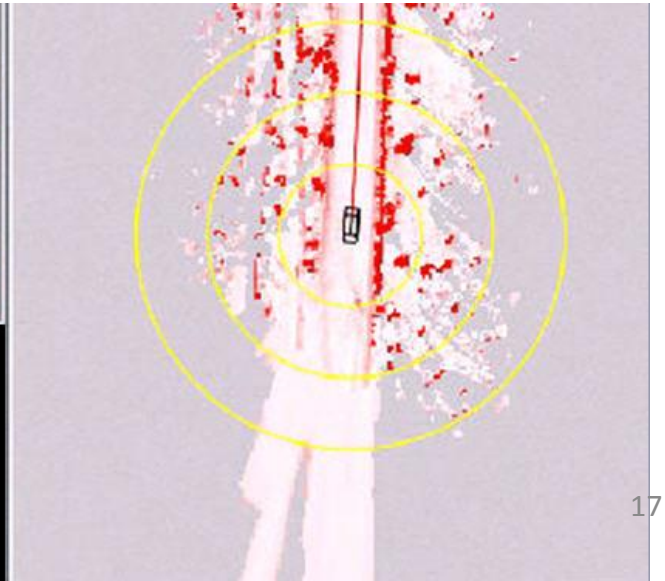


Main Current Sponsors:

Where is OpenCV Used?

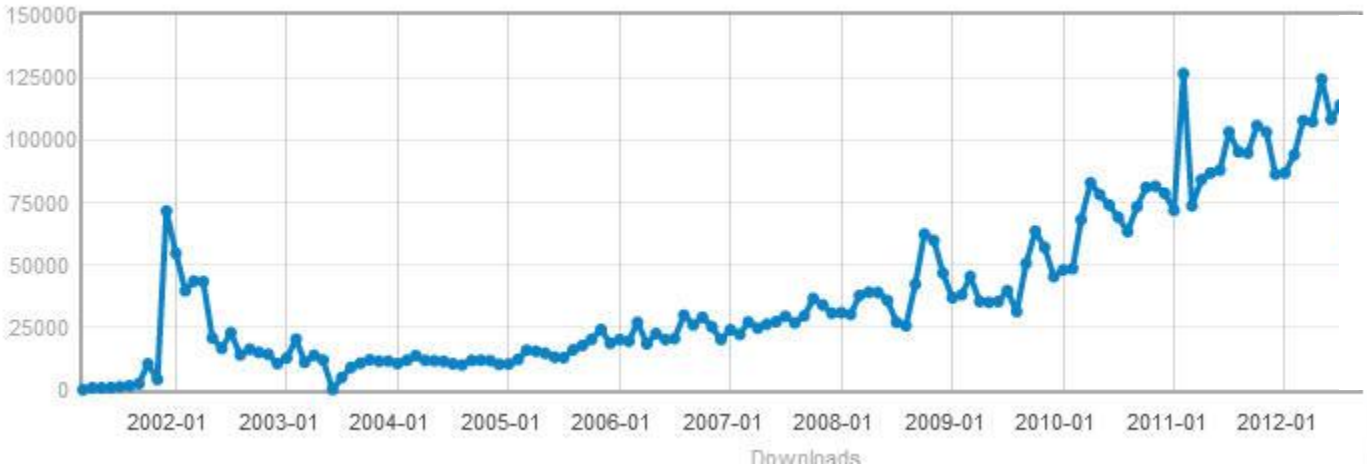
- Academic and Industry Research
- Security systems
- Google Maps, Streetview
- Image/video search and retrieval
- Structure from motion in movies
- Machine vision factory production inspection systems
- Automatic Driver Assistance Systems
- Safety monitoring (Dam sites, mines, swimming pools)
- Robotics





Where is OpenCV Used?

Over 5M downloads!



DOWNLOADS

5,334,861

In the selected date range

TOP COUNTRY *

Japan

13% of downloaders

TOP OS *

Windows

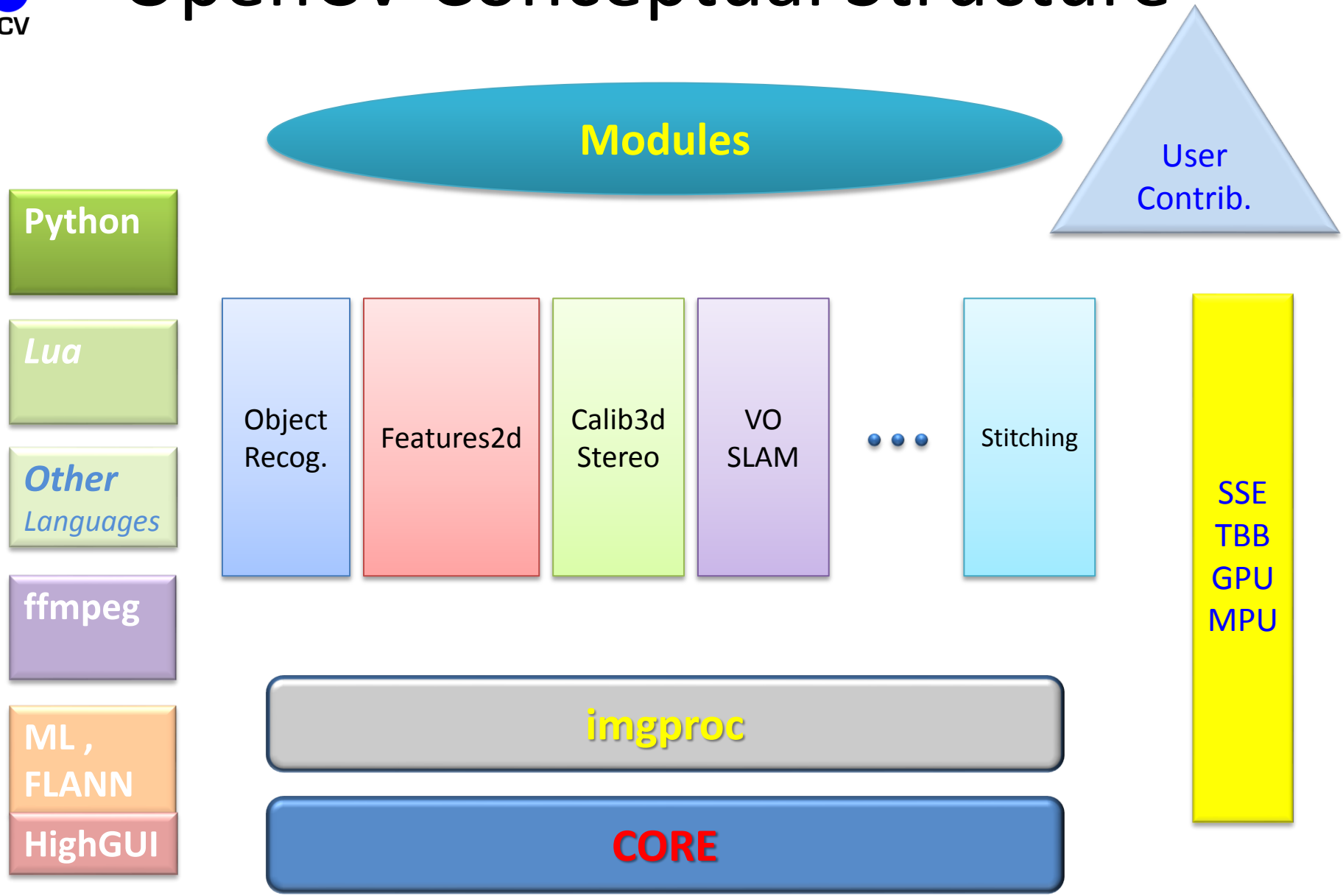
76% of downloaders

Ramping to > 110K downloads/month

Outline

1. *What is OpenCV?*
2. *The history of OpenCV*
3. **What is in OpenCV? Why Adopt it?**
4. Case studies
5. OpenCV and embedded vision
6. The future of OpenCV

OpenCV Conceptual Structure



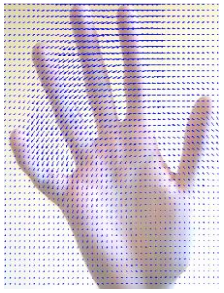
OpenCV Algorithm Modules Overview

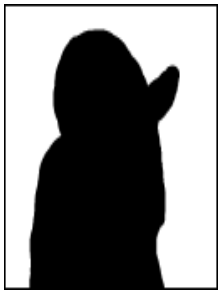
HighGUI:
I/O, Interface

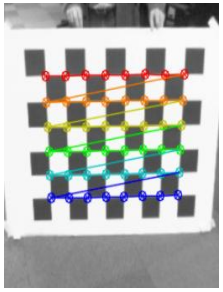

Image Processing

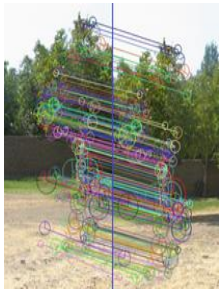

Transforms

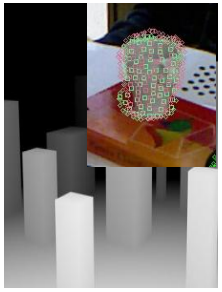

Fitting


Optical Flow
Tracking


Segmentation


Calibration


Features
VSLAM


Depth, Pose
Normals, Planes,
3D Features


Object recognition
Machine learning


Computational
Photography

CORE:
Data structures, Matrix math, Exceptions etc

CORE:
Data structures, Matrix math, Exceptions etc

OpenCV Modules: Core

OpenCV Core (C++)

The OpenCV C++ reference manual is here:
http://opencv.willowgarage.com/documentation/cpp/.
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; y < 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))	Take a matrix row/column span
m.colRange(Range(j1,j2))	
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

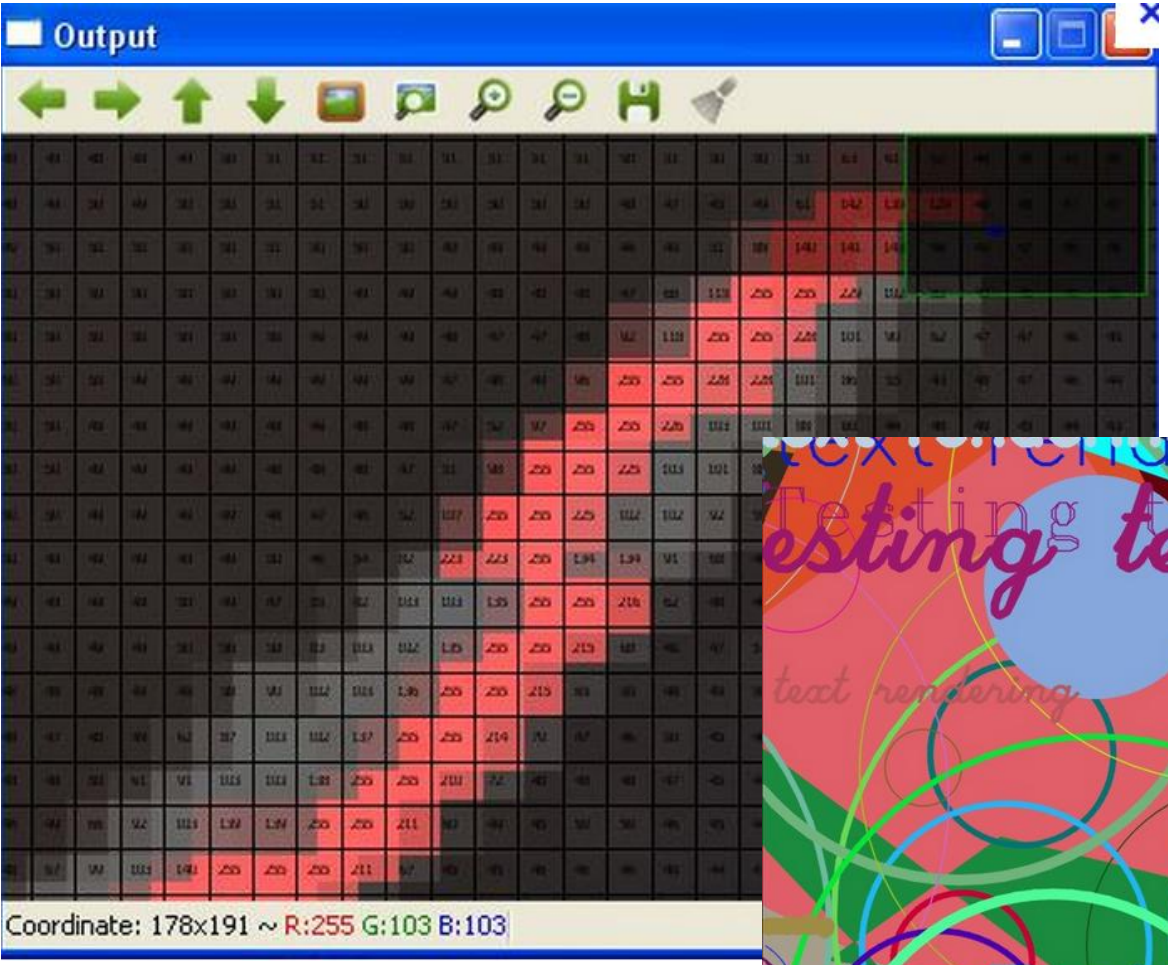
- add(), subtract(), multiply(), divide(), absdiff(), bitwise_and(), bitwise_or(), bitwise_xor(), max(), min(), compare()
  – correspondingly, addition, subtraction, element-wise multiplication ... comparison of two matrices or a matrix and a scalar.

  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.

HighGUI:
I/O, Interface

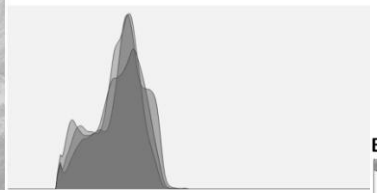
OpenCV Modules: HighGUI



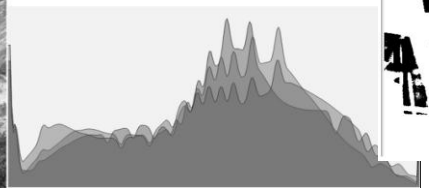
OpenCV Modules: Image Processing

Image

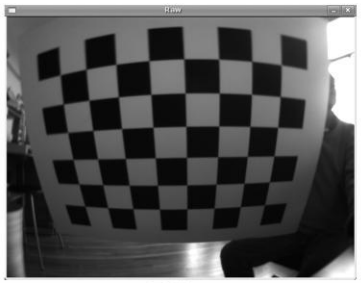
Low Dynamic Range Image and its Histogram



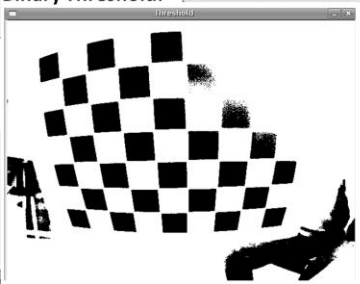
Histogram Equalized Image and its Histogram



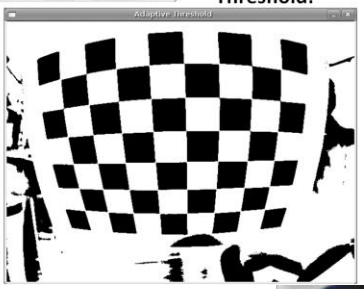
Source Image:

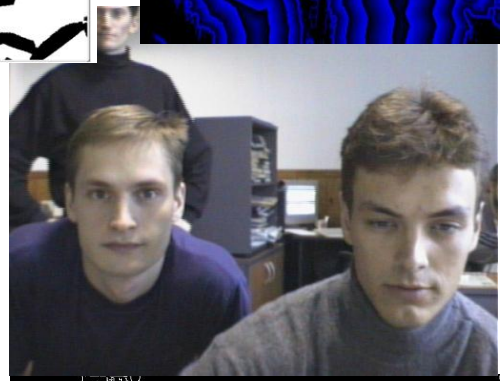


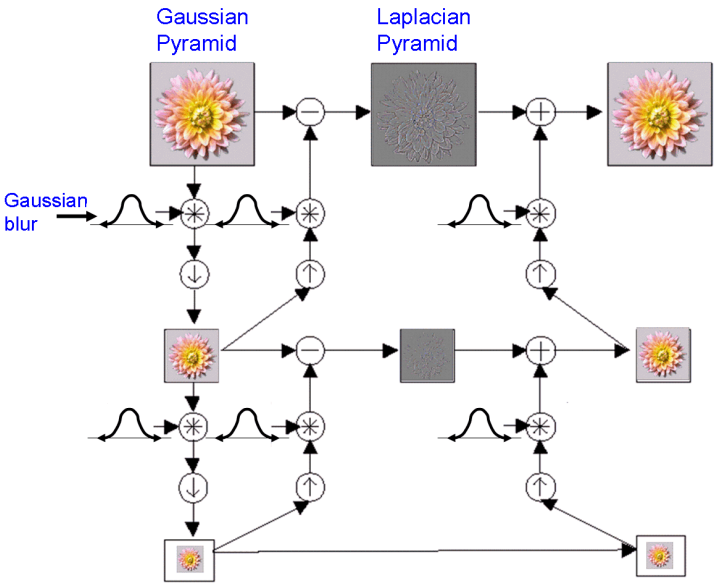
Binary Threshold:

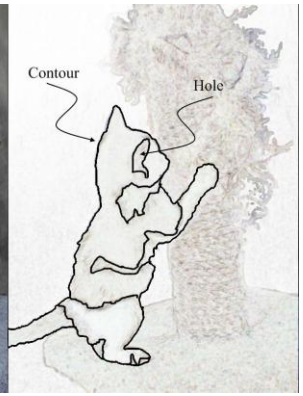


Adaptive Binary Threshold:



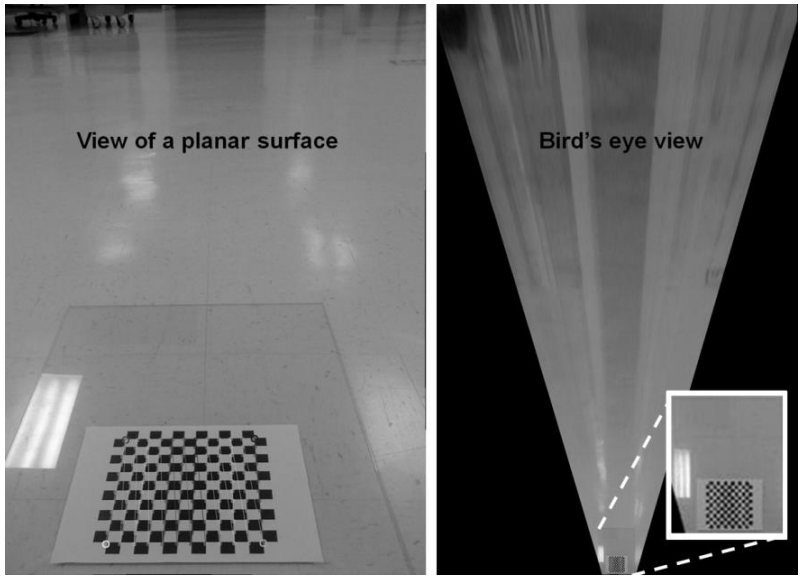


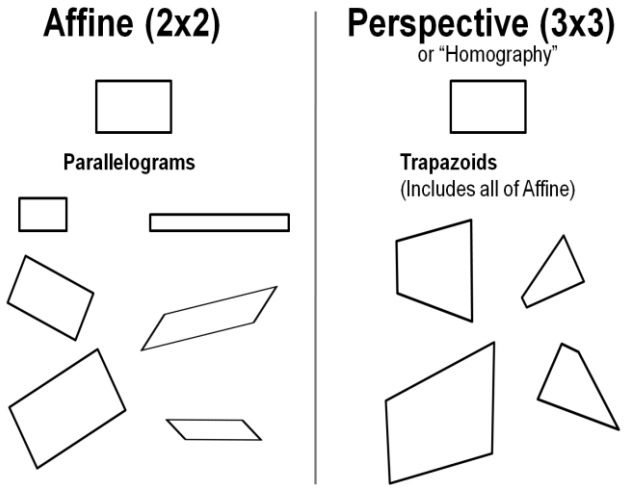


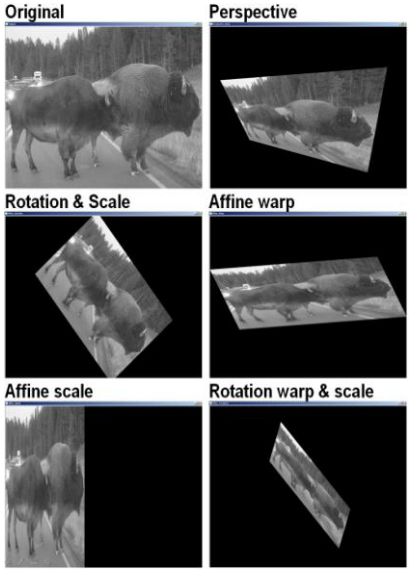


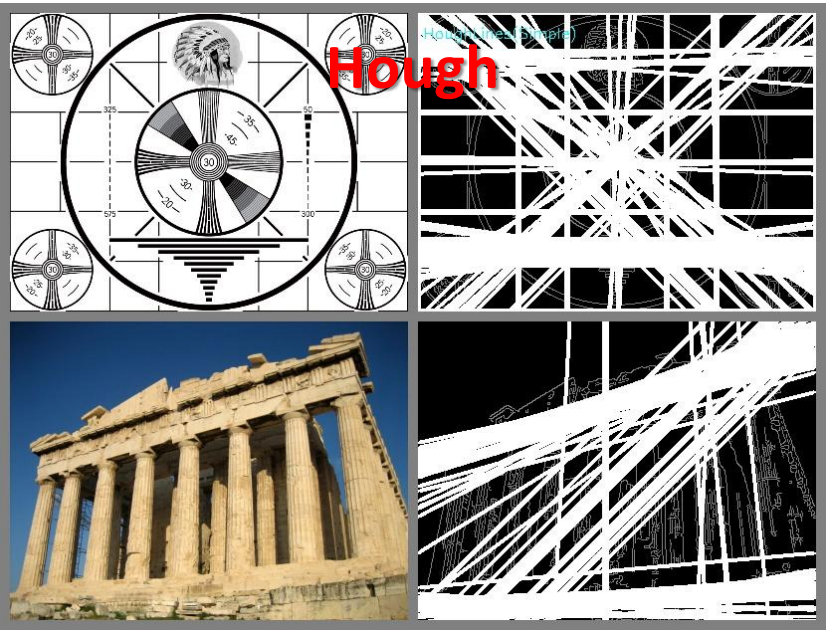
OpenCV Modules: Transforms

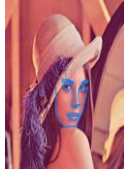
Transforms





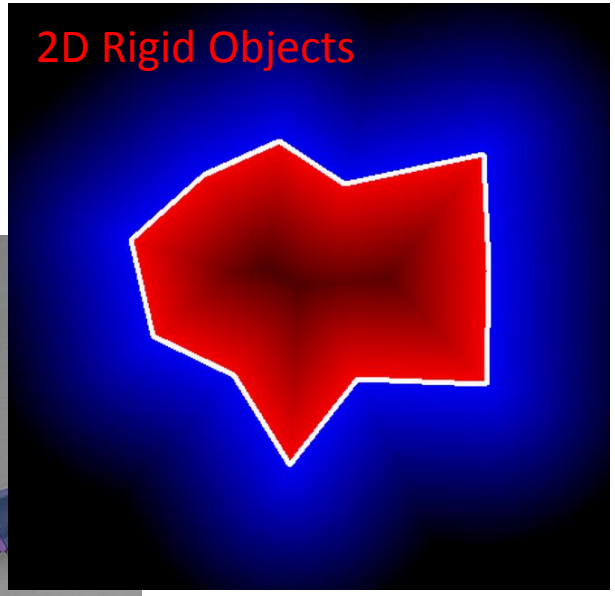


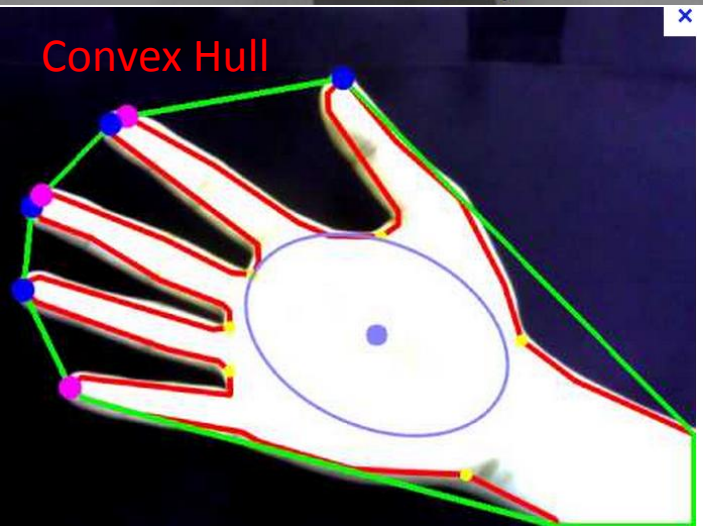


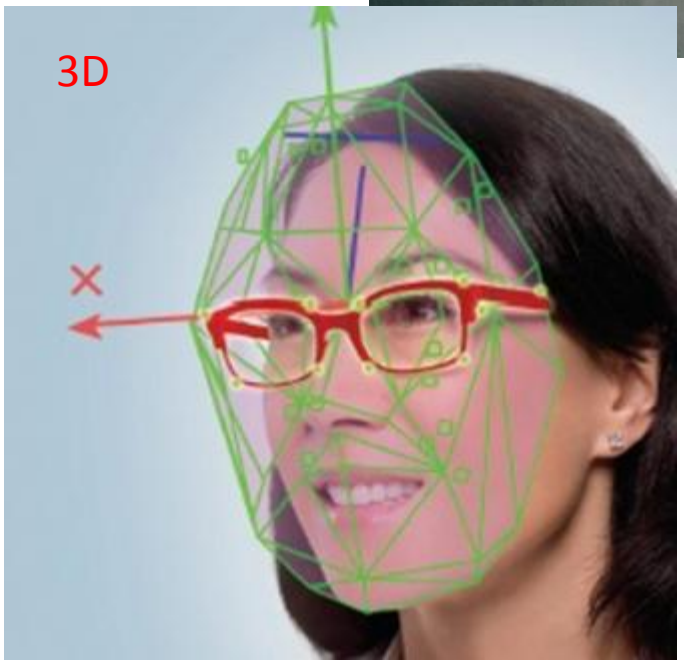


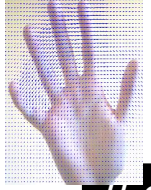
Fitting

OpenCV Modules: Fitting





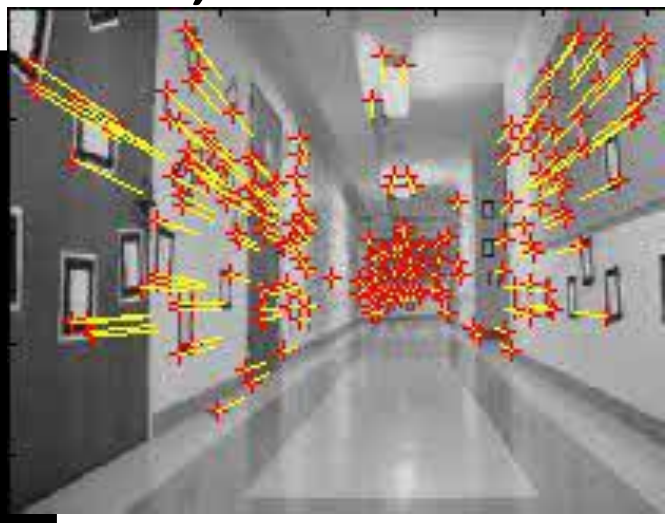




Optical Flow Tracking

OpenCV Modules: Optic Flow, Track

http://www.youtube.com/watch?v=bWyBGmzfP-g



```
// opencv/samples/c/lkdemo.c
int main(…){
…
CvCapture* capture = <…> ?
     cvCaptureFromCAM(camera_id) :
     cvCaptureFromFile(path);
if( !capture ) return -1;
for(;;) {
   IplImage* frame=cvQueryFrame(capture);
   if(!frame) break;
   // … copy and process image
cvCalcOpticalFlowPyrLK( …)
   cvShowImage( “LkDemo”, result );
   c=cvWaitKey(30); // run at ~20-30fps speed
   if(c >= 0) {
       // process key
 }}
cvReleaseCapture(&capture
```

lkdemo.c, 190 lines

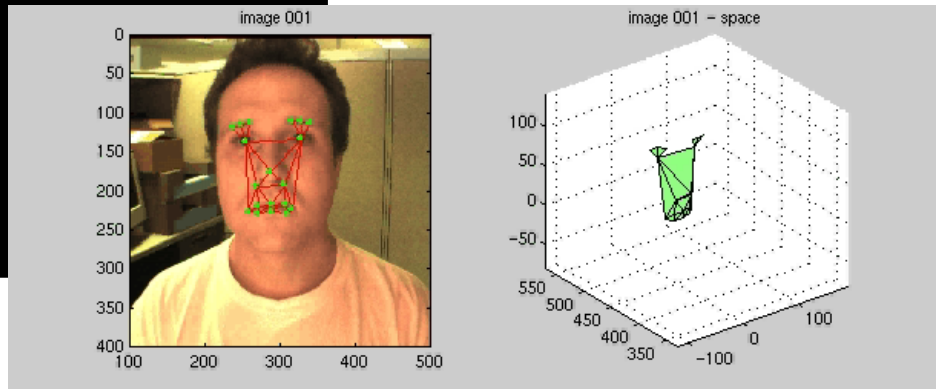
(needs camera to run)

$$I(x + dx, y + dy, t + dt) = I(x, y, t);$$

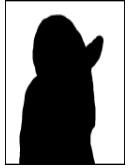
$$-\frac{\partial I}{\partial t} = \frac{\partial I}{\partial x}\cdot(dx/dt) + \frac{\partial I}{\partial y}\cdot(dy/dt);$$

$$G \cdot X = b,$$

$$\frac{\partial X}{\partial (x, y), G} = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, b = \sum \begin{bmatrix} I_x \\ I_t I_y \end{bmatrix}$$

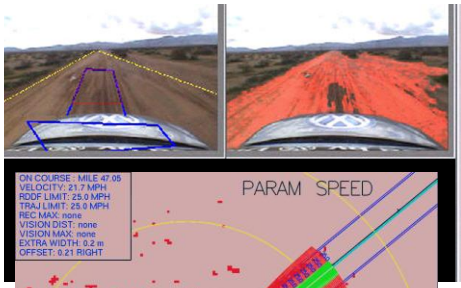


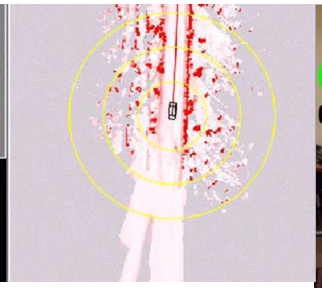
http://www.youtube.com/watch?v=1osj7kRgswk

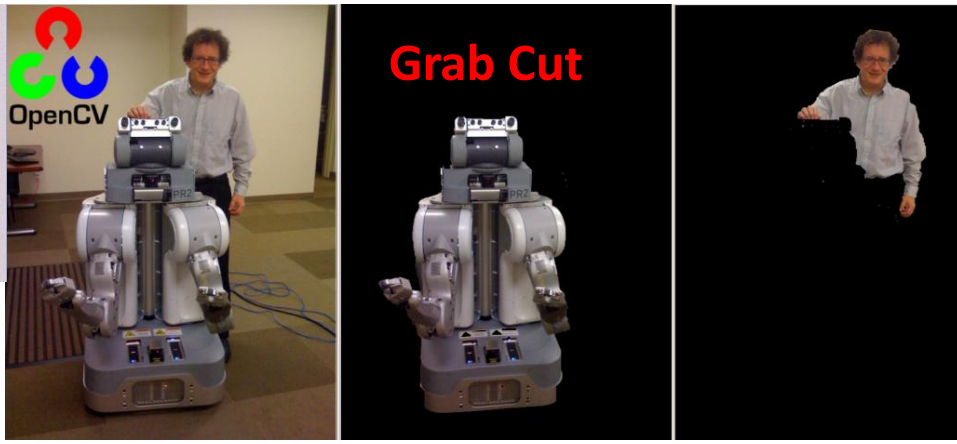


Segmentation

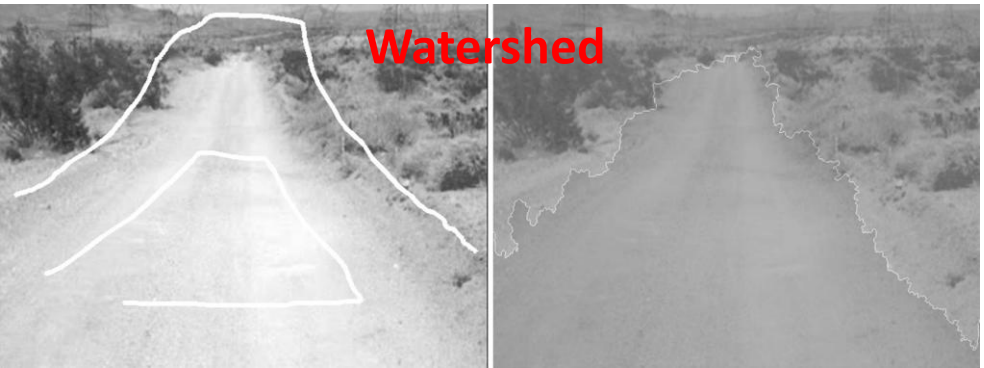
OpenCV Modules: Segmentation

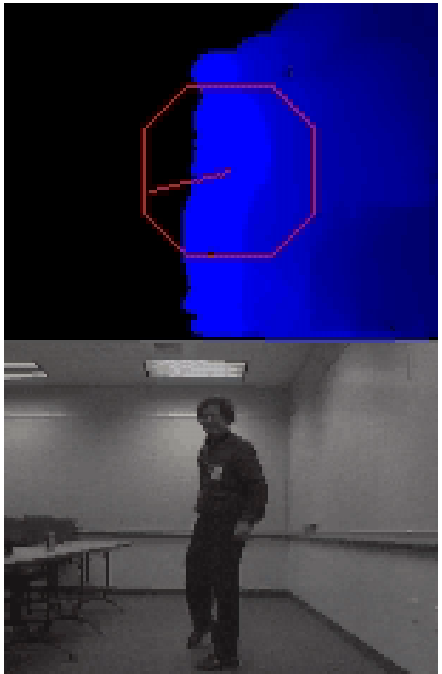


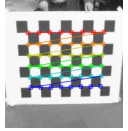




https://www.youtube.com/watch?v=OxmDonZja74
http://www.youtube.com/watch?v=Ktrjh5-KLKo



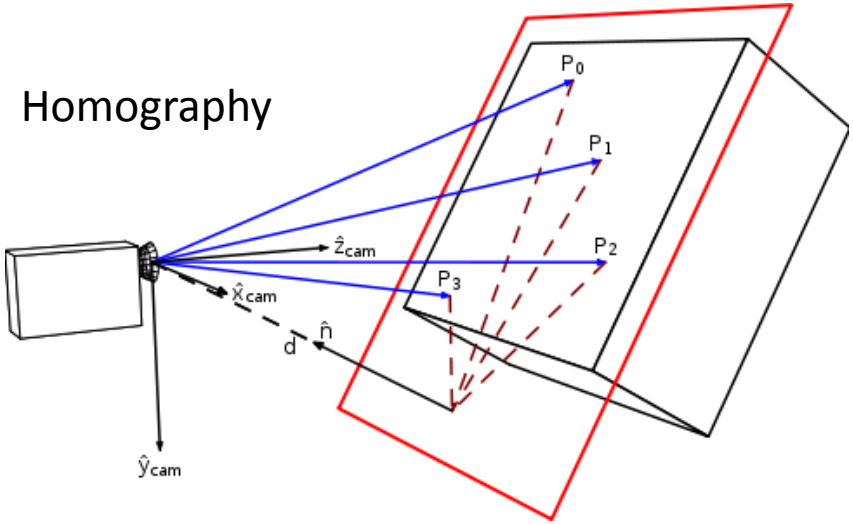


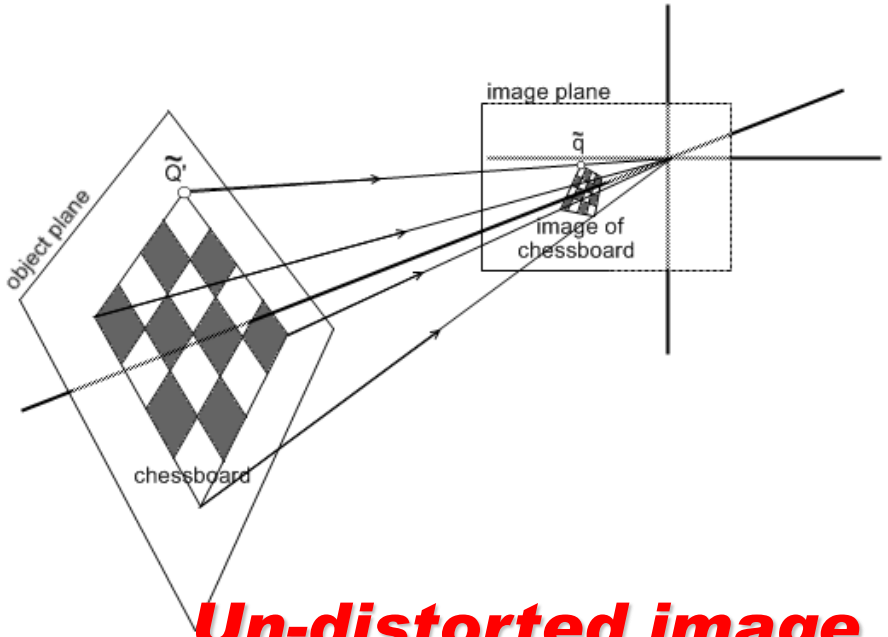


Calibration

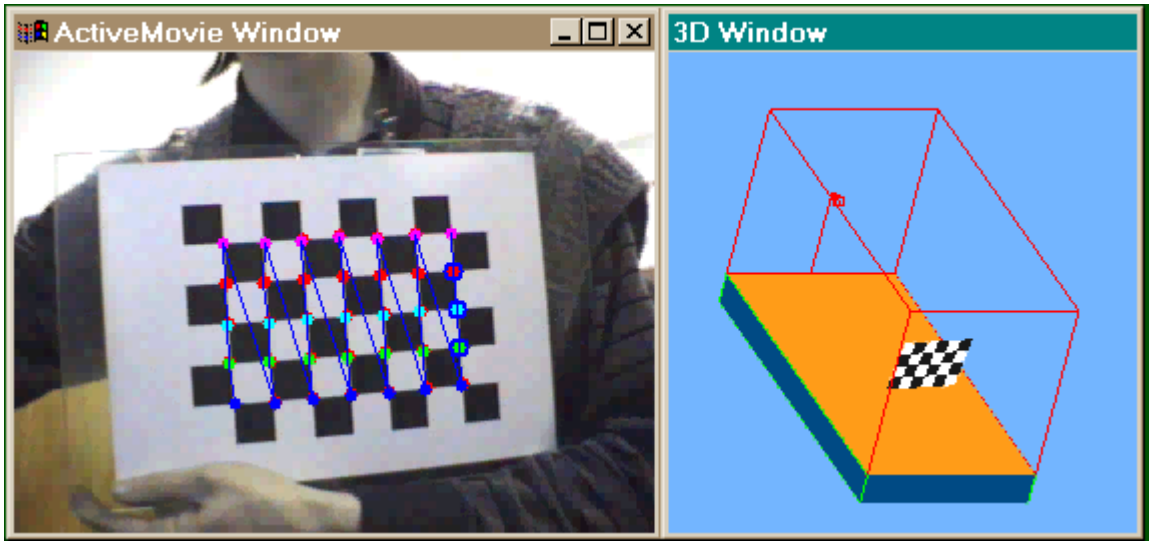
OpenCV Modules: Calibration

Homography





3D view of checkerboard

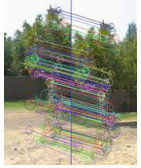


Un-distorted image

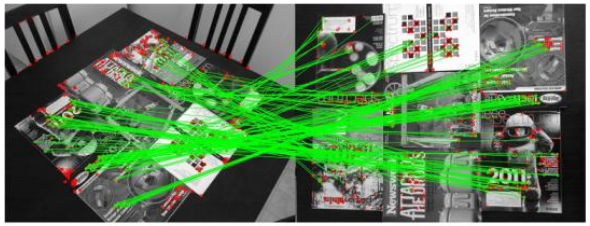
http://www.youtube.com/watch?v=DrXIQfQHFv0

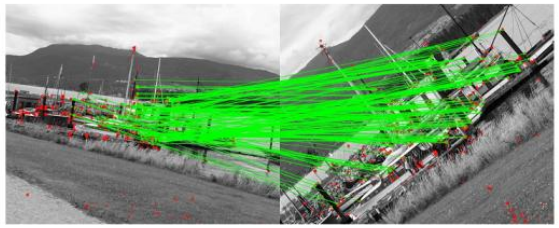
http://www.youtube.com/watch?v=PuWQnCReIeE

OpenCV Modules: Features, VSLAM



Features
VSLAM





Read two input images:

```
Mat img1 = imread(argv[1], CV_LOAD_IMAGE_GRAYSCALE);
```

Detect keypoints in both images:

```
// detecting keypoints
FastFeatureDetector detector(15);
vector<KeyPoint> keypoints1;
detector.detect(img1, keypoints1);
```

Compute descriptors for each of the keypoints:

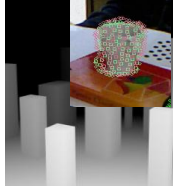
```
// computing descriptors
SurfDescriptorExtractor extractor;
Mat descriptors1;
extractor.compute(img1, keypoints1, descriptors1);
```

Now, find the closest matches between descriptors from the first image to the second:

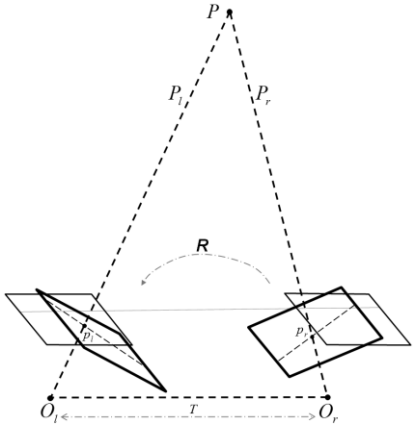
```
// matching descriptors
BruteForceMatcher<L2<float> > matcher;
vector<DMatch> matches;
matcher.match(descriptors1, descriptors2, matches);
```

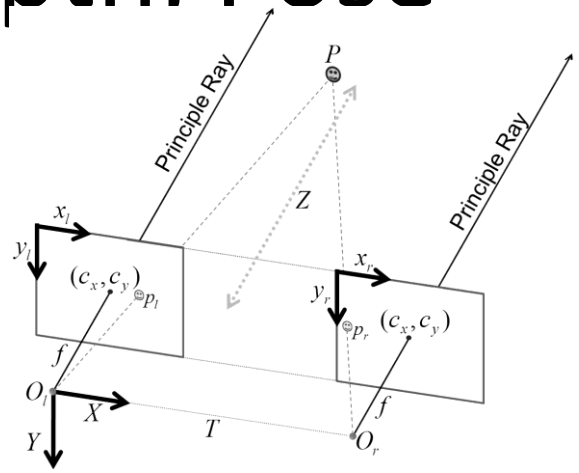
Change one or both of these lines to switch detector and/or descriptor types

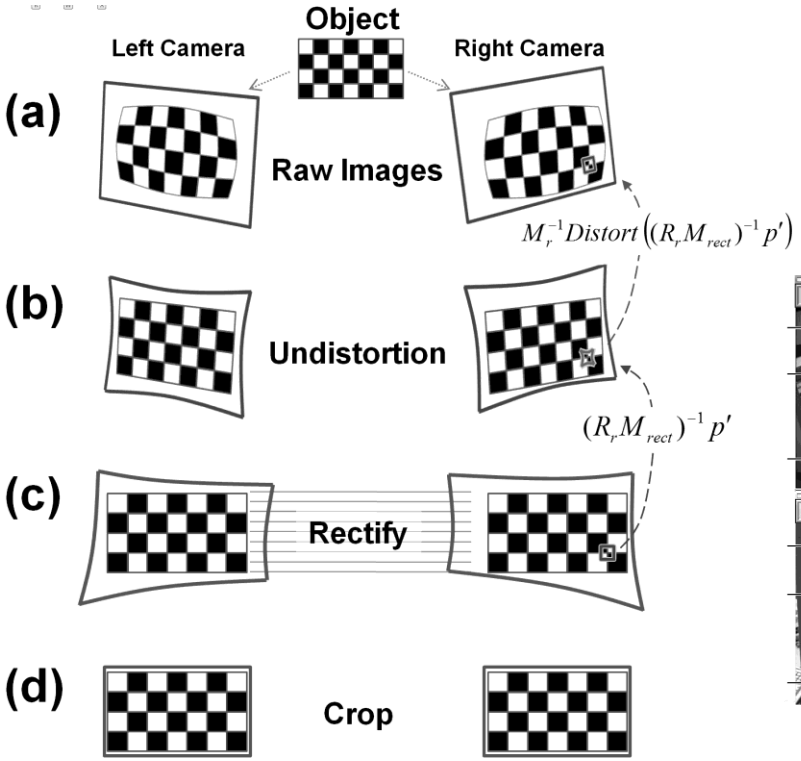
OpenCV Modules: Depth. Pose



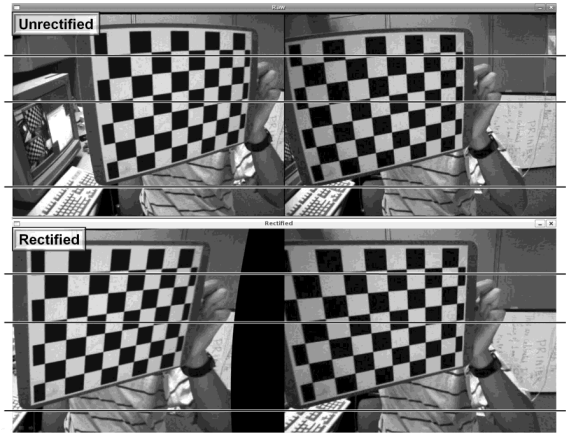
Depth, Pose
Normals, Planes,
3D Features



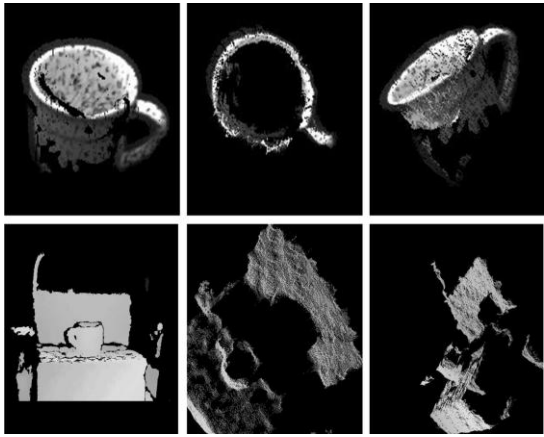




Left – right feature alignment:



Some examples of
3D stereo depth maps:



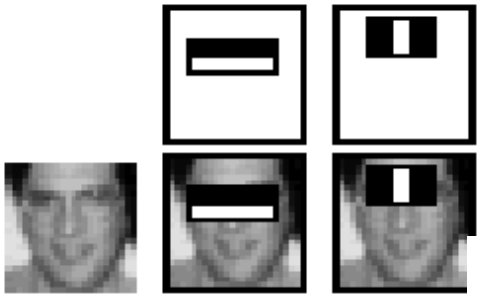
Object recognition
Machine learning

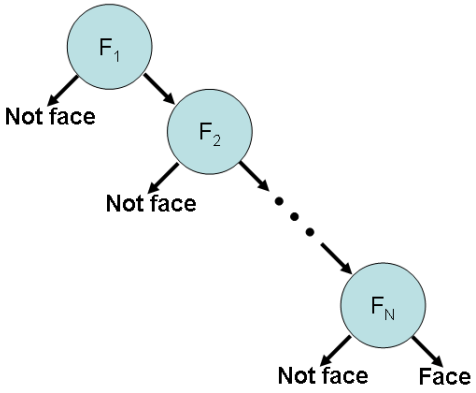
OpenCV Modules: Obj Rec/ML

VSLAM

http://youtu.be/i1uUuWwbIcc

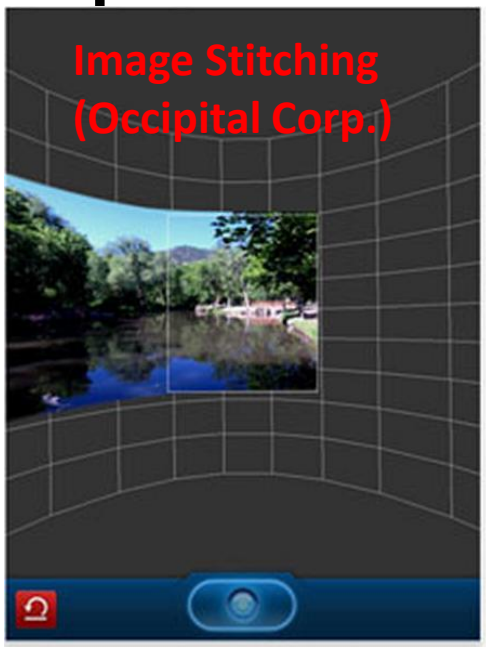
https://www.youtube.com/watch?v=_RF0VpR4xog

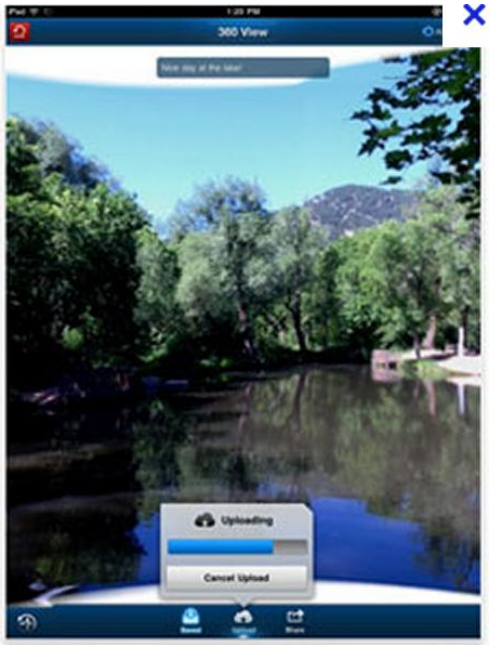


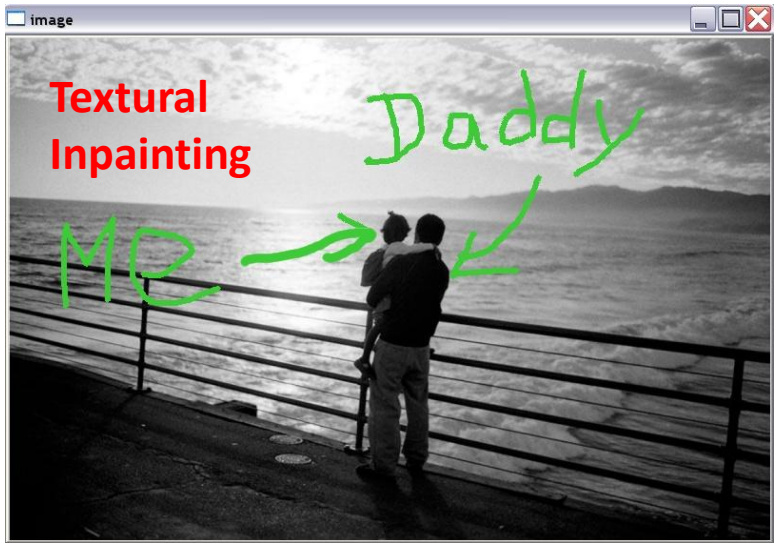


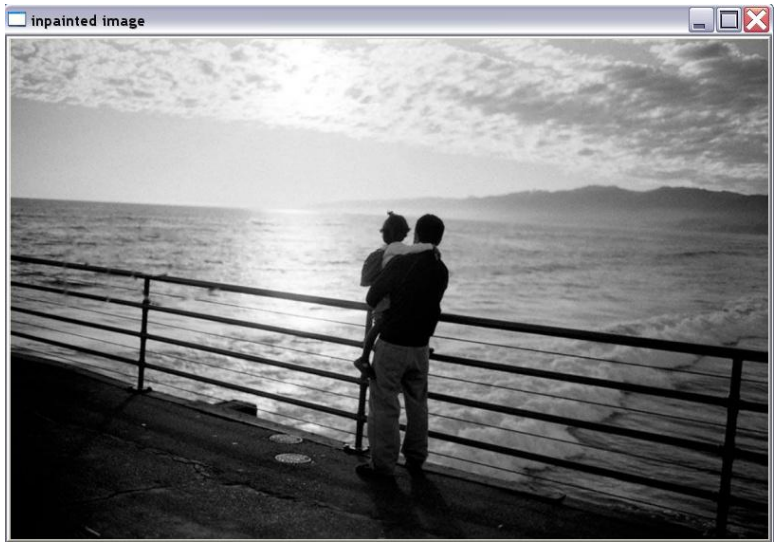
Computational Photography

OpenCV Modules: Comp Photog









Language Modules

- GPU/Cuda
- Android
- iOS
- Python
- Java

OpenCV GPU Module:

- Image processing building blocks:

Color conversions

Geometrical transforms

Per-element operations

Integrals, reductions

Template matching

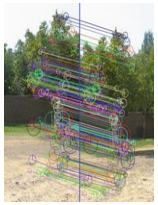
Filtering engine

Feature detectors

- High-level algorithms:

Stereo matching 

Face detection 

Feature matching 

OpenCV GPU Module Example

```
Mat frame;
VideoCapture capture(camera);
cv::HOGDescriptor hog;

hog.setSVMDetector(cv::HOGDescriptor
::
getDefaultPeopleDetectorector());

capture >> frame;




vector<Rect> found;
hog.detectMultiScale(frame, found,
    1.4,  Size(8, 8), Size(0, 0),
1.05, 8);
```

```
Mat frame;
VideoCapture capture(camera);
cv::gpu::HOGDescriptor hog;

hog.setSVMDetector(cv::HOGDescriptor
::
getDefaultPeopleDetectorector());

capture >> frame;

GpuMat gpu_frame;
gpu_frame.upload(frame);

vector<Rect> found;
hog.detectMultiScale(gpu_frame,
found,
    1.4, Size(8, 8), Size(0, 0),
1.05, 8);
```

- Designed very similar!

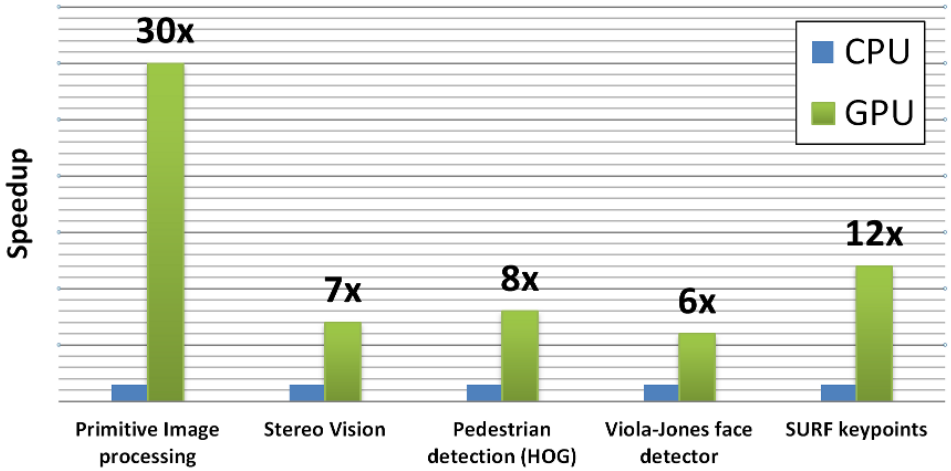
OpenCV GPU Module Performance

Tesla C2050 (Fermi) vs. Core i5-760 2.8GHz (4 cores, TBB, SSE)

- Average speedup for primitives: 33×
  - For “good” data (large images are better)
  - Without copying to GPU

What can you get from your computer?

- opencv\samples\gpu\perfomance



OpenCV Android Module

- **OpenCV 2.4 for Android:**
  - **Native Android Camera Support**
  - **Multithreading**
  - **Java API (soon)**
  - **Tegra HW Optimizations (soon)**

Wiki with the latest information:

http://opencv.org/platforms/android.html

Support/discussion group:::https://groups.google.com/group/android-opencv

OpenCV iOS Module

- In Git repo right now, OpenCV 2.4.2
- Official inclusion in package end of summer.

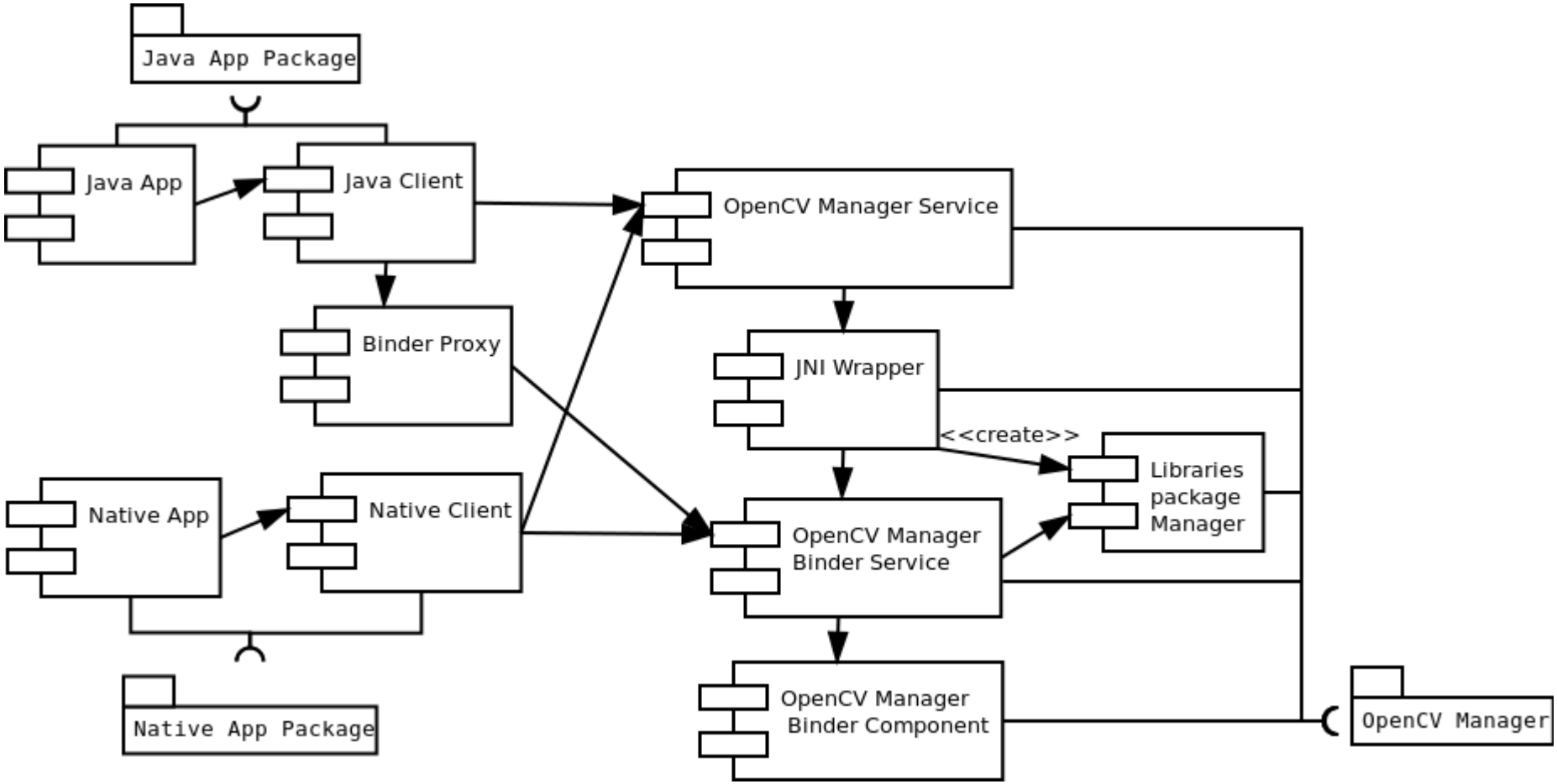
OpenCV Python Module

- Full Python interface
- Example: Depth image from Kinect:

```
import numpy
import cv
from freenect import sync_get_depth as get_depth, sync_get_rgb as
get_video
while True:
(depth,_),(rgb,_)=get_depth(),get_video()
depth=depth.astype(numpy.uint8)
cv.ShowImage("depth",depth)
cv.ShowImage("depth",rgb)
```

Depth image

OpenCV Java Module



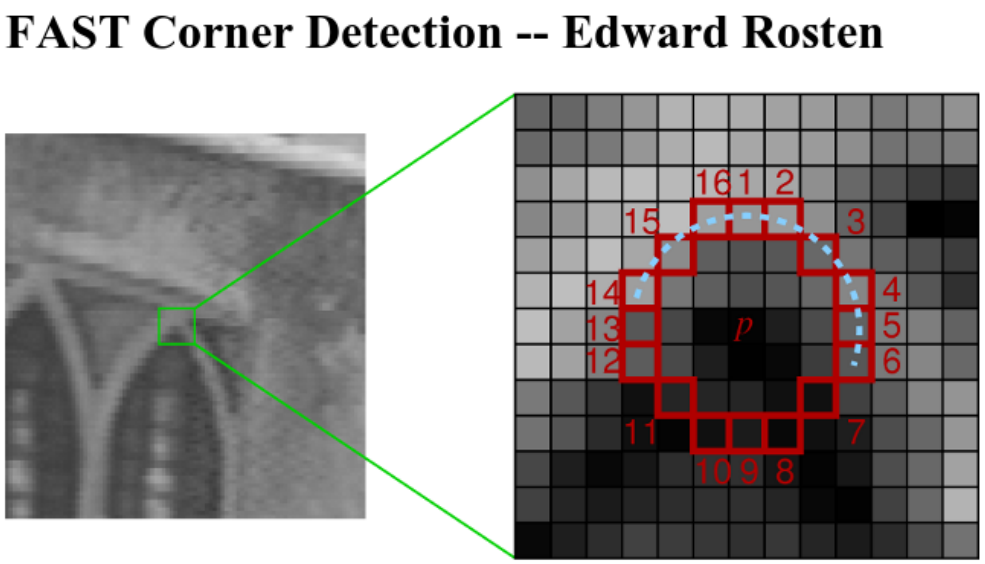
Outline

1. *What is OpenCV?*
2. *The history of OpenCV*
3. *What is in OpenCV? Why Adopt it?*
4. **Case studies**
5. OpenCV and embedded vision
6. The future of OpenCV

Orb

New Feature: ORB

- ORB (Oriented Brief) is a combination of a
  - Fast detector and
  - *Brief descriptor*

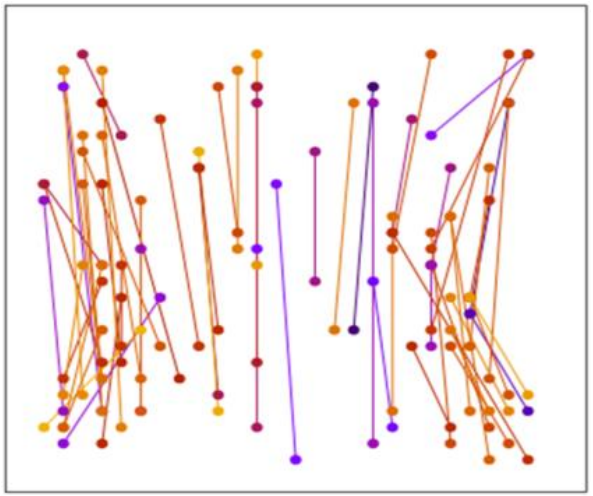


- **FAST**:
  - With reference to a central pixel "*P*" -- interest points are detected as >= 12 contiguous pixel brighter than *P* in a ring of radius 3 around *P*.

Edward Rosten and Tom Drummond, "Fusing points and lines for high performance tracking", ICCV 2005

New Feature: ORB

- ORB (Oriented Brief) is a combination of a
  - *Fast detector and*
  - Brief descriptor



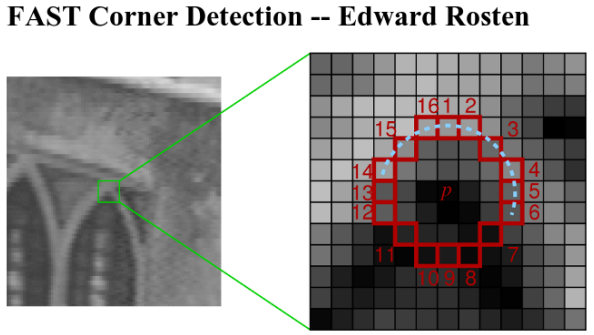
- **BRIEF**:
  - Create an integral image for rapid summation of patches
  - In a 31x31 area round an interest point,
  - Randomly create 256 9x9 pairs patches, call them A_i,B_i
  - For each pair, if , then set the corresponding bit to 1, else 0
  - The resulting 256 bit vector is the descriptor for the patch

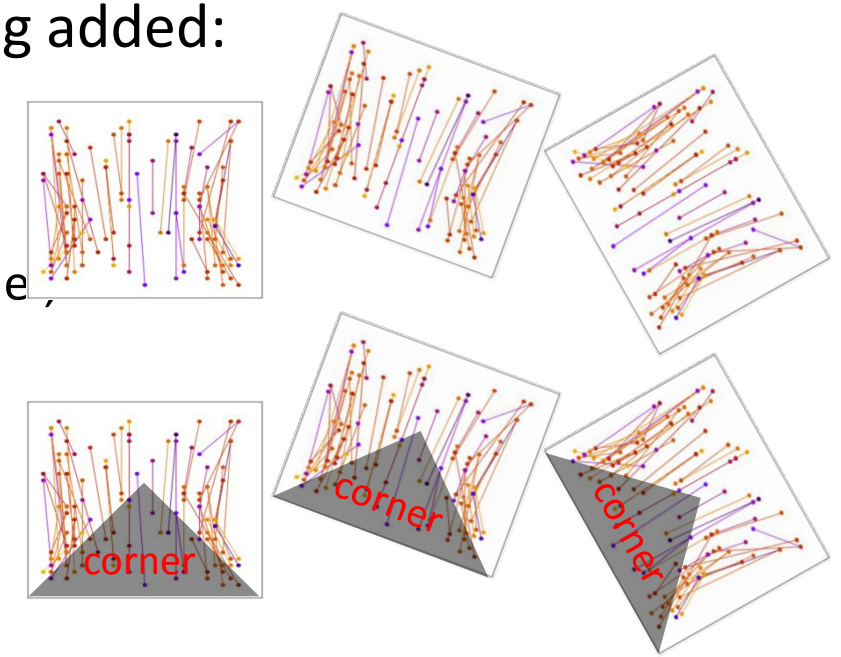
Calonder M., Lepetit V., Strecha C., Fua P.: BRIEF: Binary Robust Independent Elementary Features. ECCV 2010

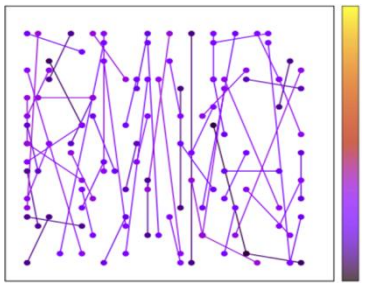
New Feature: ORB

E. Rublee, V. Rabaud, K. Konolidge, G. Bradski, "ORB: an efficient alternative to SIFT and SURF". ICCV 2011

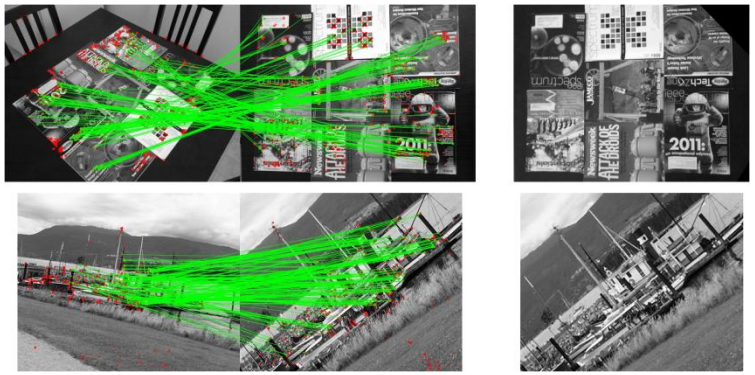
- ORB is a combination of
  - Fast detector – with orientation added
    - We used local image moments
  - Brief descriptor – with steering added:
    - Problem with steering to a corner,
    - oops, it becomes correlated:
- Finally, we de-correlated:
  - We enumerated 100Ks of pair combinations and greedily chose de-correlated ones.



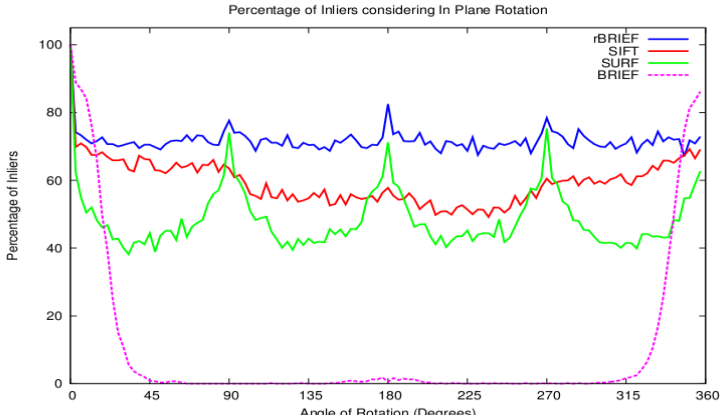




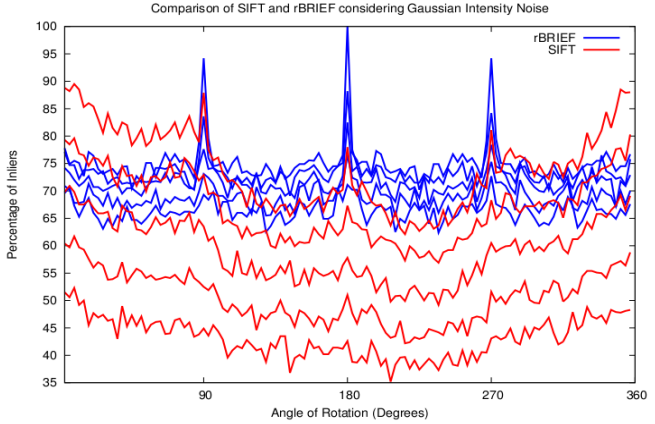
New Feature: ORB



- Performance
  - Speed 100x faster than SIFT. 10x Faster than SURF
- Viewpoint invariance



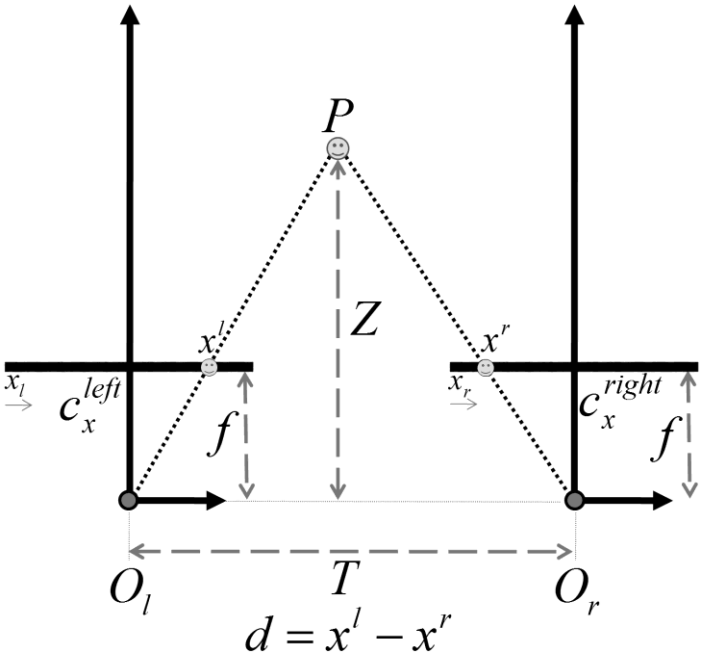
- Noise tolerance



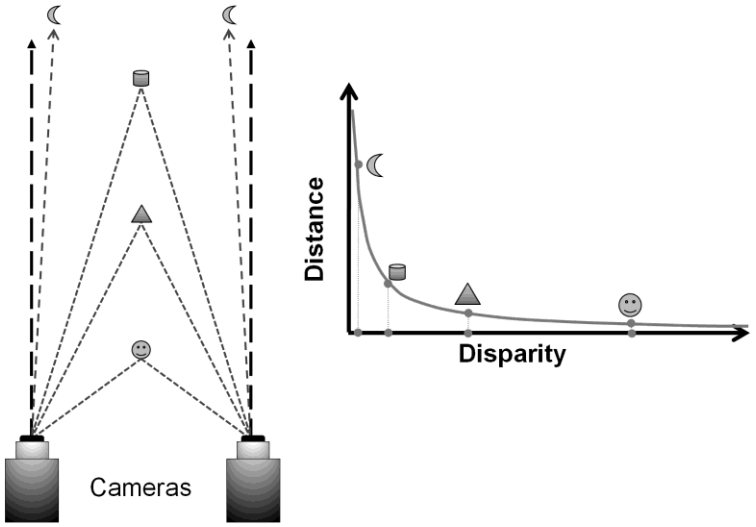
E. Rublee, V. Rabaud, K. Konolidge, G. Bradski, "ORB: an efficient alternative to SIFT and SURF". ICCV 2011

Stereo … Depth from Triangulation

- Involved topic, here we will just skim the basic geometry.
- Imagine two perfectly aligned image planes:

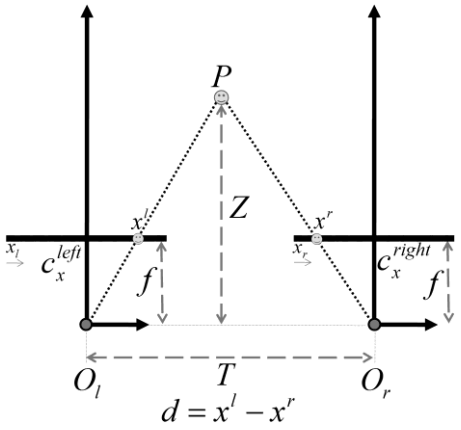


Depth “Z” and disparity “d” are inversely related:



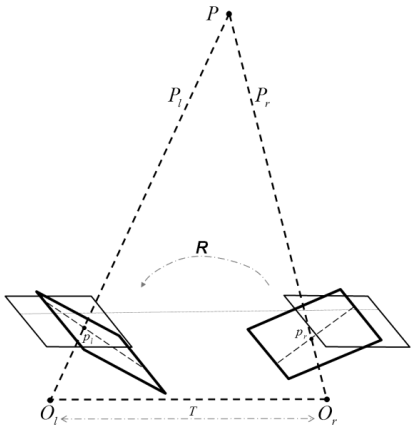
Stereo

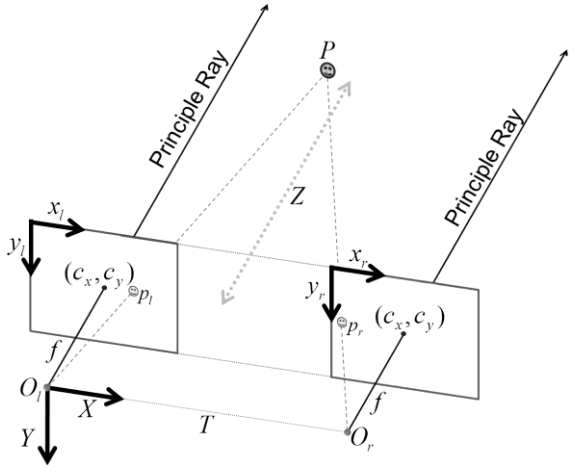
- In aligned stereo, depth is from similar triangles:



$$\frac{T - (x^l - x^r)}{Z - f} = \frac{T}{Z} \Rightarrow Z = \frac{fT}{x^l - x^r}$$

- Problem: Cameras are almost impossible to align
- Solution: Mathematically align them:

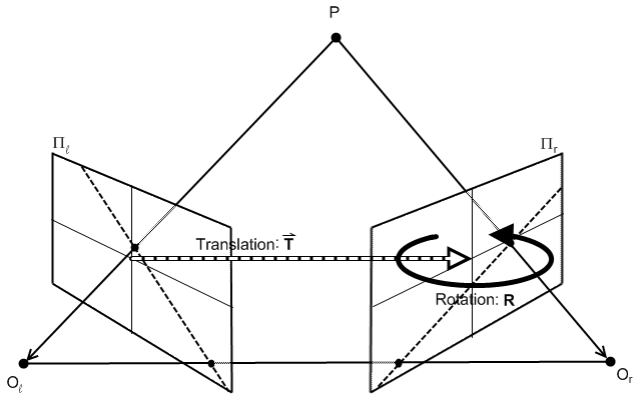




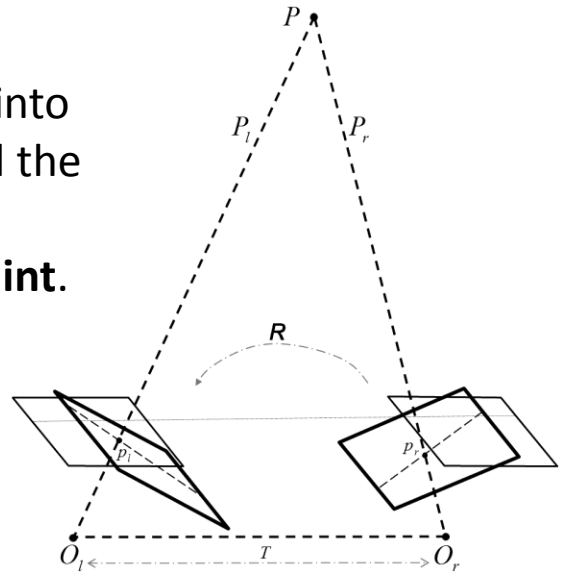
All: Gary Bradski and Adrian Kaehler: Learning OpenCV

Stereo Rectification

- We again collect many images of chessboards and use them to:
  - Calibrate the left and right cameras
  - Find the rotation and translation between them
  - Mathematically align the cameras.



Goal:
- Any given world point falls into the same row on the left and the right cameras.
- Called the **Epipolar Constraint**.

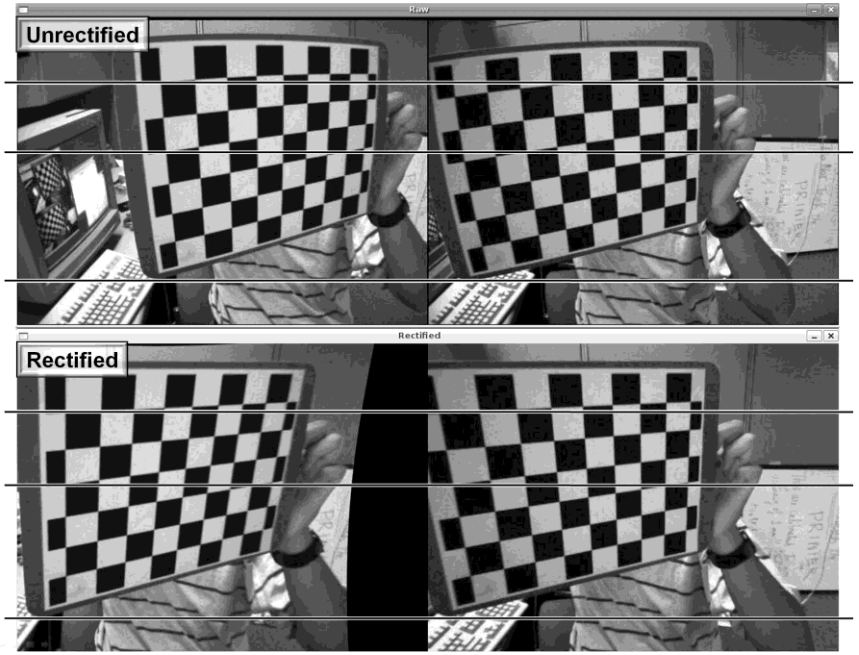


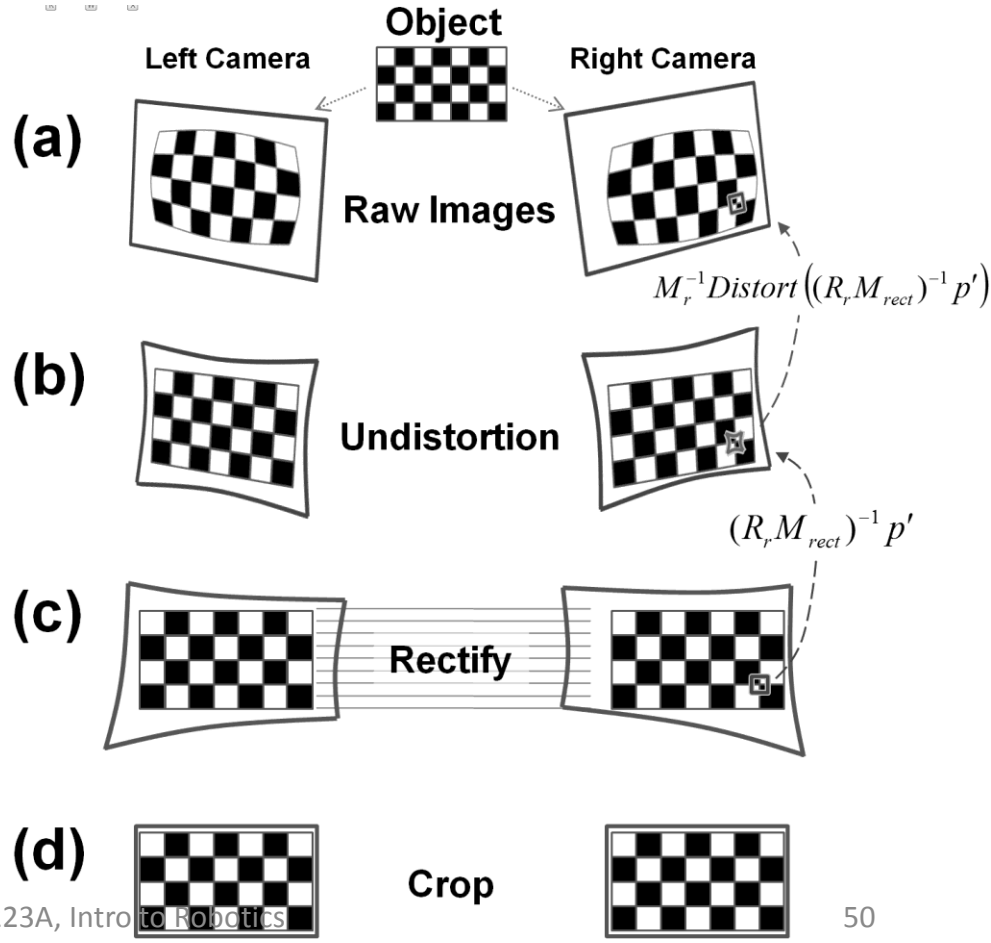
All: Gary Bradski and Adrian Kaehler: Learning OpenCV

Stereo Rectification

- Algorithm steps are shown at right:
- Goal:
  - Each row of the image contains the same world points
  - "Epipolar constraint"

Result: Epipolar alignment of features:



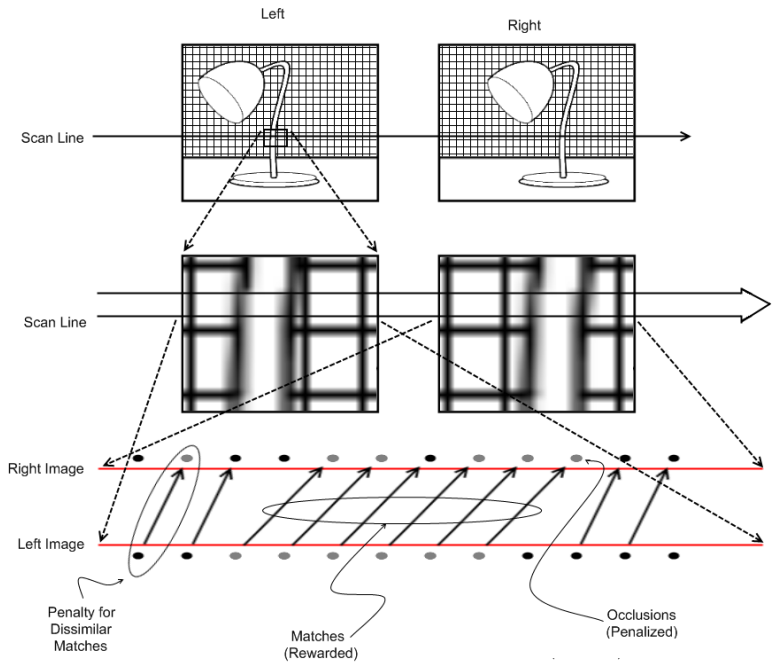


All: Gary Bradski and Adrian Kaehler: Learning OpenCV

Stereo Correspondence
(Epipolar Aligned Stereo)

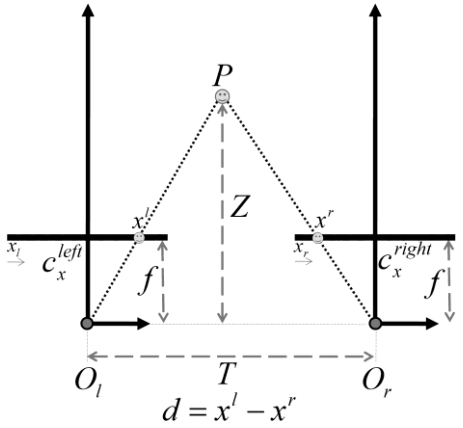
- Row by row we search for matching points
- This yields a disparity map "*d*":

Disparity Map (one row here)



Depth Map:

With known baseline "*T*", we get depth "*Z*":



$$\frac{T}{Z}(x^l - x^r) = \frac{T}{Z} \Rightarrow Z = \frac{fT}{x^l - x^r}$$

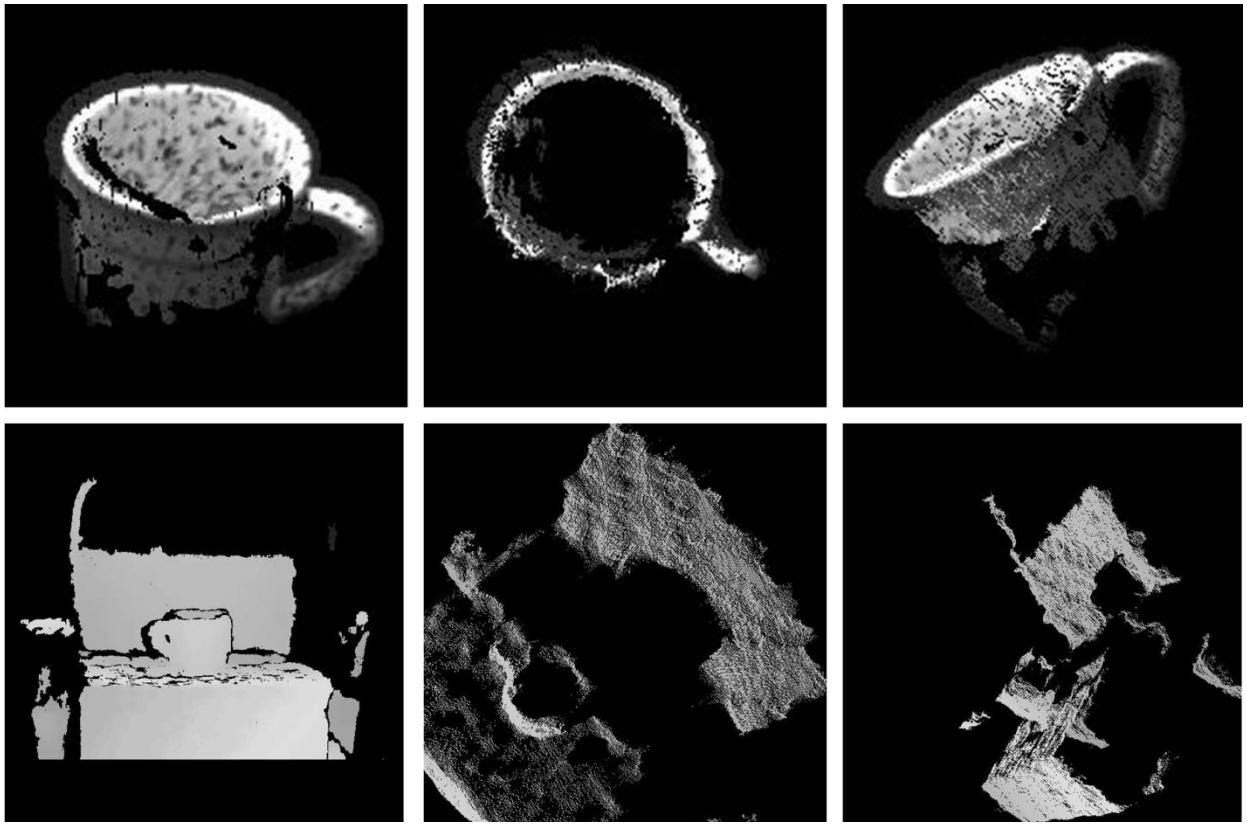
All: Gary Bradski and Adrian Kaehler: Learning OpenCV

Stereo **Code**

- The entire stereo processing pipeline is implemented in OpenCV.
- See the sample code file in the directory
- **…/opencv/samples/c/stereo_calib.cpp**

Stereo Example

- 3D perception



Gary Bradski and Adrian Kaehler: Learning OpenCV

Robots need Recognition and Pose

- In order to understand and grasp objects, robots need to recognize objects and object categories, but also
- Robots need to know the pose of objects in 6DOF relative to the robot coordinate frames.
- We can get pose and recognition in 2D and 3D

3D Pose and Recognition w/ Feature Detector Descriptors

- **Given that we know the feature locations on an object,**
  - We can use nearest neighbors or "bag of words" of features to vote for likely objects.
- **For each likely object**
  1. We can run Ransac on a subset of feature points and use homography (for planar objects) or PnP (for 3D objects) to determine a proposed pose for the proposed object.
  2. We use the proposal object pose to project the rest of the features from the model to the image and see if the correspond ("inlier points").
  3. High percentage of inlier points => we have correctly recognized this object … and already then have its pose.

Romea et. al.'s MOPED uses 2D SIFT features on textured object to get pose and recognition

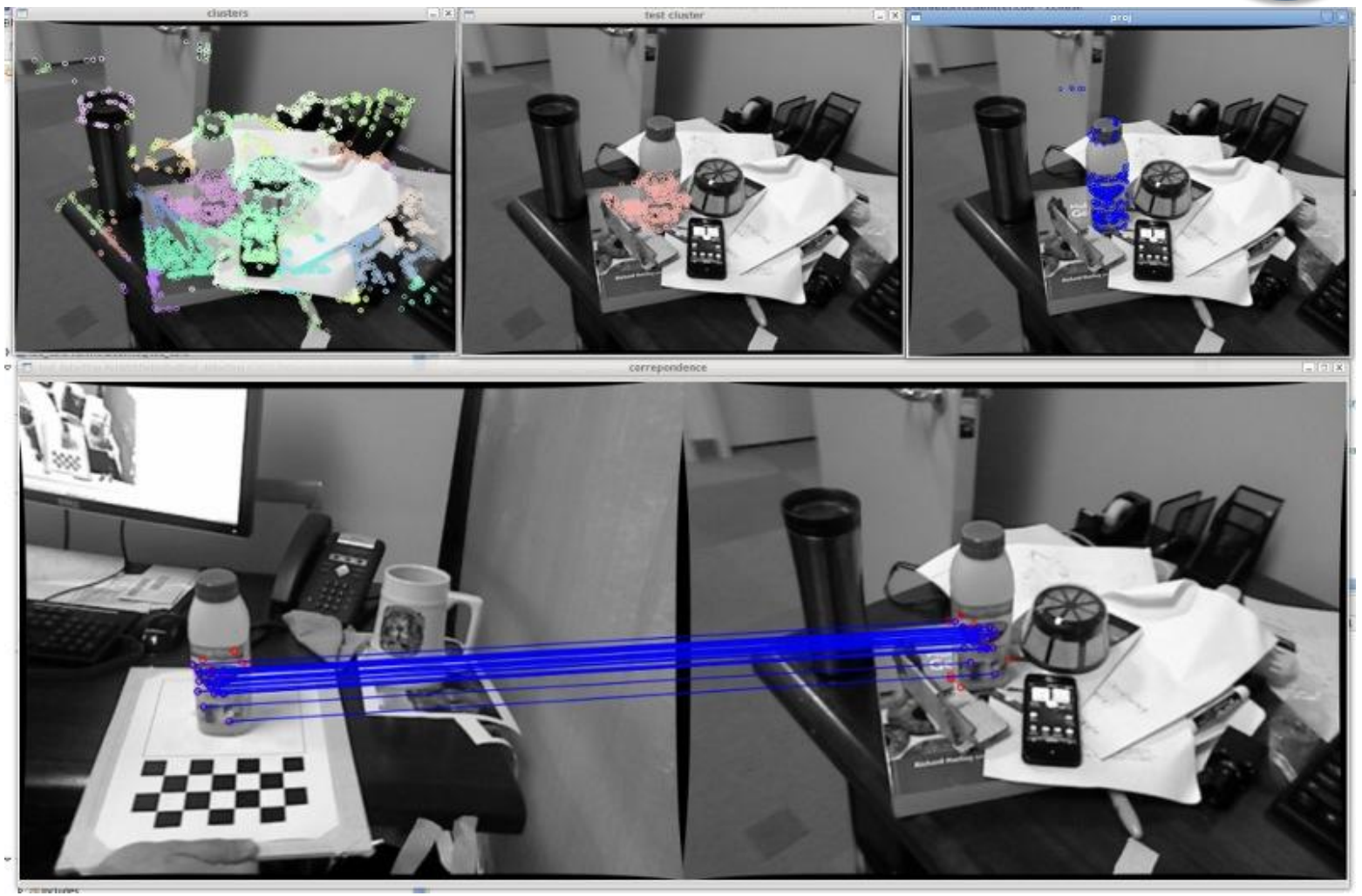
Alvaro Collet Romea, *Dmitry Berenson*, *Siddhartha Srinivasa*, *and* *David Ferguson*, Object Recognition and Full Pose Registration from a Single Image for Robotic Manipulation, ICRA 2009

OpenCV Code: Not yet, but in process and adding:

- Allowance for stereo
- 3D model capture
- Finding flexible objects

Textured Object Recognition (TOD)

http://www.youtube.com/watch?v=TFKQepLXNZs



Outline

1. *What is OpenCV?*
2. *The history of OpenCV*
3. *What is in OpenCV? Why Adopt it?*
4. *Case studies*
5. **OpenCV and embedded vision**
6. The future of OpenCV

Embedded Vision

- ARM: Many ports
  - TI, Android, iOS
- An optimized version for ARM+Neon is needed
  - Resources … OpenCV foundation
  - There are **contract programmers** available for ports, talk to me
- We use quad core Intel chips in a fit-PC3 Pro 4GB industrial package.

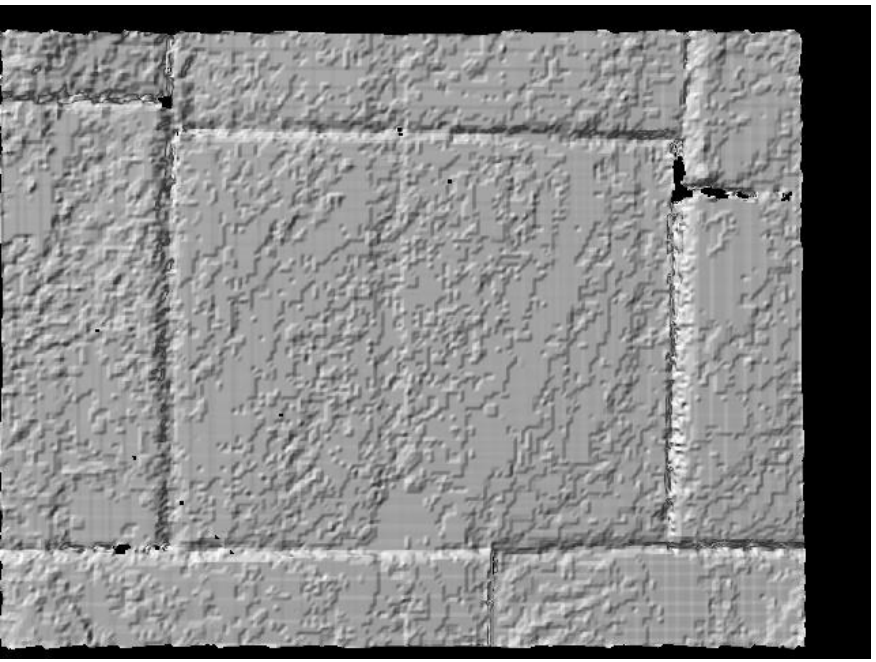
Embedded Example Industrial Perception: **Planar Navigation**

- >> Real time surface understanding for navigation and obstacle detection.

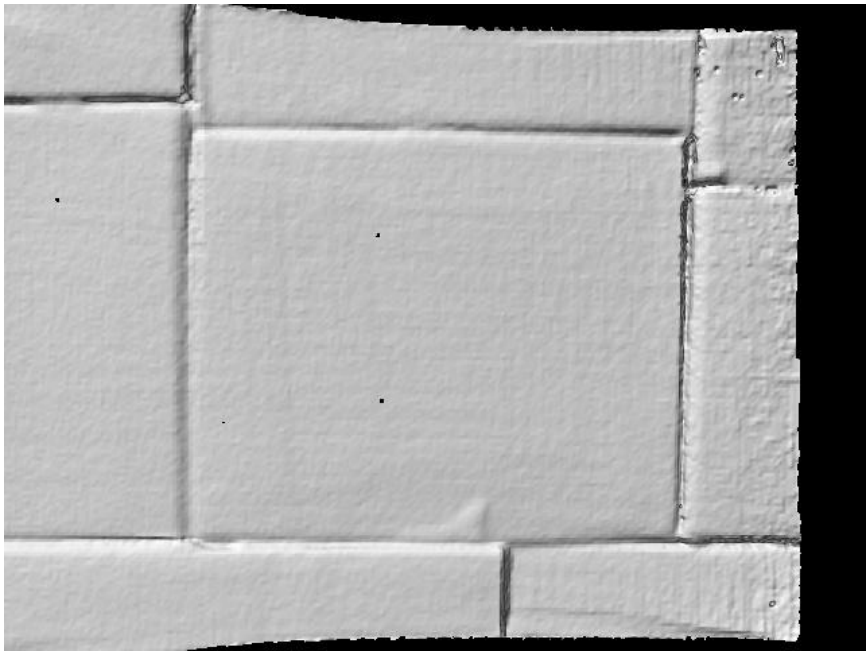
Embedded Example Industrial Perception:

3D Hyper-resolution for box hunting

Raw Sensor Data:



Stabilized Integration on a Surface:



RUN

https://www.youtube.com/watch?v=1DBhouen424

Embedded Example Industrial Perception:

Box Unloading

- Use perception + manipulation + planning to unload trucks and shipping containers.

Box Unloading

https://www.youtube.com/watch?v=bJOTqjRdgSA

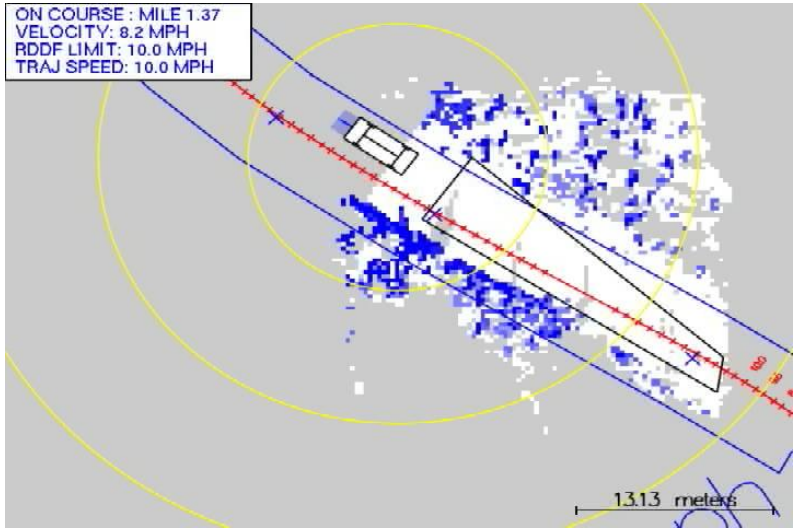
In Transportation: DARPA Grand Challenge:

Stanley Won the $2M Autonomous Race Across the Desert

Fusing Sensing in a Bird's Eye World Model:

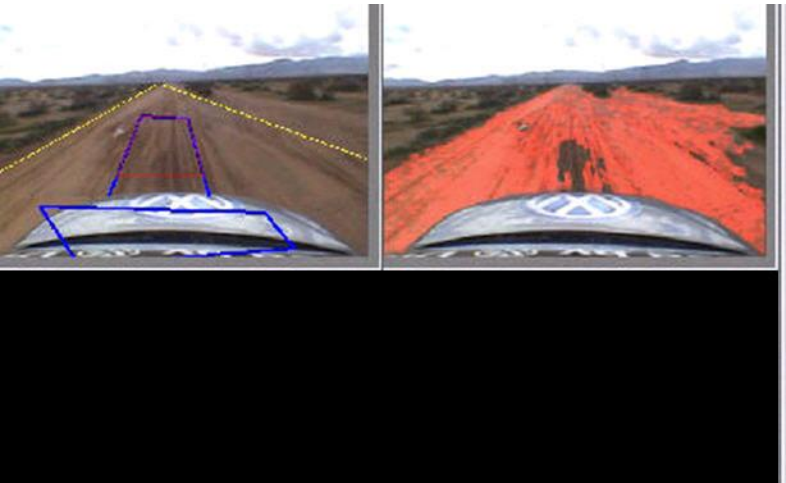
http://youtu.be/4-inw2Tw_zo

http://youtu.be/R9U2i6bjgII

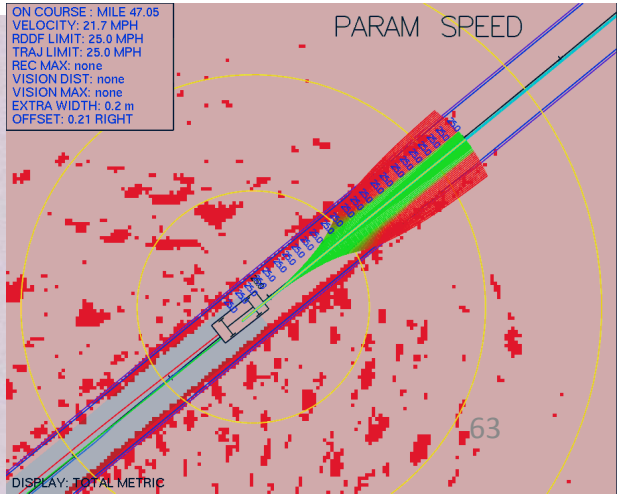


http://youtu.be/JM6fBtE2xK0

Fuse laser with camera:



Planning in world model:



Transportation:

OpenCV=>Stanley=>Tech:

- Google Robot Car

- Streetview

- Book scan

Outline

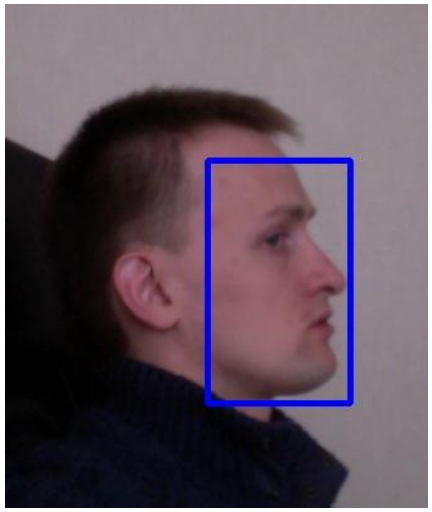
1. *What is OpenCV?*
2. *The history of OpenCV*
3. *What is in OpenCV? Why Adopt it?*
4. *Case studies*
5. *OpenCV and embedded vision*
6. **The future of OpenCV**

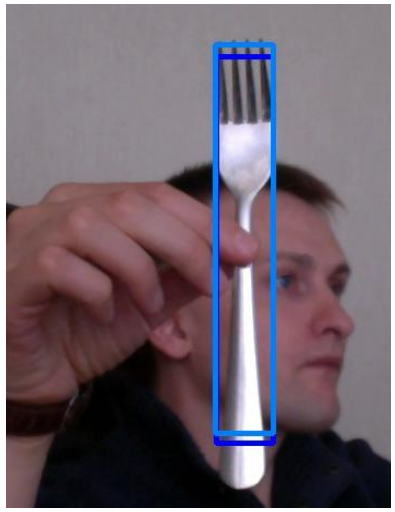
Recent Infrastructure in OpenCV

- A whole new infrastructure:
  - user site opencv.org
  - developer zone: code.opencv.org. We now use very convenient chiliproject bug tracker (descendant of redmine)
  - the new GIT repository, which we moved to from SVN.
  - the GIT mirror at github.com, which we use for pull requests. github.com/opencv was taken by some cybersquatter, so we used github.com/itseez/opencv.
  - autogenerated documentation: docs.opencv.org
  - StackOverflow-like answers site: answers.opencv.org
  - enhanced and extended buildbot: build.opencv.org.
    - It now includes over 40 different builders, testing OpenCV in various configurations on different OSes, mobile and desktop.
    - There is also "binary compatibility builder" that checks binary compatibility of the current snapshot against the latest OpenCV stable release (2.4.2), "documentation builder" that builds reference manuals and uploads them to docs.opencv.org,
    - A "windows superpack builder" that builds installation packages for Windows, Android and iOS packages builders etc.
- Several important functions have been optimized using SSE instructions and/or by employing parallel processing:
  - bilateral filter (by factor of >3x on dual-core Core i5)
  - image warping, remap and image resize (up to 12x speedup)
  - mixture-of-Gaussian background subtraction (~3x faster)
- 400 Bugs fixed

Recent Functionality in OpenCV

- The following major new functionality
  - **face recognition** (contributed by Philipp Wagner)
  - **FREAK keypoint descriptor** (from EPFL lab)
  - **GMG background subtractor** (contributed by A. B. Godbehere)
  - **video stabilization** module (by OpenCV NVidia team)
  - enhanced **LogPolar transform**
  - **OpenFABMAP** image recognition algorithm (for image retrieval)
  - **Better solvePnP** algorithms 2D points to 3D pose (implementations of EPFL algorithms)
  - From Google Summer of Code
    - image denoising,
    - dense optical flow,
    - 2 new object detectors,
    - OpenCV iOS port,
    - more C++ samples,
    - Python samples.

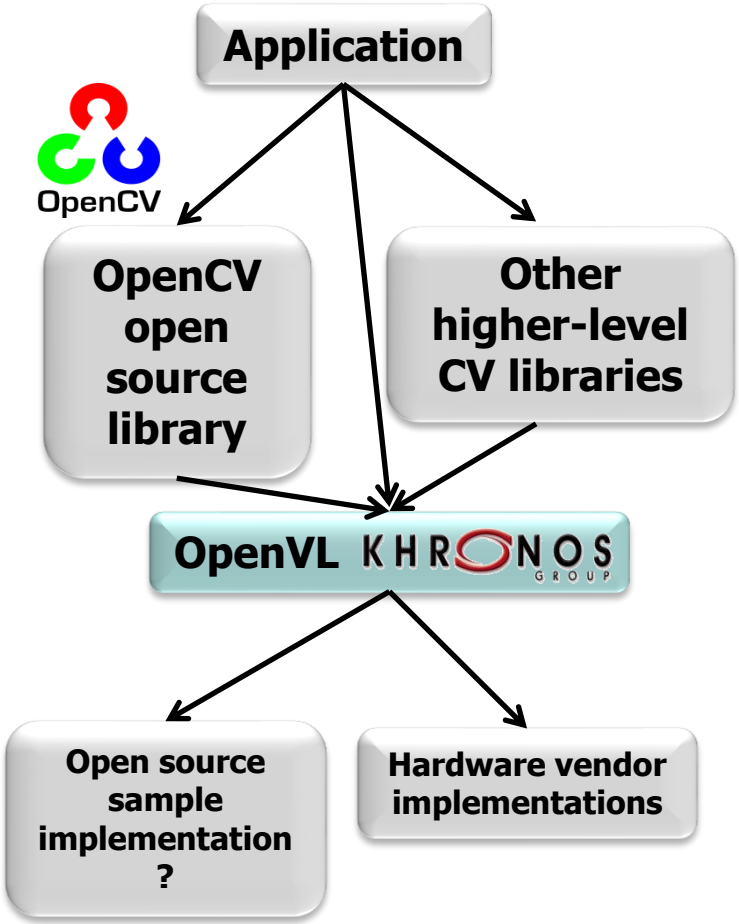




Cascade: Side face, and silverware

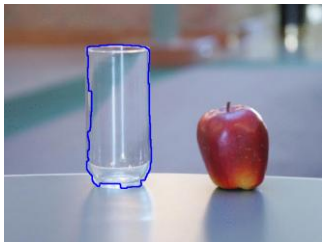
OpenCV Helping Drive the New Khronos Standard: **OpenVL**

- **Vision Hardware Acceleration Layer**
  - Enable hardware vendors to implement accelerated imaging and vision algorithms
- **OpenVL can be used by high-level libraries or applications directly**
  - Primary focus on enabling real-time vision apps on mobile and embedded systems
- **OpenCV is widely used open source library for vision projects**
  - Future version will leverage OpenVL
- **Working group aiming for stable draft spec in 2012**
- **Itseez actively participates in the development of OpenVL**

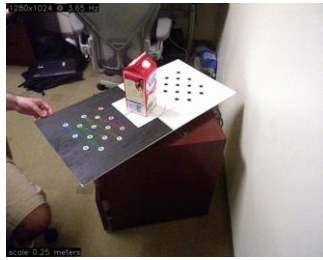


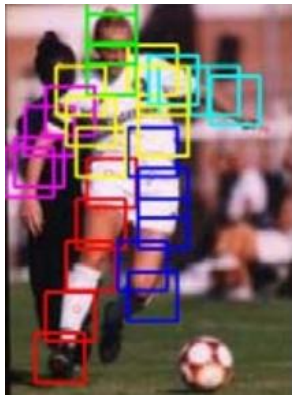
Coming Highlights in OpenCV

- Faster releases 4x-6x/year
- Cloud Support (python on Amazon servers)
- Revamped mathematical framework for detectors and descriptors:
  - Faster and way more accurate
- Depth motion fusion



- Iris Recognition
- Transparent item ID
- ARM Optimization(?)
- 3D model training
- 2D barcodes
- 2D line matching
- Parts from whole





- More modular
- More optimized

User: http://opencv.org; **Developer: http://code.opencv.org**

OpenCV Now a non-profit: OpenCV.Org

- **MISSION:** Advance all aspects of educational, commercial and research computer vision
  - *By leveraging an open and free, industrial quality, optimized coding infrastructure*.
- Please join, contribute, **and support!**
- We will sponsor workshops, tutorials, solution code sprints, prizes, crowd sourcing, solutions in perception and cloud infrastructure.
- **Contact:** garybradski@gmail.com

Contribute (via Credit, debit or PayPal):
http://tinyurl.com/7eujyo2

User: http://opencv.org; Developer: http://code.opencv.org

Useful OpenCV Links

Developer OpenCV Site:
http://code.opencv.org

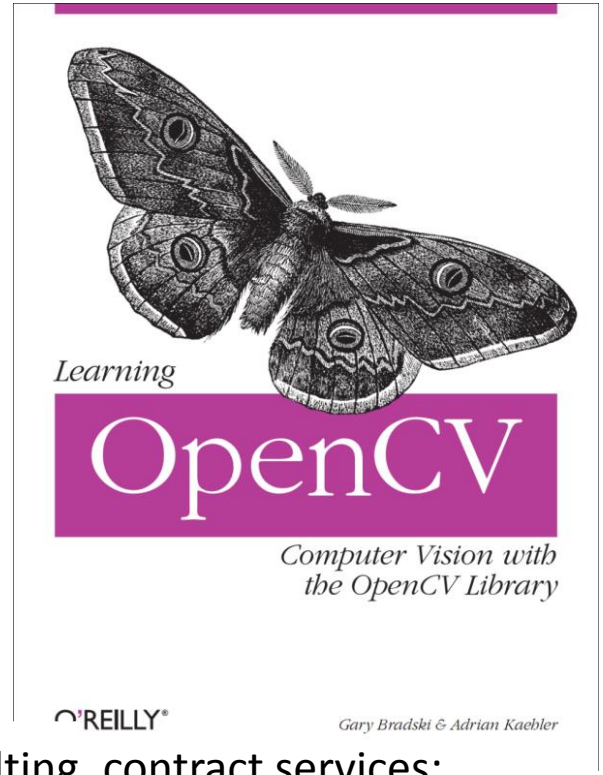
User Group:
http://tech.groups.yahoo.com/group/OpenCV/join

User OpenCV Site:
http://opencv.org

Best seller in Computer Vision and Machine Learning for 3 years.
Version 2 coming this summer.

Book on OpenCV:
http://oreilly.com/catalog/9780596516130/

Or, direct from Amazon:
http://www.amazon.com/Learning-OpenCV-Computer-Vision-Library/dp/0596516134



Code examples from the book:
http://examples.oreilly.com/9780596516130/

Version 2 of the book is coming Q4, 2012

For high level issues, partnering, financial contributions, consulting, contract services:
Contact: garybradski@gmail.com

Photo: Gary Bradski