Willow Garage, OpenCV, ROS, And Object Recognition

ICRA Semantic Perception Workshop

Gary Bradski

garybradski@gmail.com

Willow Garage

Outline

- What's Willow Garage
- Perception is Hard
- Open Source Computer Vision Library (OpenCV)
- Point Cloud Library (PCL)
- Current Research Results
- (if time) Speculations on Perception

Willow Garage

What is Willow Garage?

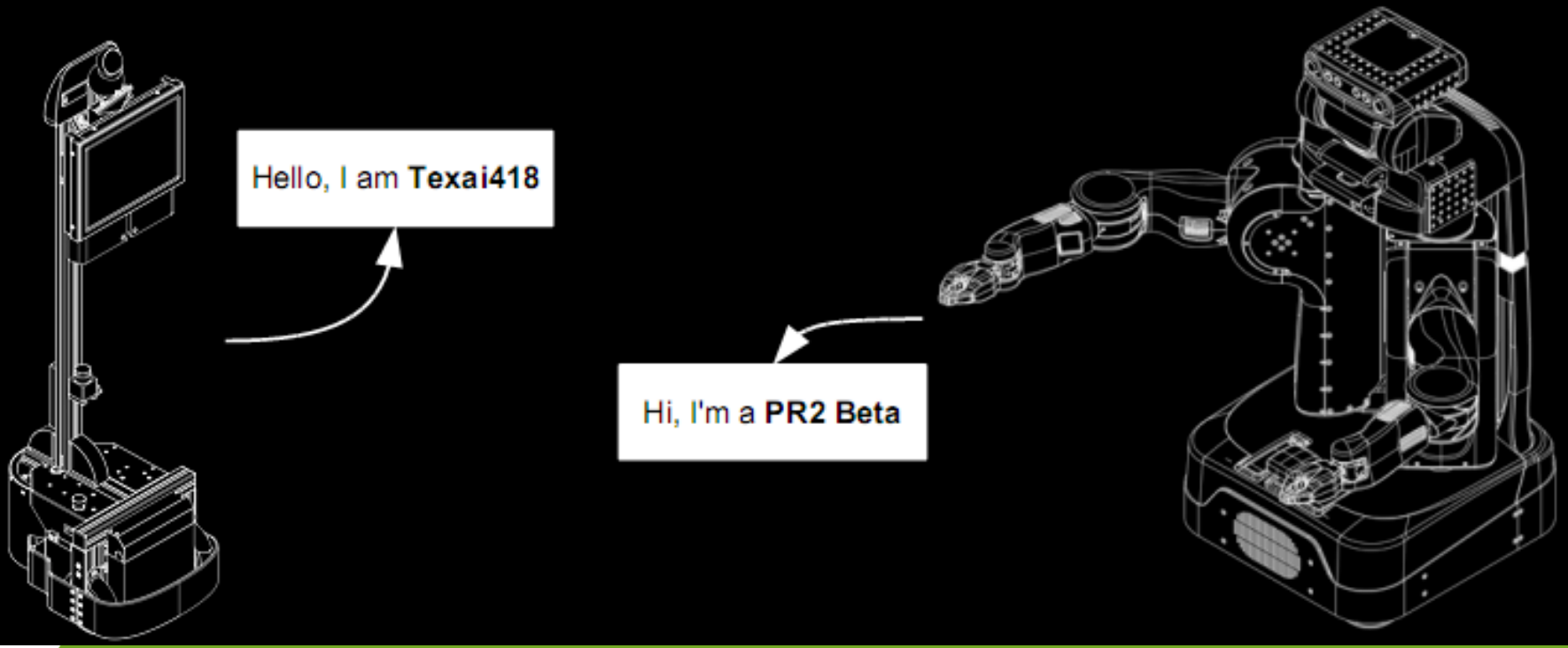
- It's a privately funded robotics institute/incubator
- Mission to revolutionize civilian robotics
  - Not just companies, but the whole industry
    - Strong support of open source: ROS, OpenCV, PCL
  - Spin off companies and products
- Focus is on sensor based adaptive robots

Willow Garage

Willow Garage

What is ROS?

- Meta operating system for robotics
- Obtain, build, write, and run code across multiple computers, and multiple robots



Software Products

- ROS (Robot Operating System)
  - http://www.ros.org

- OpenCV (Open Source Computer Vision Lib)
  - http://opencv.willowgarage.com/wiki/

- PCL (Point Cloud Library)
  - http://www.pointclouds.org/

Willow Garage

Hardware Products

PR2s,

Turtlebots,

Spinouts

Willow Garage

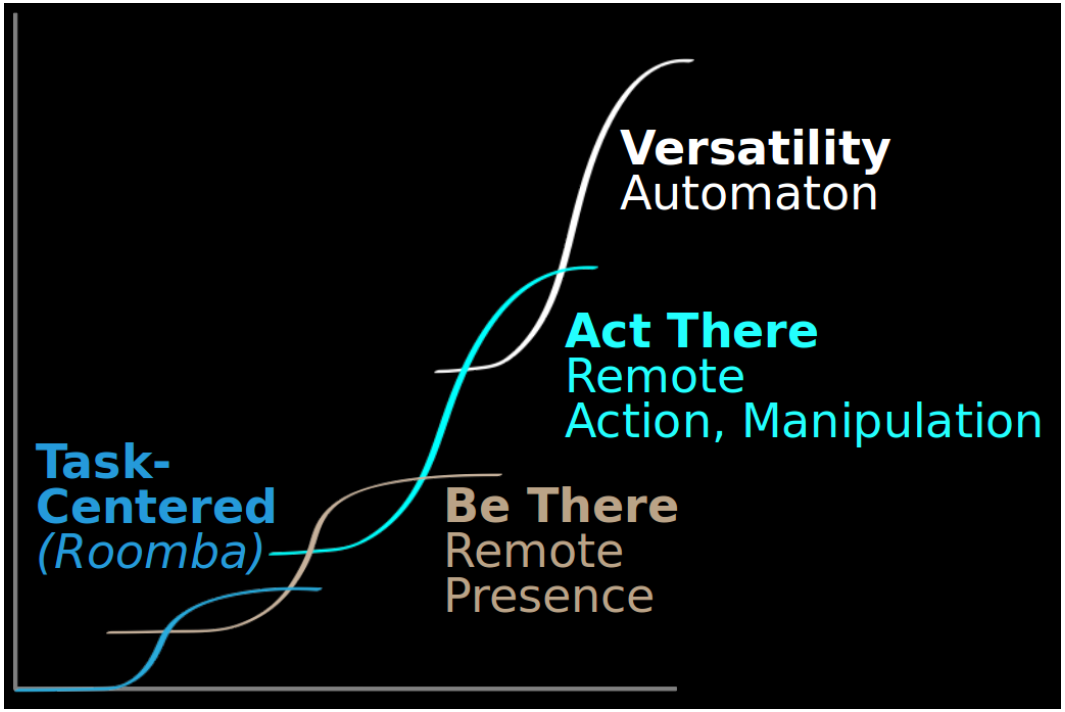
Working On

Software:

- Higher level object and scene recognition
- People, pose and tracking
- Perception Apps Store

Hardware:

- “ROS Arm”: Capable but cheap
- Capable but cheap 2D+3D Sensing



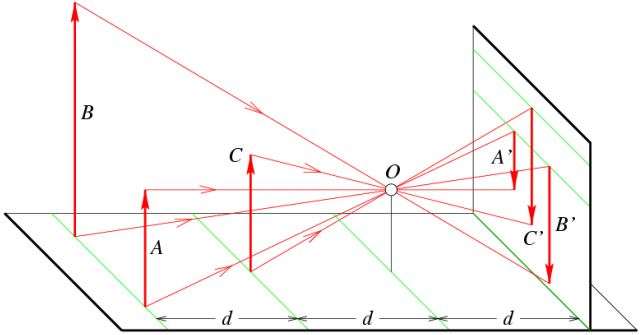
Willow Garage

Outline

- *What's Willow Garage*
- Perception is Hard
- Open Source Computer Vision Library (OpenCV)
- Point Cloud Library (PCL)
- Current Research Results
- (if time) Speculations on Perception

Willow Garage

Vision is Hard



- What is it?
  - Turning sensor readings into perception.
- Why is it hard?
  - It's just numbers.

We perceive this:

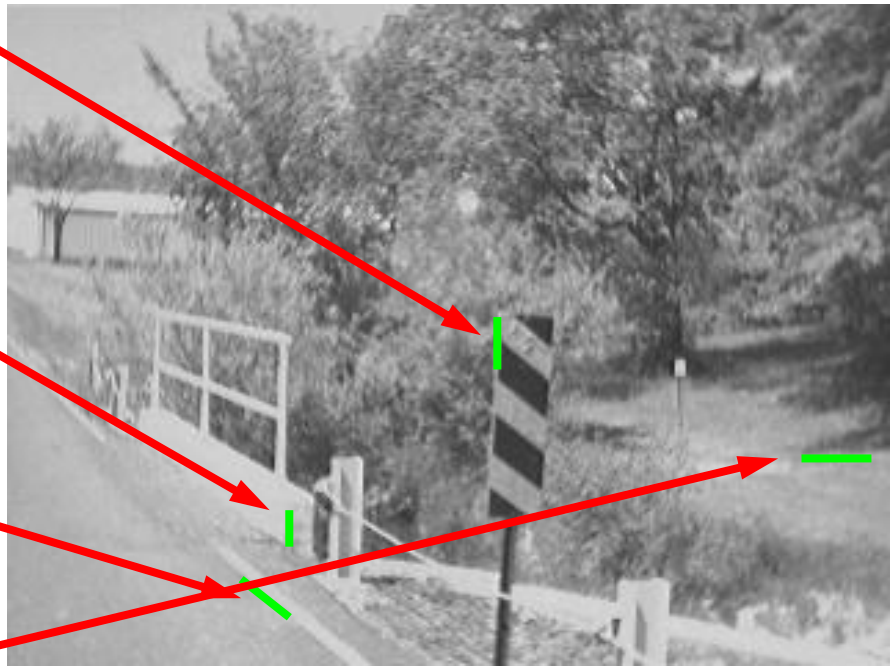
Maybe try gradients to find edges?

But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Use Edges? … It's not so simple

- Depth discontinuity
- Surface orientation discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)
- Illumination discontinuity (e.g., shadow)



Slide credit: Christopher Rasmussen

To Deal With the Confusion, Your Brain has Rules... That can be wrong

OpenCV's purpose is to help turn "seeing" into perception.

We will also use active depth sensing to "cheat".

11

Outline

- *What's Willow Garage*
- *Perception is Hard*
- Open Source Computer Vision Library (OpenCV)
- Point Cloud Library (PCL)
- Current Research Results
- (if time) Speculations on Perception

Willow Garage

OpenCV: Open Source Computer Vision Library

- Launched in 1999 while I was at Intel Corp.
  - **Purpose:** To advance computer vision by creating a comprehensive, mostly real time infrastructure available to all.
- Free and Open Source, BSD license
- 3.5M downloads
- 45K member user group
- Supported by Willow Garage, Nvidia, Google
- Learning OpenCV book by O'Reilly has been the best seller in Computer Vision and Machine Learning for 3 years now.

Willow Garage

OpenCV Overview:

opencv.willowgarage.com

> 2000 algorithms

General Image Processing Functions

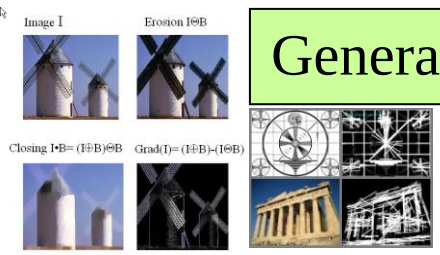
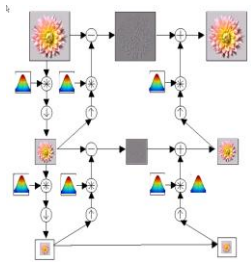
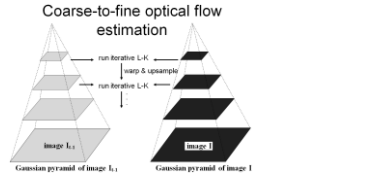
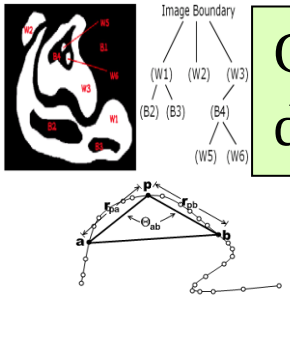


Image Pyramids

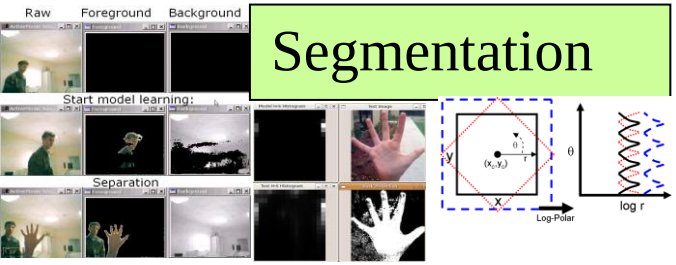




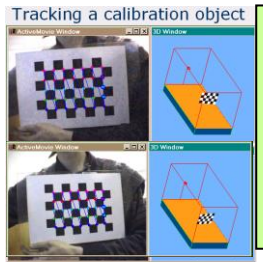
Geometric descriptors



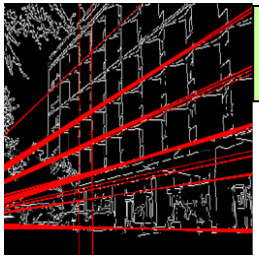
Segmentation

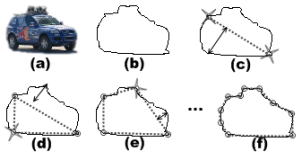


Camera calibration, Stereo, 3D

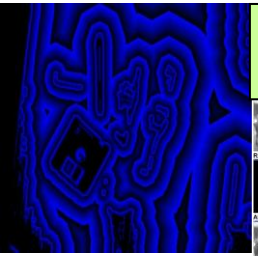


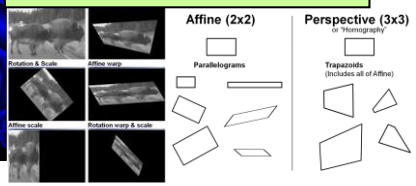
Features



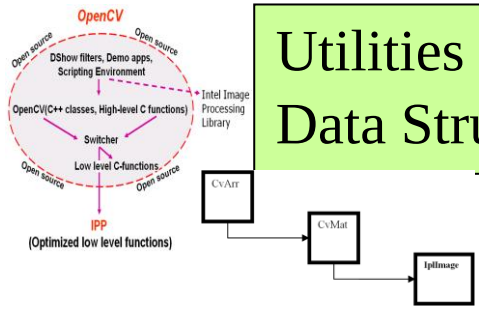


Transforms

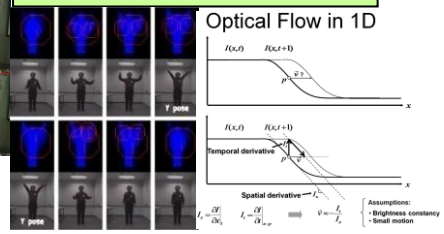




Utilities and Data Structures

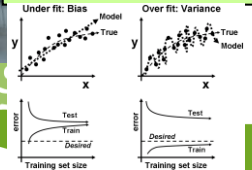


Tracking

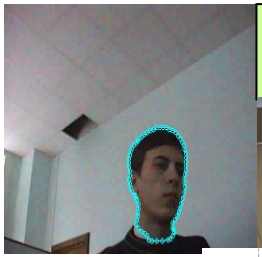


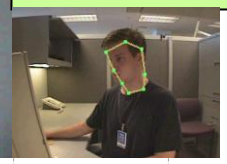
Machine Learning:

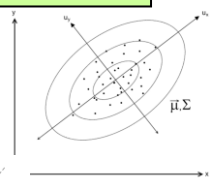
- Detection,
- Recognition

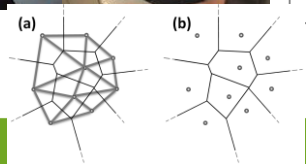


Fitting

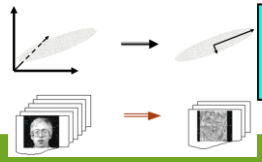








Matrix Math



Gary Bradski

Machine Learning Library (MLL)

CLASSIFICATION / REGRESSION

(new) Fast Approximate NN (FLANN)

(new) Extremely Random Trees

CART

Naïve Bayes

MLP (Back propagation)

Statistical Boosting, 4 flavors

Random Forests

SVM

Face Detector

(Histogram matching)

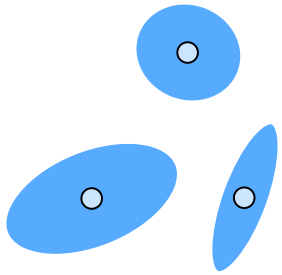
(Correlation)

CLUSTERING

K-Means

EM

(Mahalanobis distance)



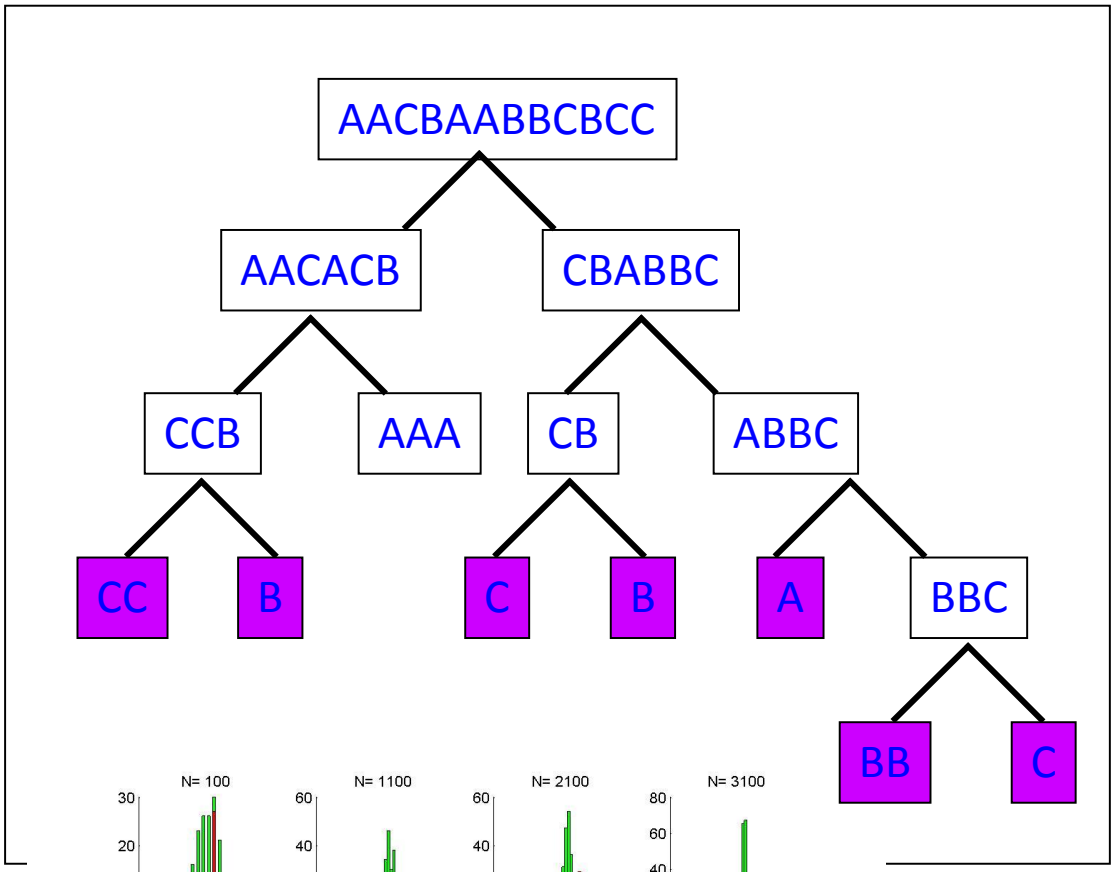
TUNING/VALIDATION

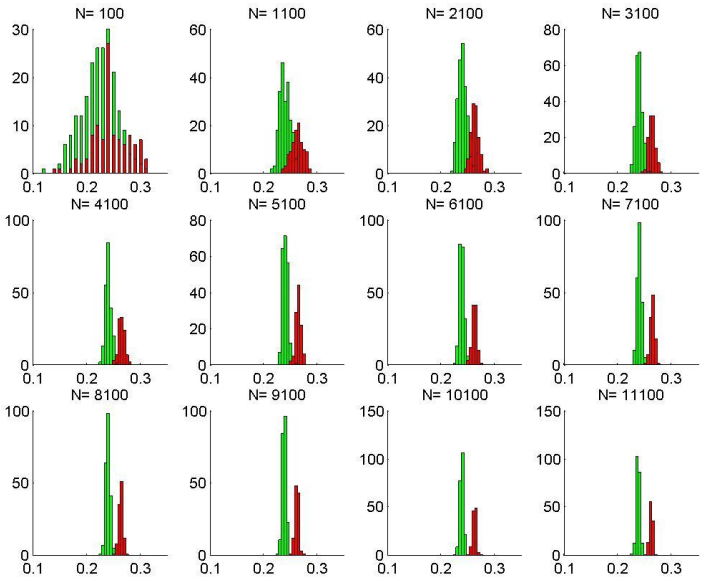
Cross validation

Bootstrapping

Variable importance

Sampling methods





Willow Garage

http://opencv.willowgarage.com

15

OpenCV Contracting Group

itseez

- **Software development and contract consulting for OpenCV**

Willow Garage

OpenCV - What's new, What's coming

New in OpenCV

- Full support for Android
- Ever growing GPU port
- Direct Kinect support
- Full C++ and STL compatible interface
- Full python interface
- Features2D – detectors/descriptors
- New, more accurate calibration patterns
- Fast Approximate Nearest Neighbor learning

What's Coming

Port to iOS iPhone/iPad

A processing flow graph:

Write in python output in C++ or run as a ROS node

Higher level components

Interoperability with PCL

Perception App store

Willow Garage

New C++ and Python APIs:

Focus Detector

C:

```
double calcGradients(const IplImage *src, int aperture_size = 7)
{
    CvSize sz = cvGetSize(src);
      IplImage* img16_x = cvCreateImage( sz, IPL_DEPTH_16S, 1);
      IplImage* img16_y = cvCreateImage( sz, IPL_DEPTH_16S, 1);

      cvSobel( src, img16_x, 1, 0, aperture_size);
      cvSobel( src, img16_y, 0, 1, aperture_size);

    IplImage* imgF_x = cvCreateImage( sz, IPL_DEPTH_32F, 1);
    IplImage* imgF_y = cvCreateImage( sz, IPL_DEPTH_32F, 1);

    cvScale(img16_x, imgF_x);
    cvScale(img16_y, imgF_y);

    IplImage* magnitude = cvCreateImage( sz, IPL_DEPTH_32F, 1);
    cvCartToPolar(imgF_x, imgF_y, magnitude);
    double res = cvSum(magnitude).val[0];

    cvReleaseImage( &magnitude );
    cvReleaseImage(&imgF_x);
      cvReleaseImage(&imgF_y);
      cvReleaseImage(&img16_x);
      cvReleaseImage(&img16_y);

    return res;
}
```

C++:

```
double contrast_measure(const Mat& img)
{
  Mat dx, dy;

  Sobel(img, dx, 1, 0, 3, CV_32F);
  Sobel(img, dy, 0, 1, 3, CV_32F);
  magnitude(dx, dy, dx);

  return sum(dx)[0];
}
```

Python API: Optical Flow Features

```
import cv
>>> img = cv.LoadImageM("building.jpg", cv.CV_LOAD_IMAGE_GRAYSCALE)
>>> eig_image = cv.CreateMat(img.rows, img.cols, cv.CV_32FC1)
>>> temp_image = cv.CreateMat(img.rows, img.cols, cv.CV_32FC1)
>>> for (x,y) in cv.GoodFeaturesToTrack(img, eig_image, temp_image, 10, 0.04, 1.0, useHarris = True):
...     print "good feature at", x,y
```

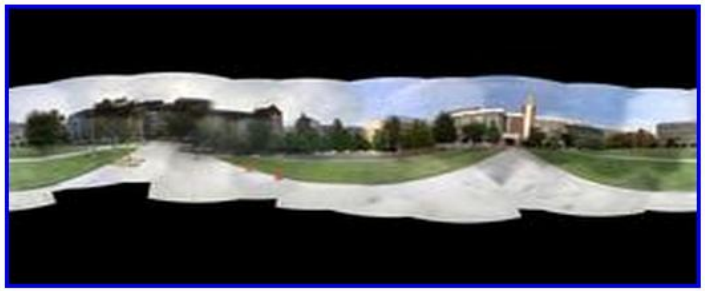
Android Port

Example: Panorama using an a-Phone. These have the same data structure as used in Streetview.

See “The Vegan Robot” http://theveganrobot.com/

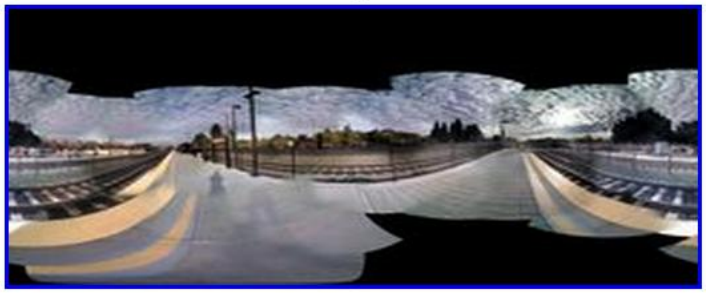
Palo Alto Traffic Circle

Georgia Tech Quad



Georgia Tech Office

MountainView Rail Stop



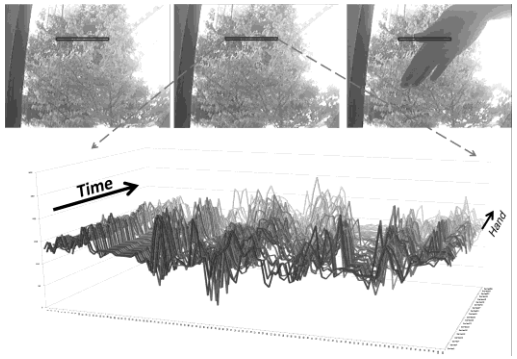
Office Complex 1

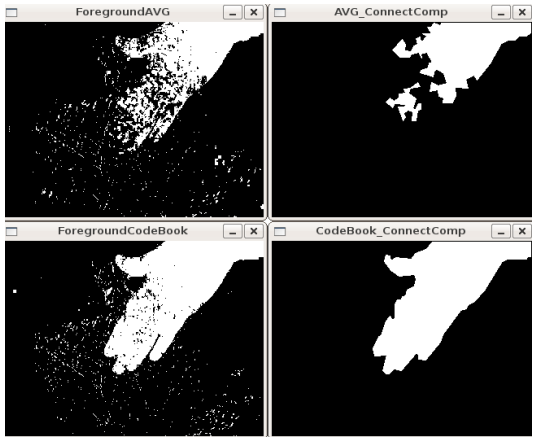
TheVeganKitchen

Segmentation

BackgroundSubtractorMOG2(), see samples/cpp/bgfg_segm.cpp

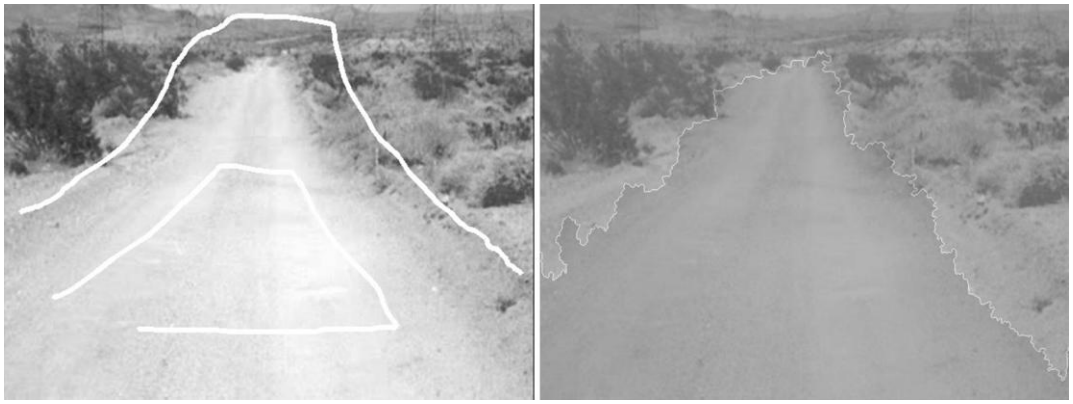
- Background subtraction,





- *pyramid, mean-shift, graph-cut*

- Watershed

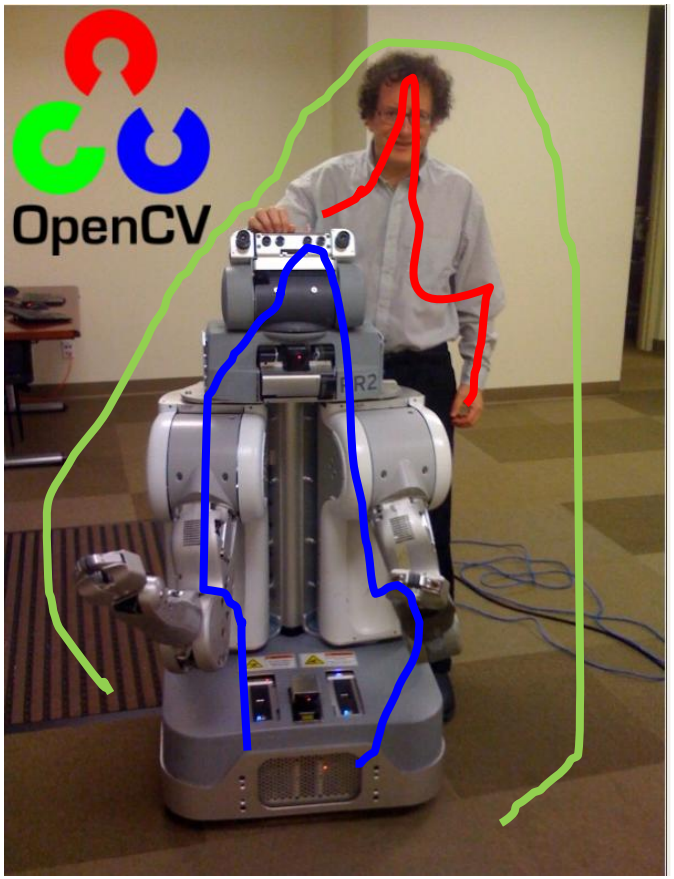


void watershed(const Mat& image, Mat& markers)

GrabCut

void grabCut(const Mat& image, Mat& mask, Rect rect, Mat& bgdModel, Mat& fgdModel, int iterCount, int mode)

Graph Cut based segmentation



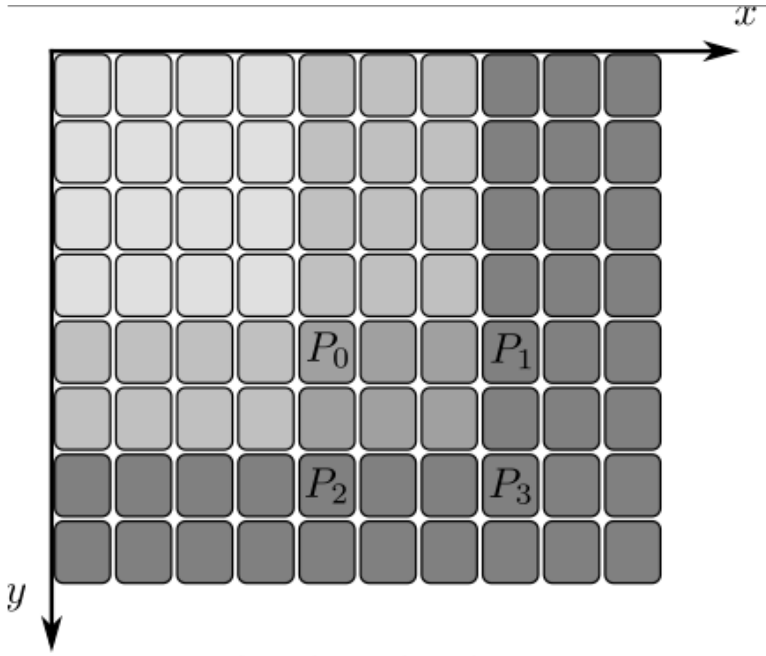
Background

Robot

Person

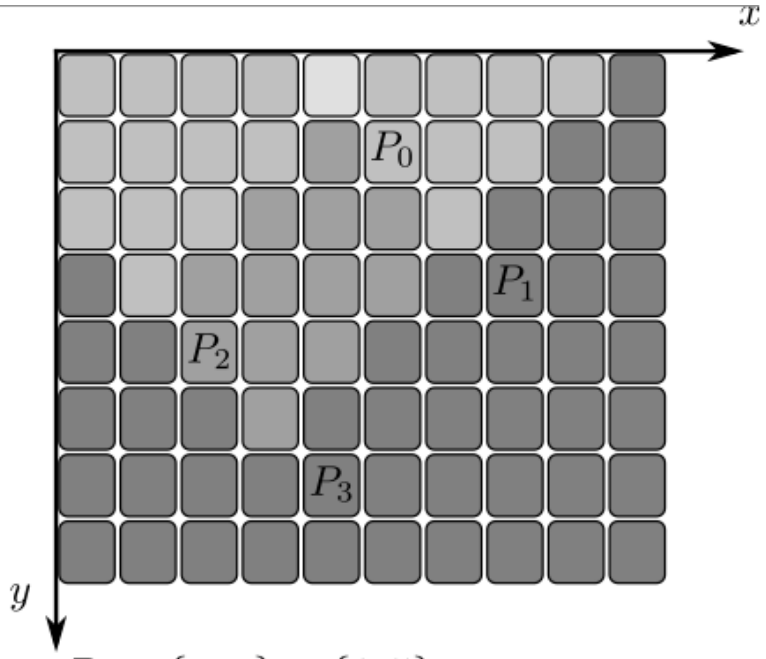
Integral images

- Fast calculation of rectangular regions



$$P_0 = \{y, x\} = \{4, 4\}$$
$$P_1 = \{y, x + w\} = \{4, 7\}$$
$$P_2 = \{y + h, x\} = \{6, 4\}$$
$$P_3 = \{y + h, x + w\} = \{6, 7\}$$

Upright rectangle

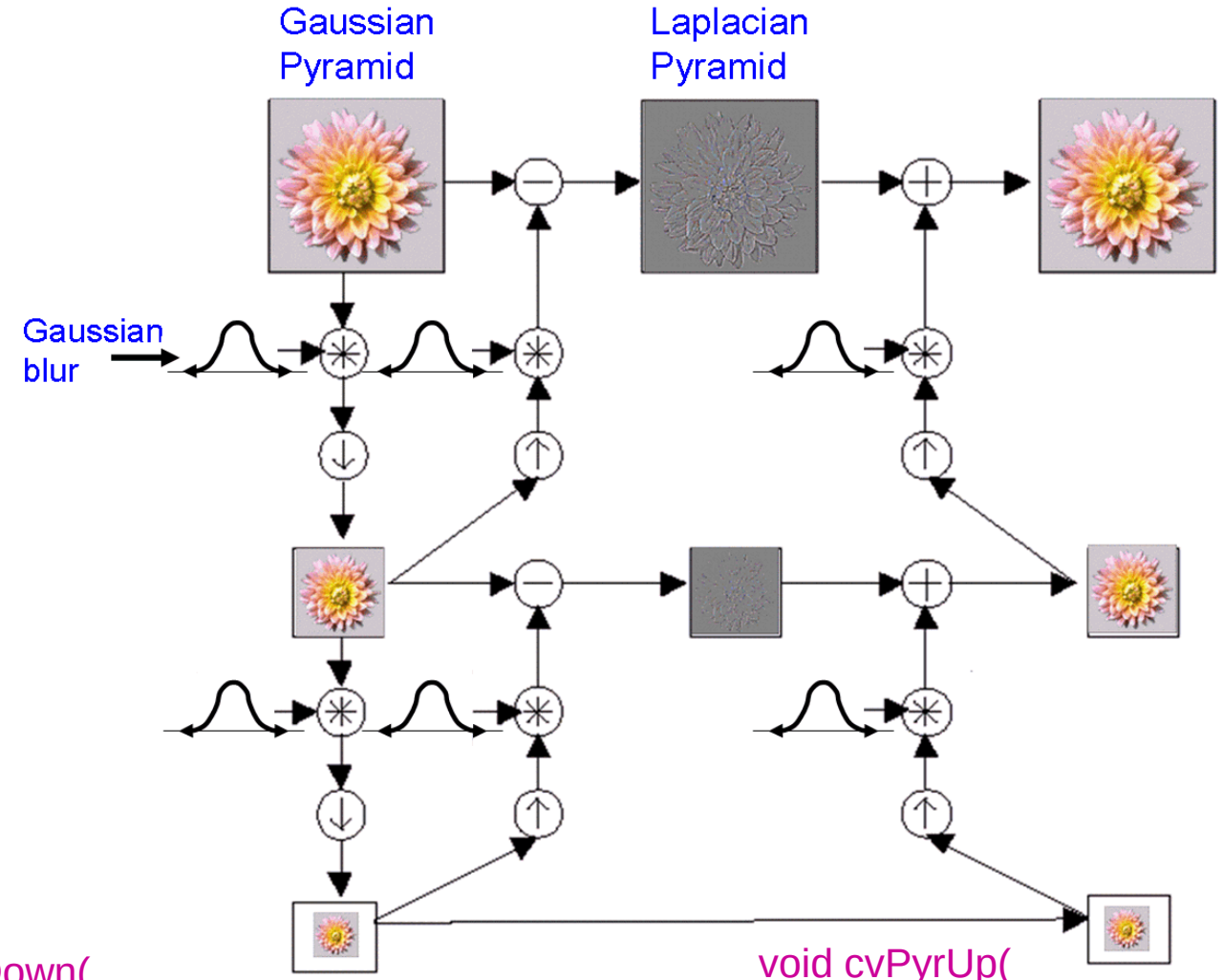


$$P_0 = \{y, x\} = \{1, 5\}$$
$$P_1 = \{y + w, x + w\} = \{3, 7\}$$
$$P_2 = \{y + h, x - h\} = \{4, 2\}$$
$$P_3 = \{y + w + h, x + w - h\} = \{6, 4\}$$

rotated rectangle

void integral()

Scale Space




```
void cvPyrDown(
    IplImage*          src,
    IplImage*          dst,
    IplFilter filter = IPL_GAUSSIAN_5x5);
```

```
void cvPyrUp(
    IplImage*          src,
    IplImage*          dst,
    IplFilter filter = IPL_GAUSSIAN_5x5);
```

Features 2D

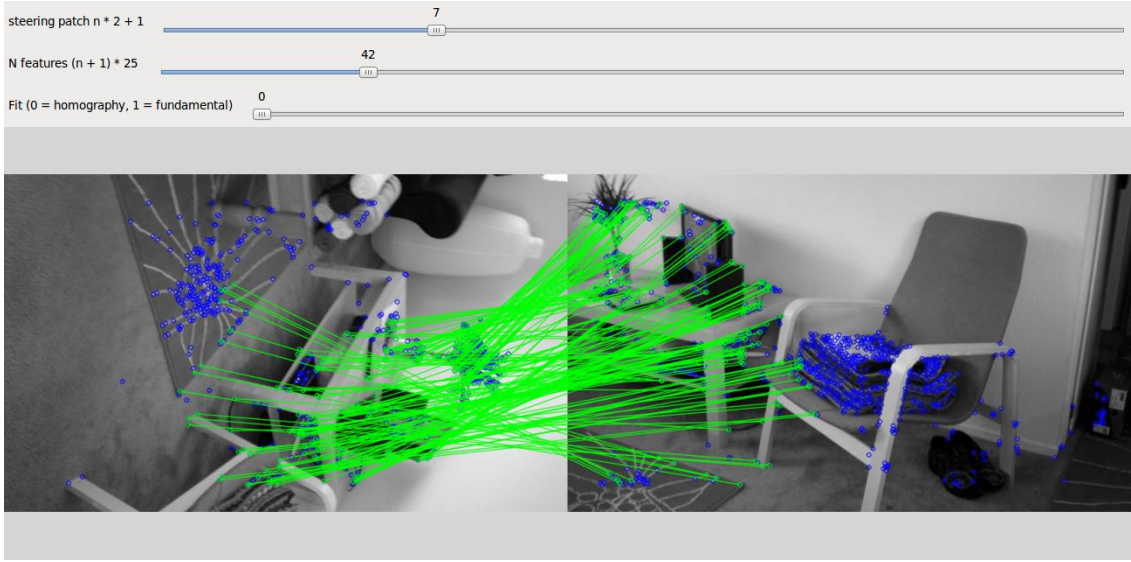
Encompasses SIFT, SURF, etc

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);
```

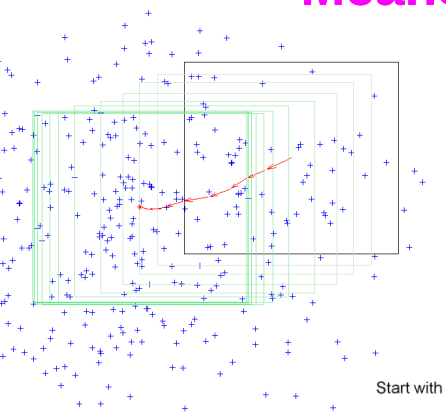
Now, find the closest matches between descriptors from the first image to the second:

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

Tracking

2D

CamShift();
MeanShift();

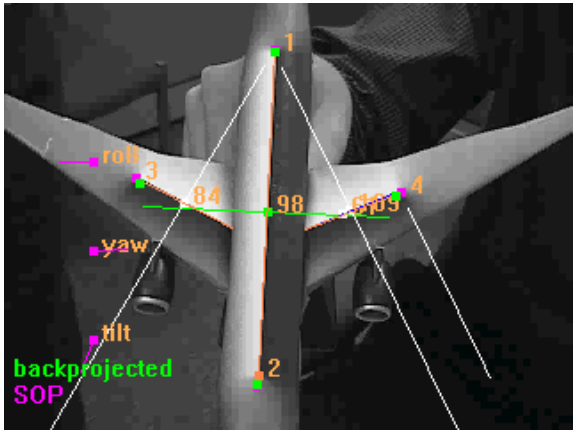


KalmanFilter::

calcOpticalFlowPyrLK()
Also see dense optical flow:
calcOpticalFlowFarneback()

3D

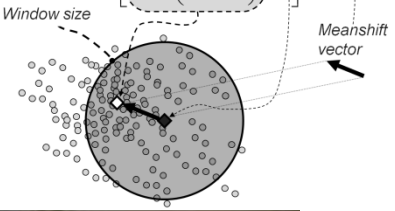
Posit();
SolvePnP();

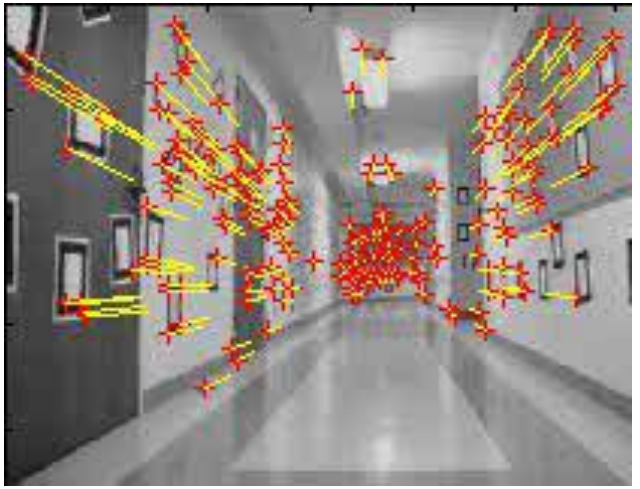


Start with a kernel $K(\mathbf{x}-\mathbf{x}_i) = ck\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)$ approximation of a probability distribution $P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x}-\mathbf{x}_i)$. Focus on the gradient $\nabla P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla K(\mathbf{x}-\mathbf{x}_i)$.

Let: $\mathbf{g}(\mathbf{x}) = -k'(\mathbf{x})$, the derivative of the kernel and we get:

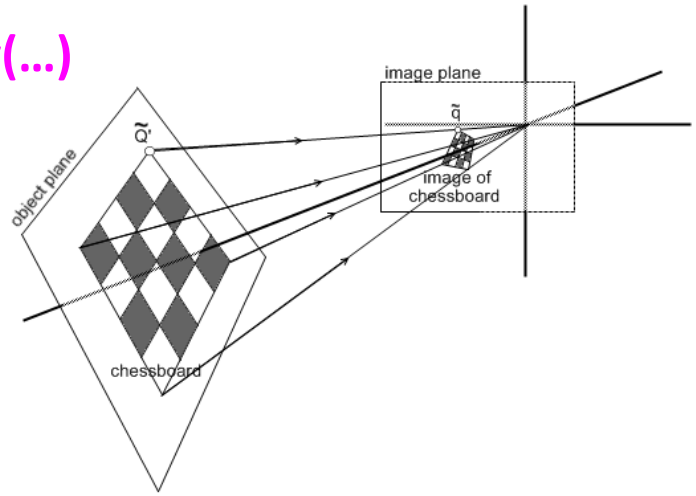
$$\nabla P(x) = \frac{c}{n}\sum_{i=1}^{n}\nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\left(\frac{\|x-x_i\|^2}{h}\right)\right]\left[\frac{\sum_{i=1}^{n} x_i g_i\left(\frac{\|x-x_i\|^2}{h}\right)}{\sum_{i=1}^{n} g_i\left(\frac{\|x-x_i\|^2}{h}\right)} - x\right]$$





Homography & Camera Calibration

findHomography(…)



Gary Bradski and Adrian Kaehler: Learning OpenCV

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

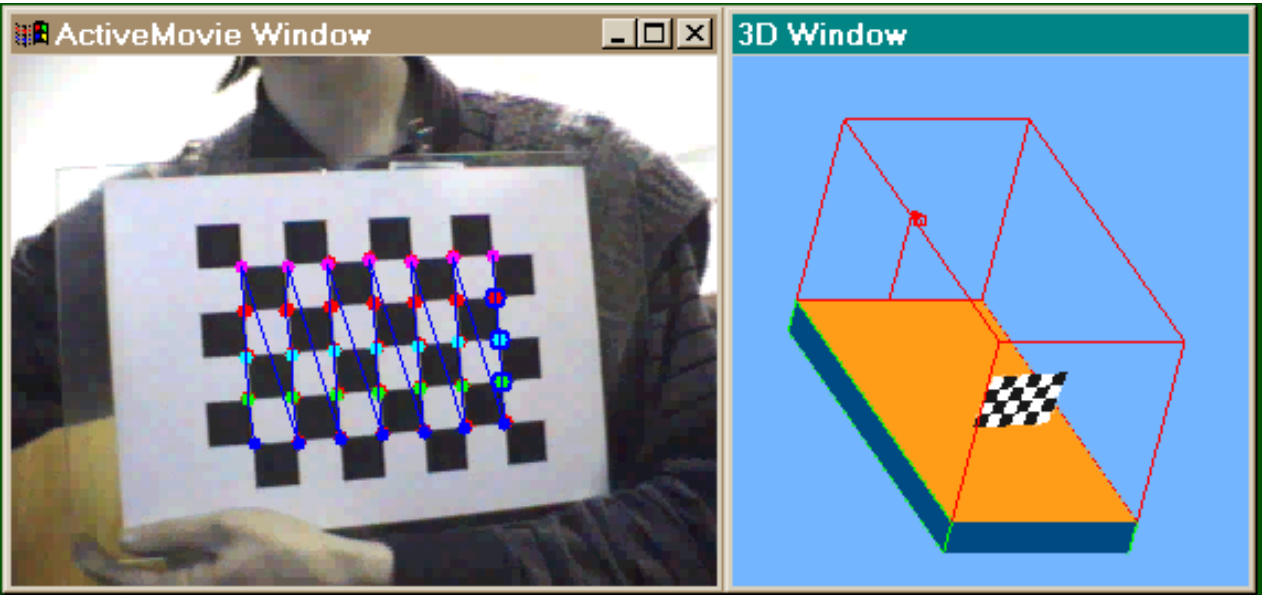
$$x = f\frac{X}{Z}$$

$$y = f\frac{X}{Z}$$

$$p = M_{int} P_C$$

See samples/cpp/calibration.cpp

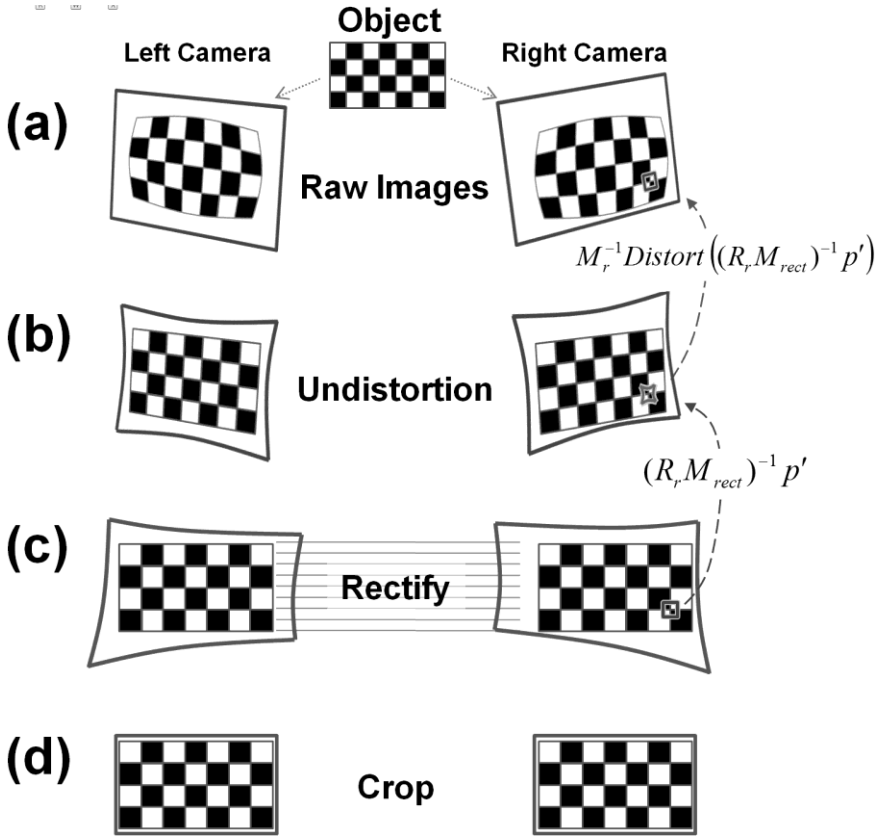
3D view of checkerboard

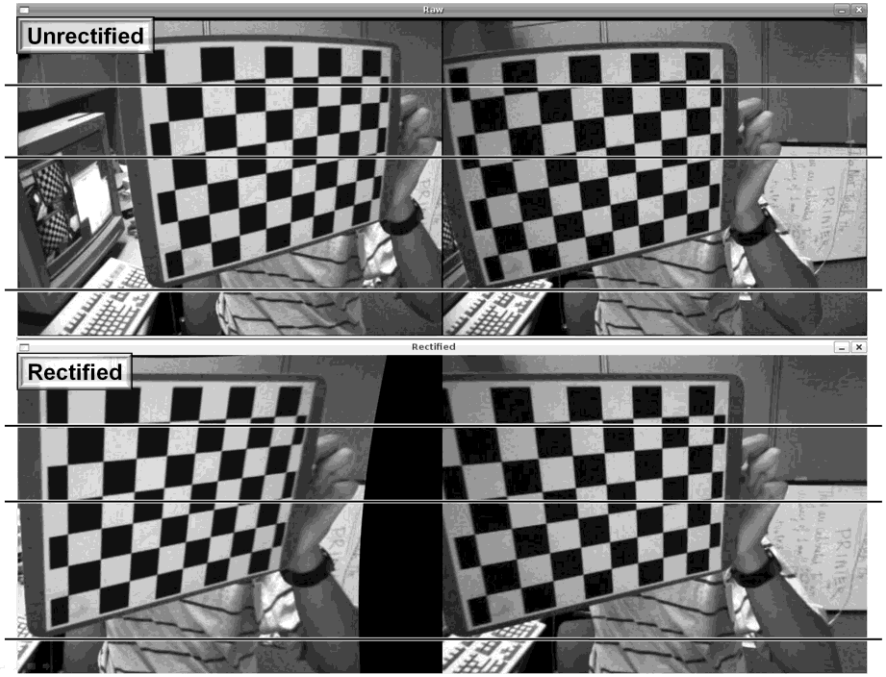


Un-distorted image

Stereo

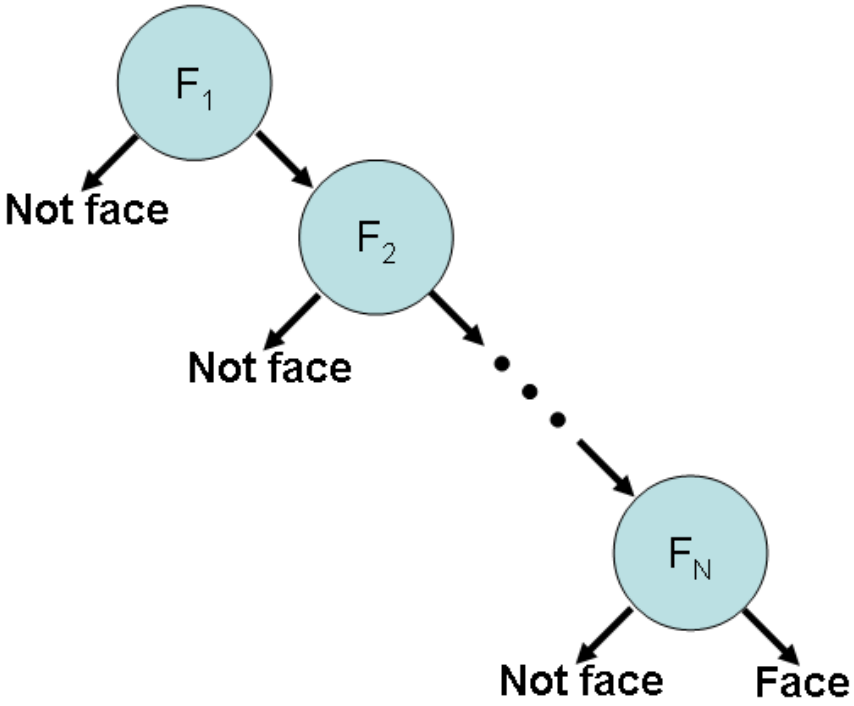
- Once the left an right cameras are calibrated internally (intrinsics) and externally (extrinsics), we need to rectify the images





ML Lib Example:

Boosting: Face Detection with Viola-Jones Rejection Cascade



In samples/cpp, see:
Multicascadeclassifier.cpp

Useful OpenCV Links

OpenCV Wiki:
http://opencv.willowgarage.com/wiki

User Group (44700 members 4/2011):
http://tech.groups.yahoo.com/group/OpenCV/join

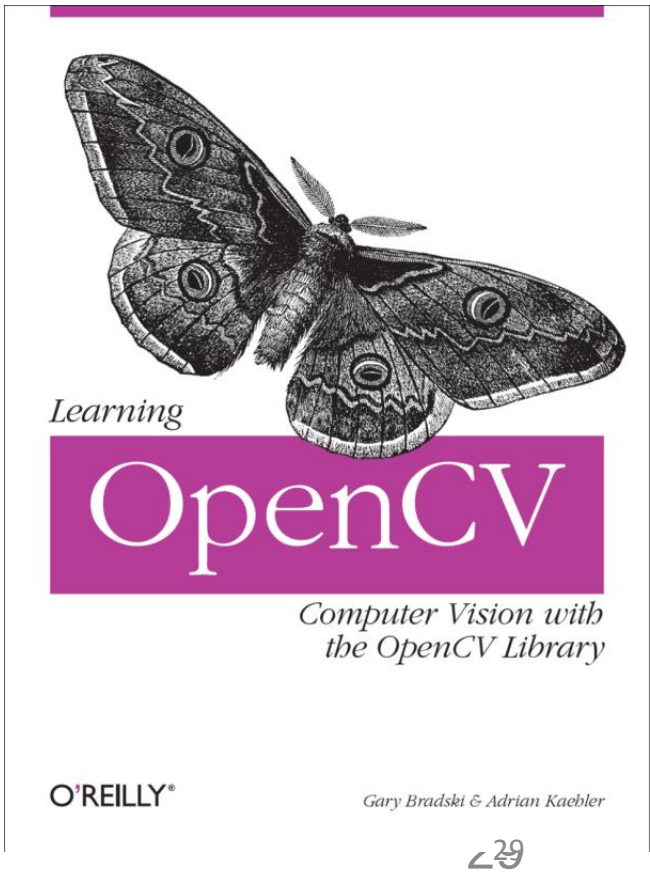
OpenCV Code Repository:
svn co https://code.ros.org/svn/opencv/trunk/opencv

New 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/

Documentation
http://opencv.willowgarage.com/documentation/index.html



Willow Garage

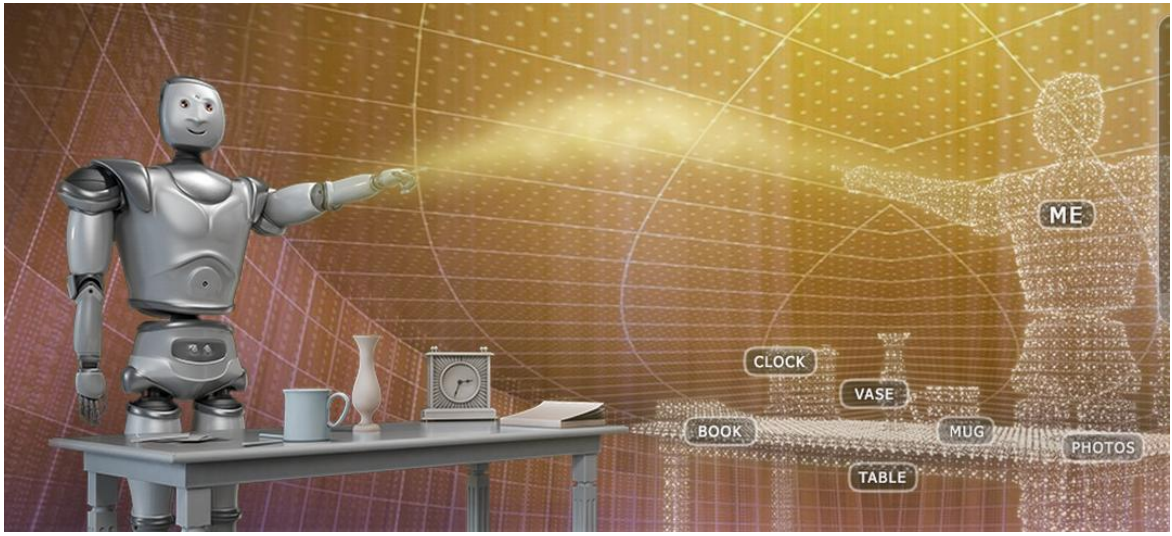
Outline

- *What's Willow Garage*
- *Perception is Hard*
- *Open Source Computer Vision Library (OpenCV)*
- Point Cloud Library (PCL)
- Current Research Results
- (if time) Speculations on Perception

Willow Garage

3D Processing: PCL

- Point Cloud Library
  - http://pointclouds.org/



Misc, stats:

- 35 releases already (0.1.x → 0.9.9)
- over 100 classes
- over 80k lines of code (PCL, ROS interface, Visualization)
- young library: only 12 months of development so far, but we had code lying around for 3-5 years
- external dependencies on eigen, cminpack, FLANN

Summary

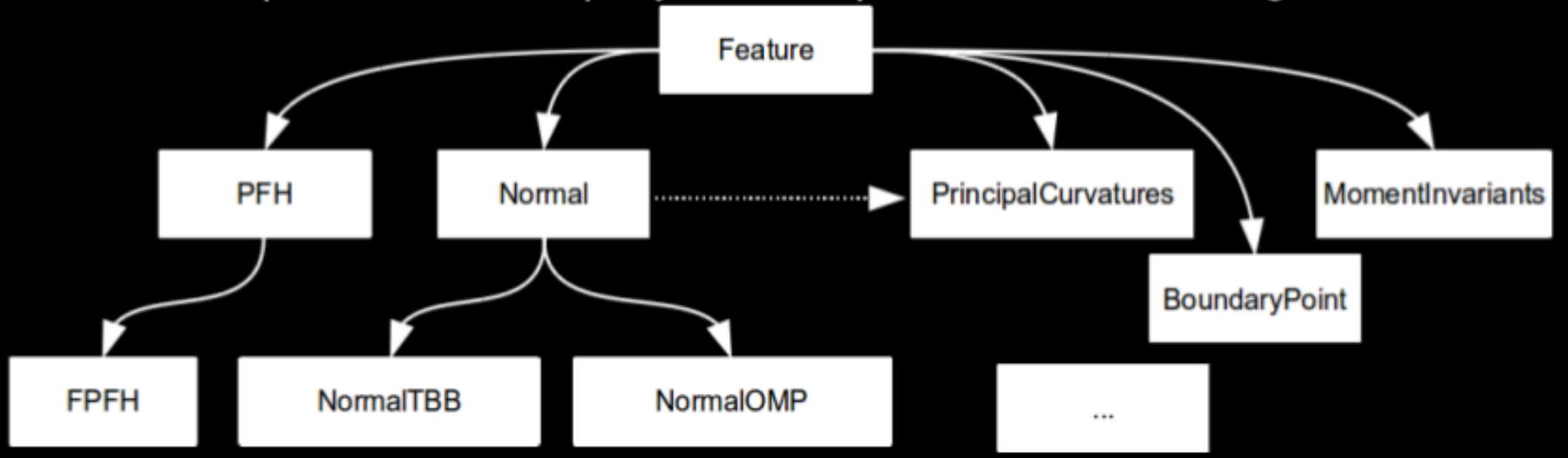
PCL (Point Cloud Library) structure

PCL

- uses SSE optimizations for fast computations
- uses OpenMP and Intel TBB for parallelization
- data passing between modules using shared pointers
- ... GPU (...)
- is split into a collection of smaller, modular C++ libraries:
  - libpcl_keypoints: nD interest points
  - libpcl_features: nD feature descriptors
  - libpcl_surface: surface meshing/reconstruction techniques
  - libpcl_filters: point cloud data filters and smoothing
  - libpcl_io: I/O operations, 3D camera drivers (e.g., Kinect)
  - libpcl_kdtree: fast nearest neighbor operations
  - libpcl_segmentation: model segmentation operations
  - libpcl_registration: point cloud registration methods
- unit tests, examples, tutorials (!)

PCL Architecture

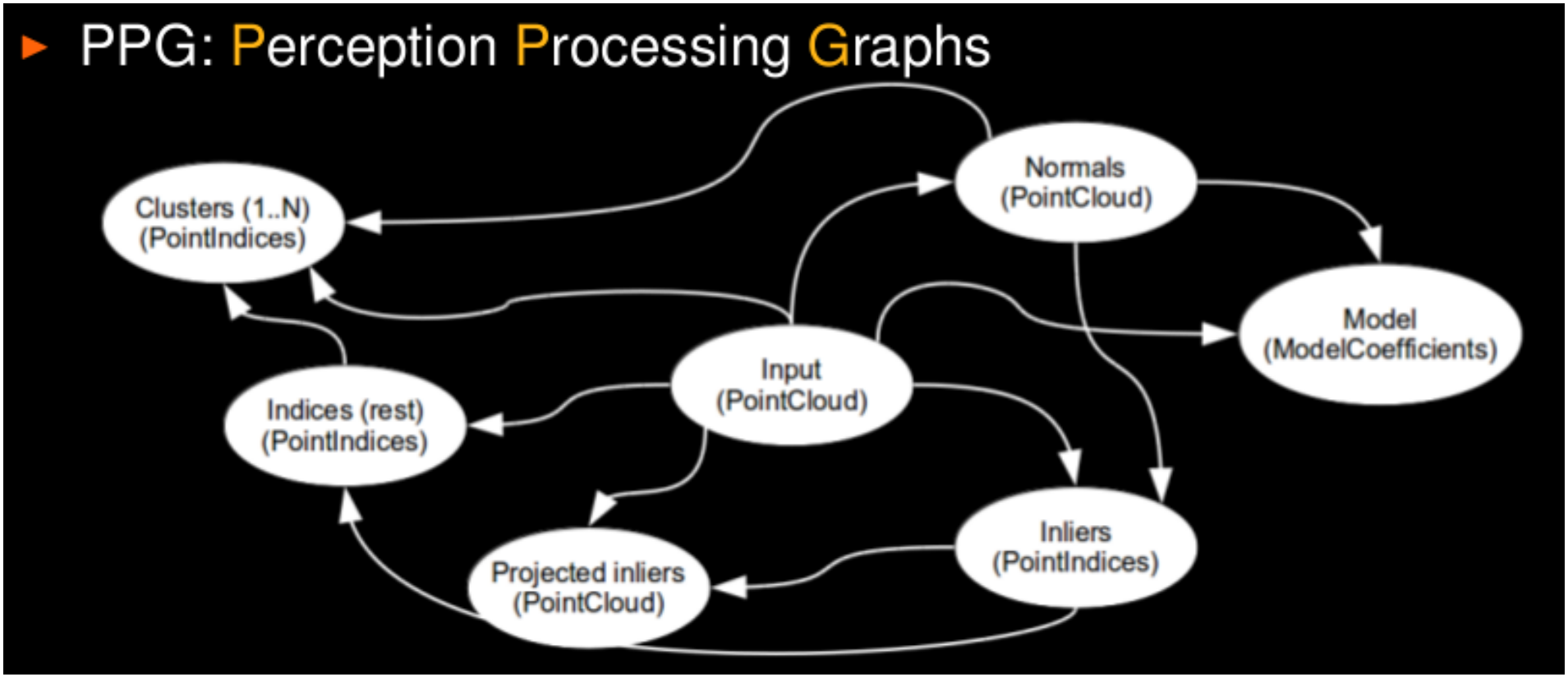
Good API practices simplify development and testing:




```
pcl::Feature<PointT> feat;
feat = pcl::Normal<PointT> (input);
feat = pcl::FPFH<PointT> (input);
feat = pcl::BoundaryPoint<PointT> (input);
...
feat.compute (&output);
...
```

Willow Garage

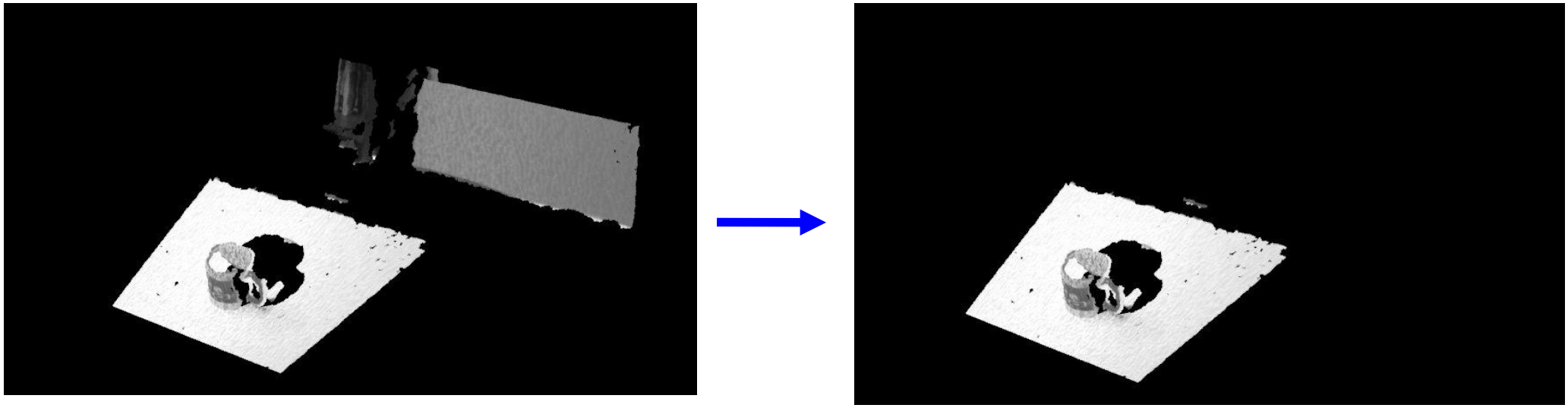
PCL: Processing Graphs



Willow Garage

PCL: Filtering by depth

```
p.setInputCloud (data);
p.FilterLimits (0.0, 0.5);
p.SetFilterFieldName ("z");
```



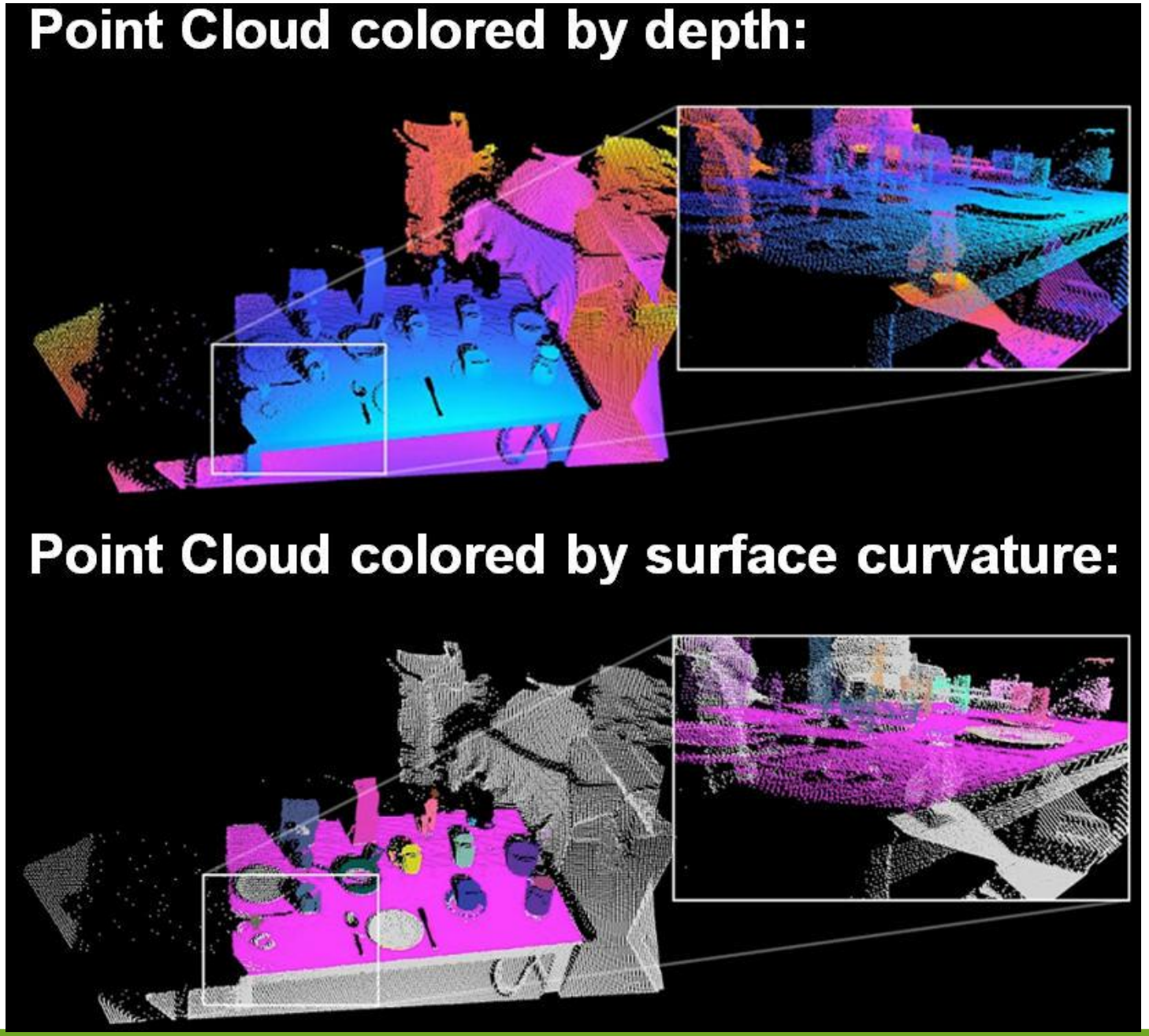
Willow Garage

PCL: Finding Normals

```
p.setInputCloud (data);
p.setInputNormals (normals);
p.SetRadiusSearch (0.01);
```

Willow Garage

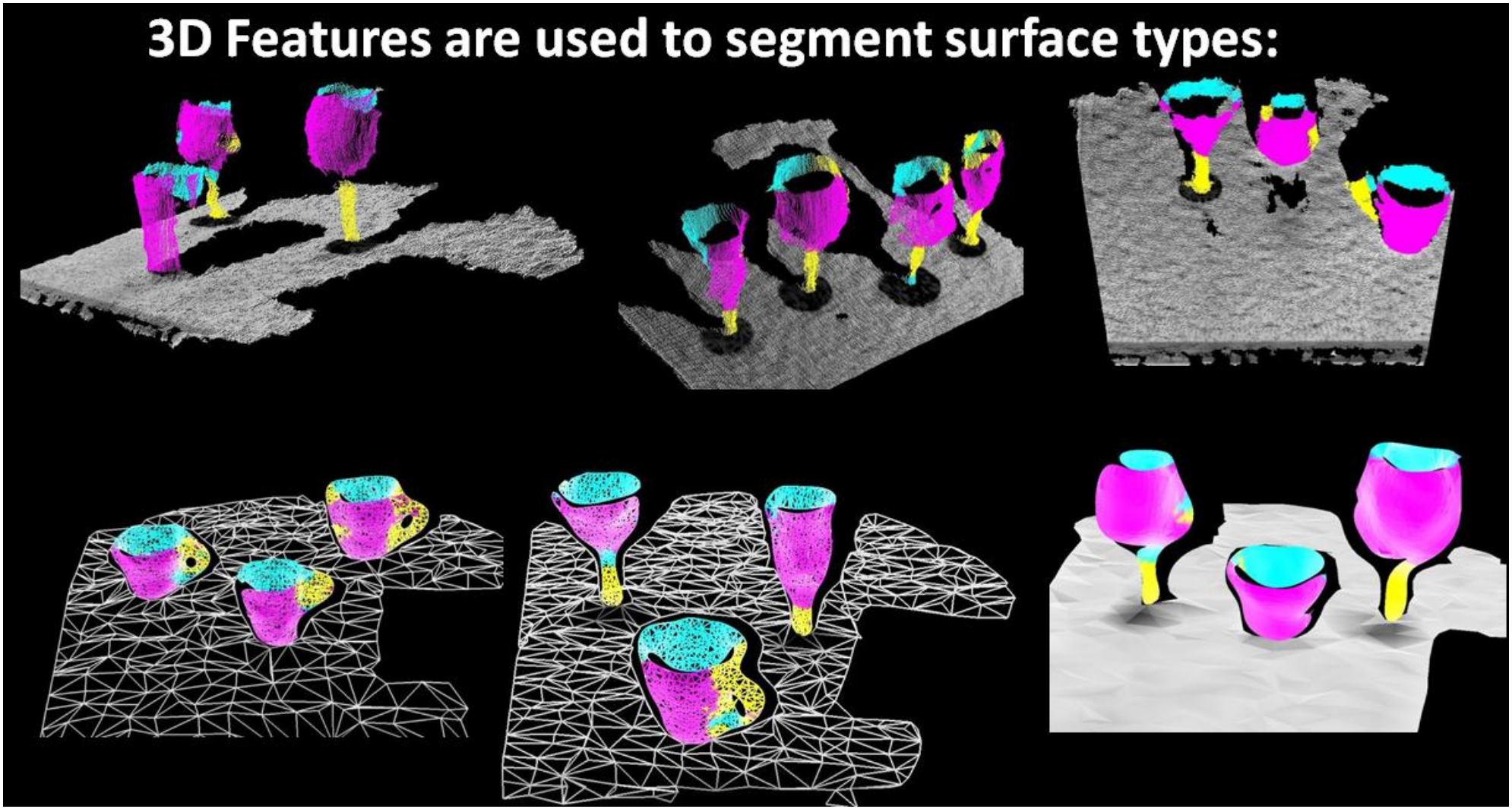
PCL: Filtering by surface curvature



Willow Garage

PCL:

Using 3D features to classify surface types



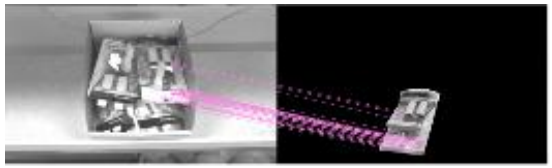
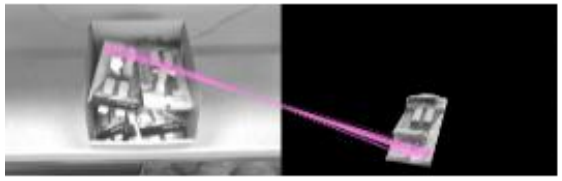
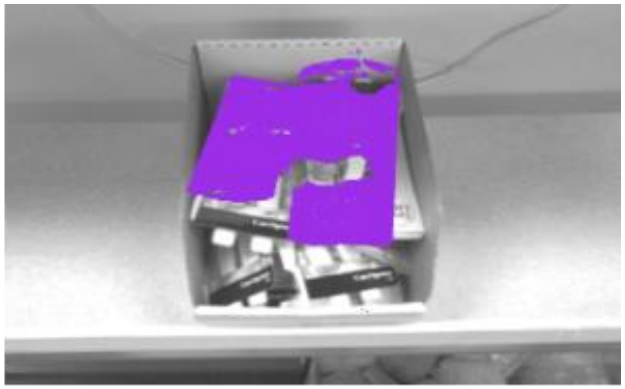
Willow Garage

Outline

- *What's Willow Garage*
- *Perception is Hard*
- *Open Source Computer Vision Library (OpenCV)*
- *Point Cloud Library (PCL)*
- Current Research Results
- (if time) Speculations on Perception

Willow Garage

OpenCV - Recent
TOD* (Textured Object Detection)

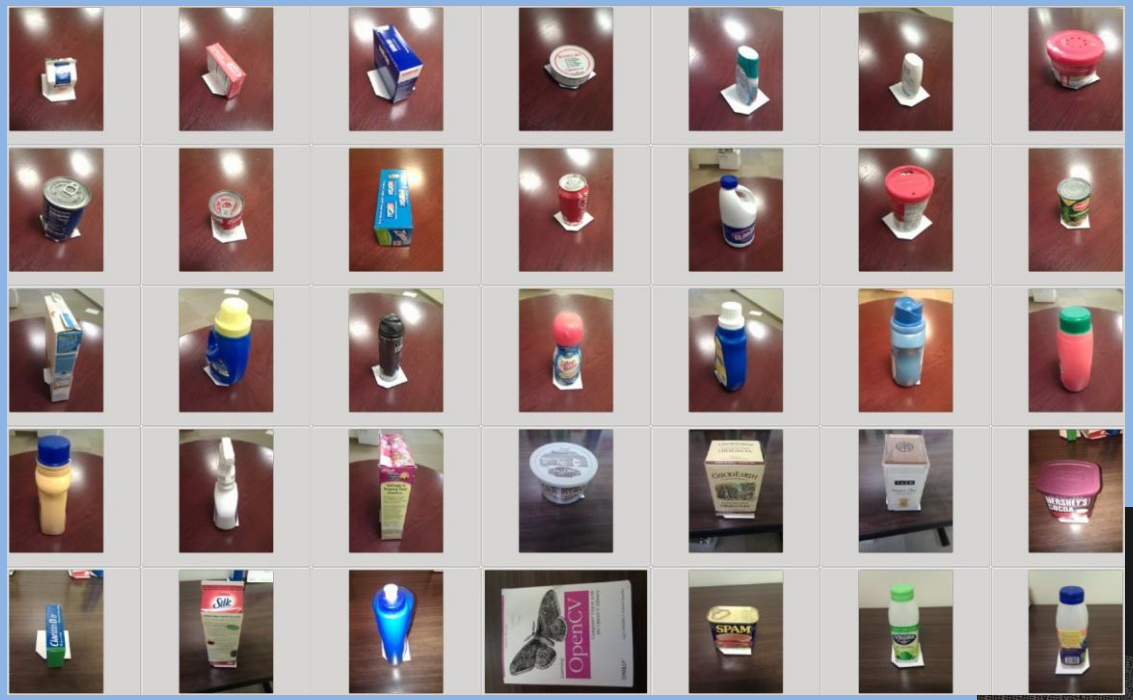


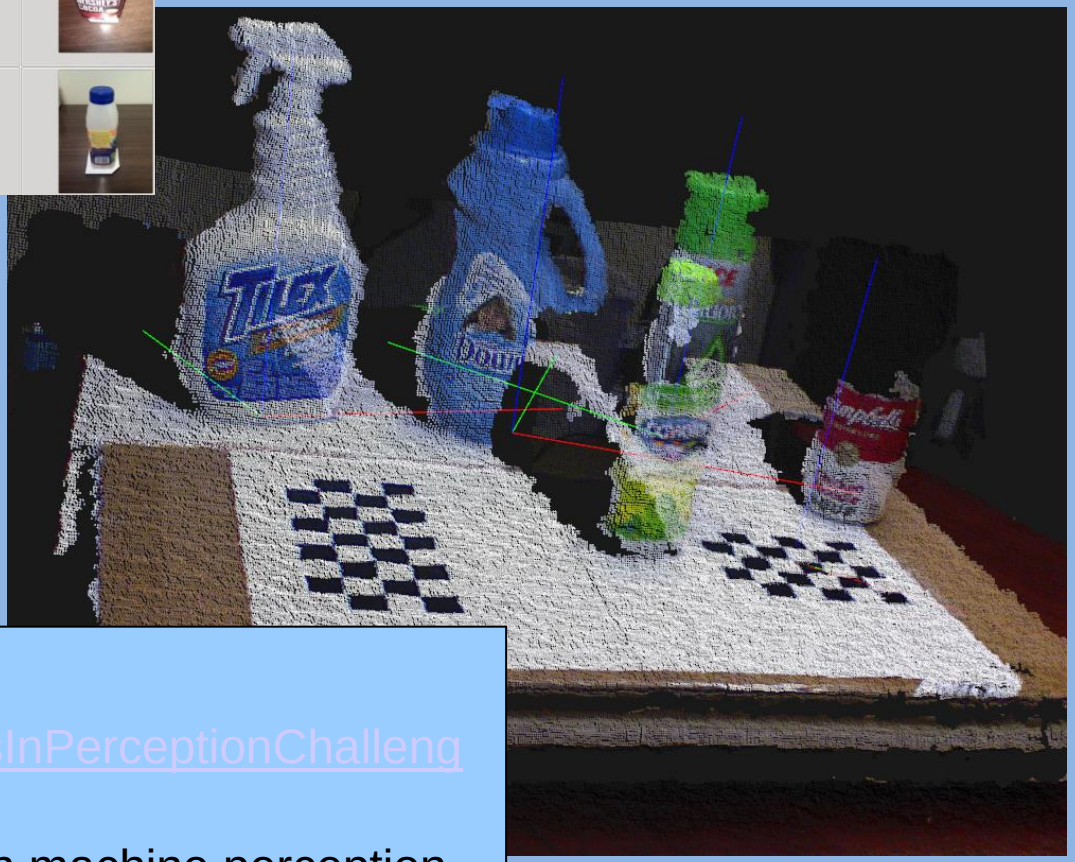
- 2D descriptors and detectors in 3D constellation using Kinect depth
- Bag of words to propose objects
- 3D to 3D fit to confirm recognition and
- Yield object pose in 6 degrees of freedom

* Similar to David Lowe's work as well as MOPED (developed by Srinivasa Siddhartha, et. al.)

Efficient Textured Object Detection (TOD), Vincent Rabaud and Ethan Rublee and Kurt Konolige and Gary Bradski. Submitted to: IROS 2011

A TOD Result

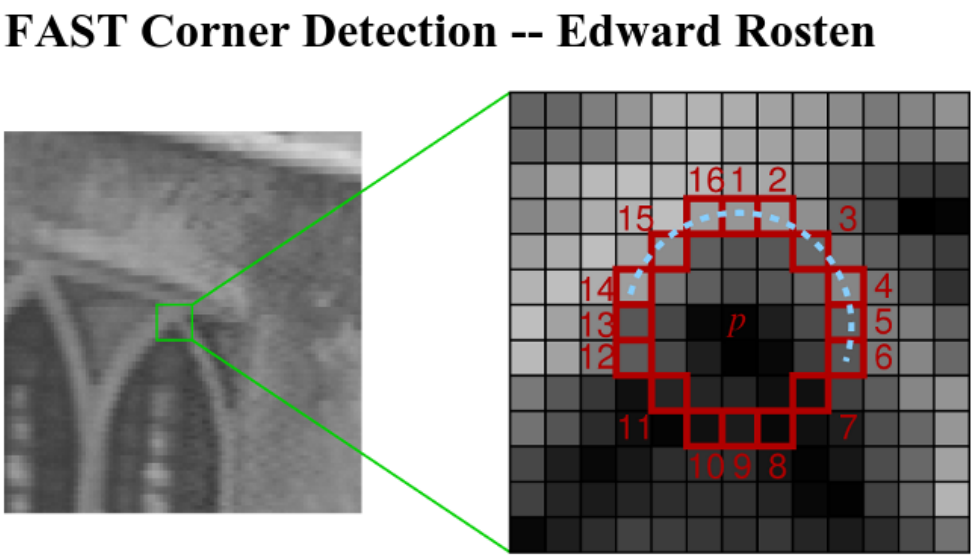




See my Solutions in Perception Challenge:
http://opencv.willowgarage.com/wiki/SolutionsInPerceptionChallenge
Effort to establish what are solved problems in machine perception
And to drive solutions in new areas

New Feature: ORB

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

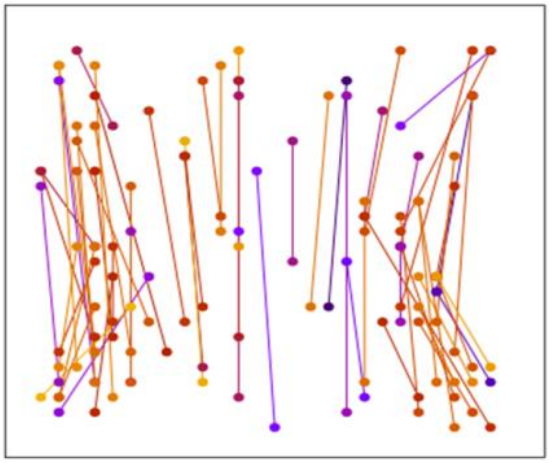


- **FAST**:
  - With reference to a central pixel "*P*" -- Interest points are detected as >= 12 contiguous pixel brighter than *P* in a ring of radius 3 around *P*.

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

Oriented FAST

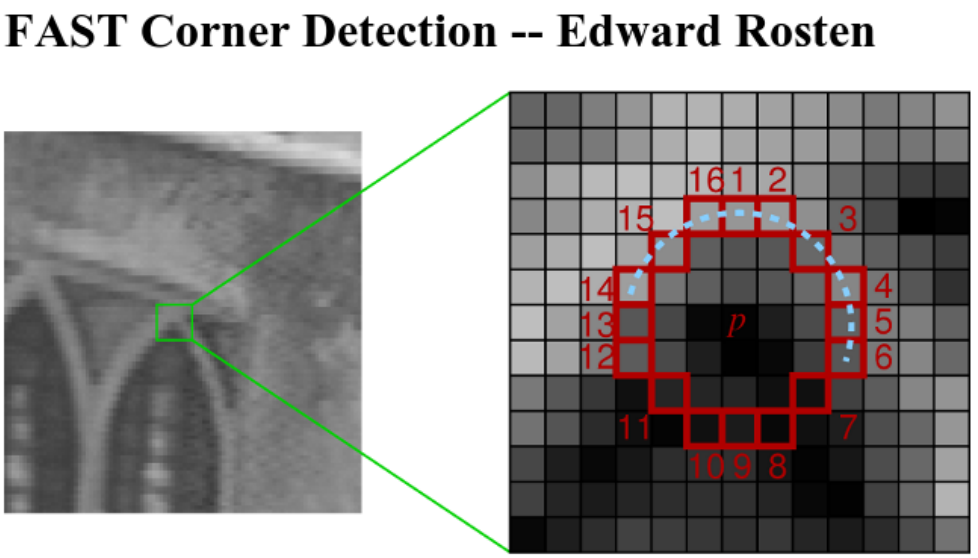
- We orient the Fast detector by taking image moments at the corner:
- Moments:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y)$$

- Corner orientation:

$$c_x = \frac{M_{10}}{M_{00}}, \quad c_y = \frac{M_{01}}{M_{00}}$$

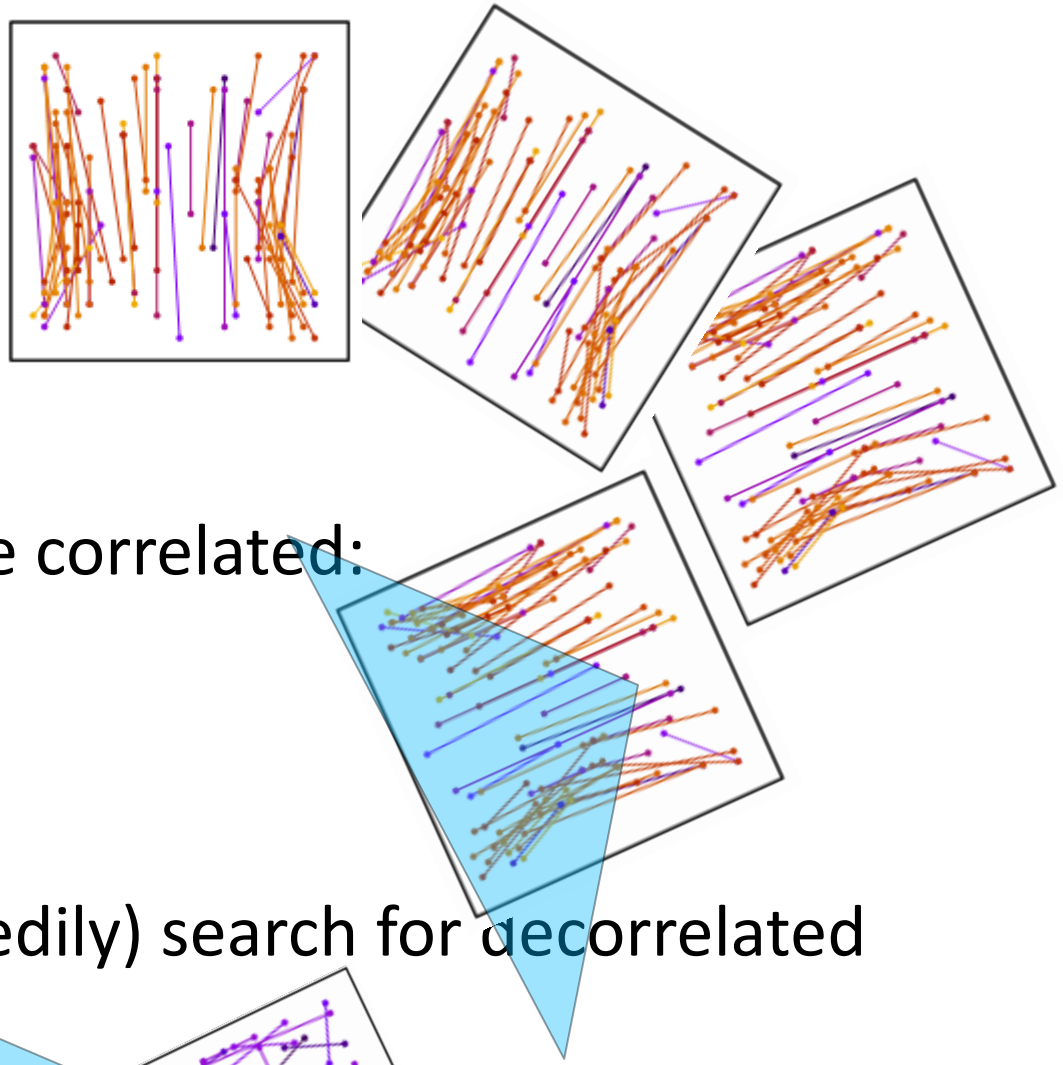
$$C_{ori} = \tan^{-1}\left(\frac{c_y}{c_x}\right)$$



Ethan Rublee and Vincent Rabaud and Kurt Konolige and Gary Bradski, ORB: an efficient alternative to SIFT or SURF, ICCV 2011 (probably)

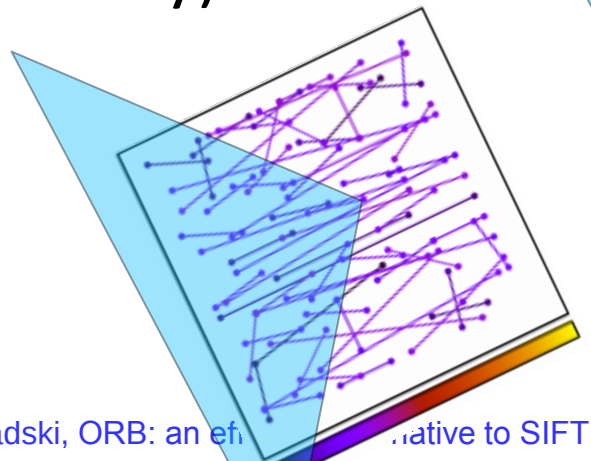
Sterable Brief

- We add sterability to BRIEF:



- **Problem**: The patches become correlated:

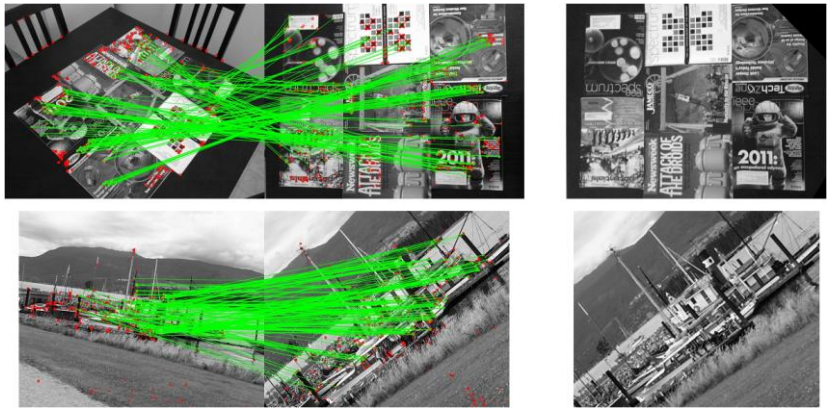
- **Solution**: We exhastively (greedily) search for decorrelated BRIEF patterns



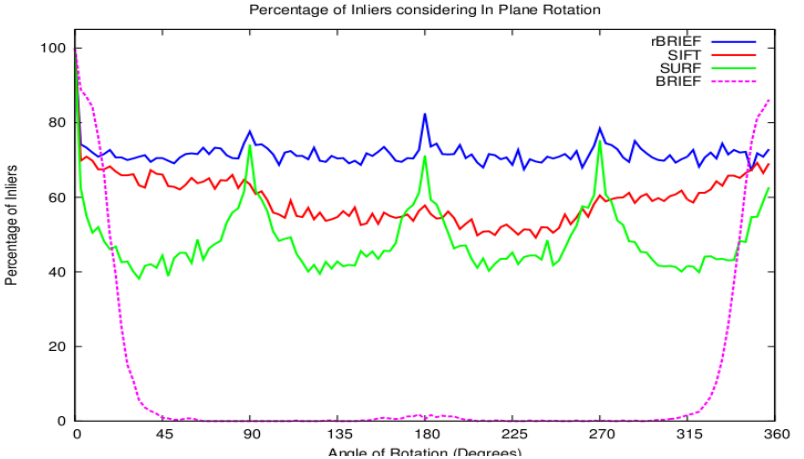
Ethan Rublee and Vincent Rabaud and Kurt Konolige and Gary Bradski, ORB: an ef… …native to SIFT or SURF, ICCV 2011 (probably)

New Feature: ORB

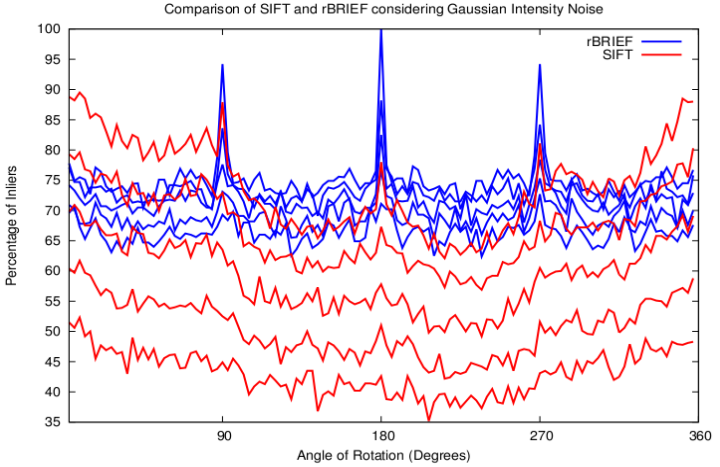
E. Rublee, V. Rabaud, K. Konolidge, G. Bradski, “ORB: an efficient alternative to SIFT and SURF”. ICCV 2011 (Submitted)



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

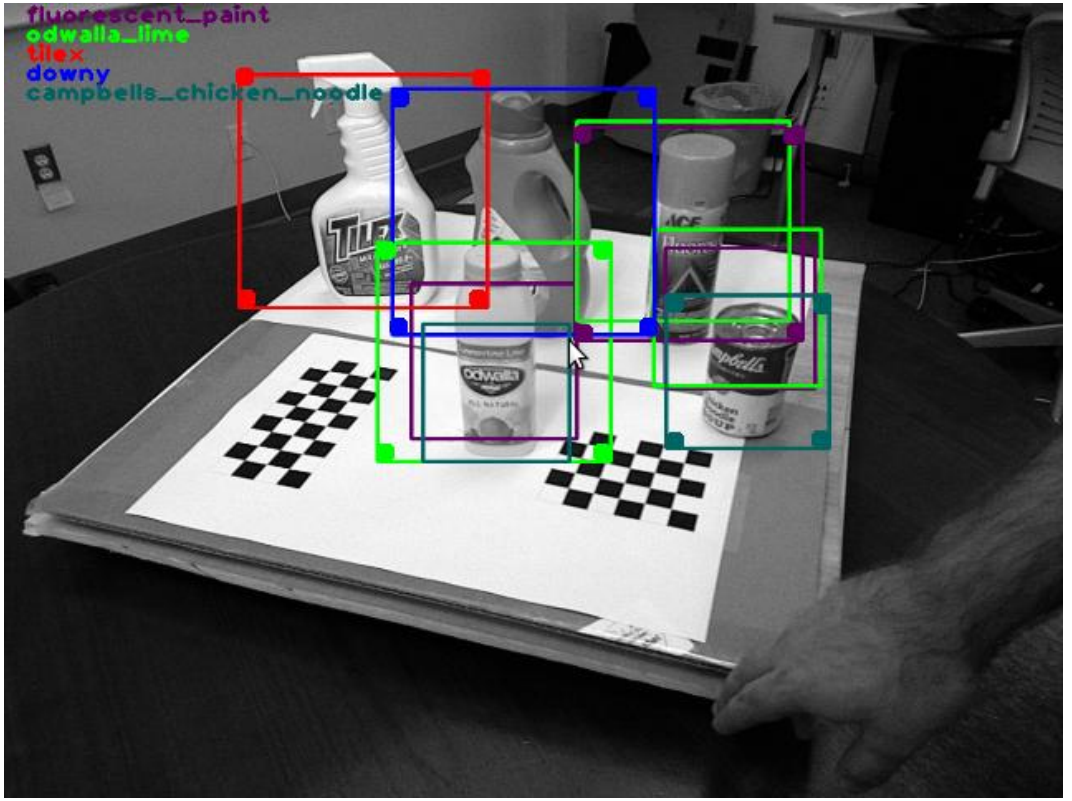


- Noise tolerance



Binarized Grid (BIG)

- Used sets of binarized features from different modalities to recognize objects (Stephen Hinterstoisser's idea).
- Here, we use a binarized grid of dominant orientations "**DOT**" for object recognition proposal.

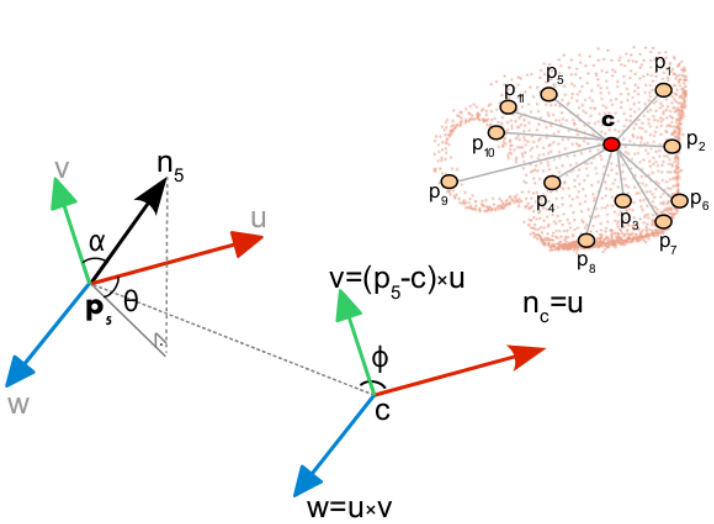


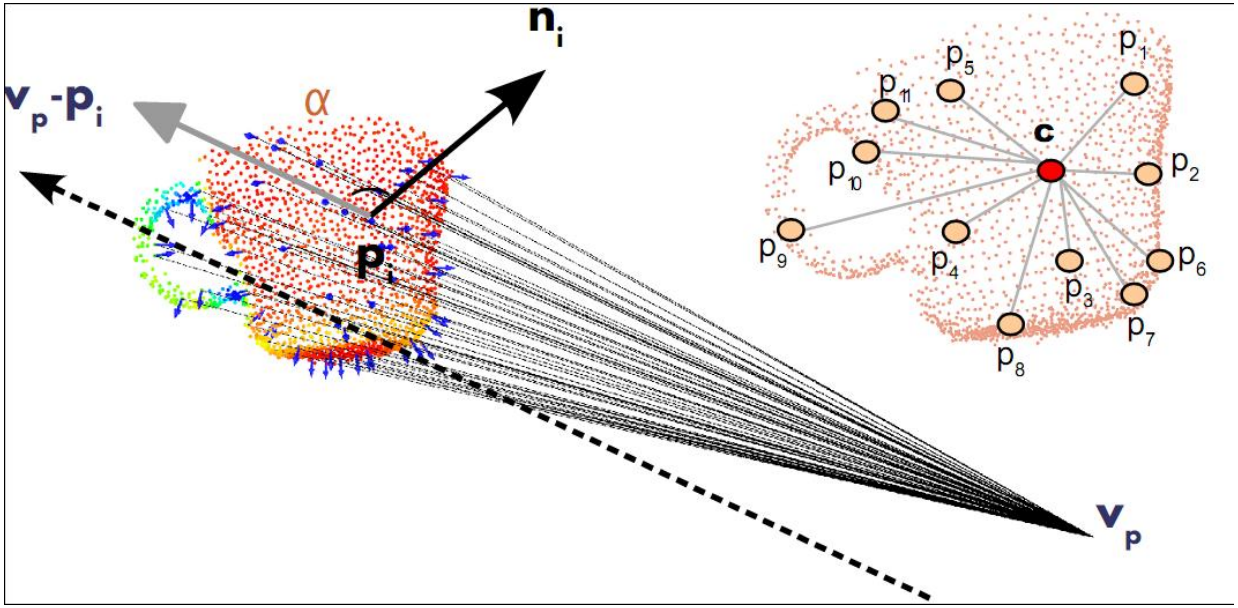
Robot Challenge: Solutions in Perception

- We (and Stanford) used ORB on the textured object dataset.
- Our (un-entered) entry is “TOD” (Textured Object Recognition)
  - Will be running in demo on the show floor all week
- Robot Challenge: Solutions in Perception.
  - Tues: Competition
  - Wed: Top competitors use recognition for grasping on PR2

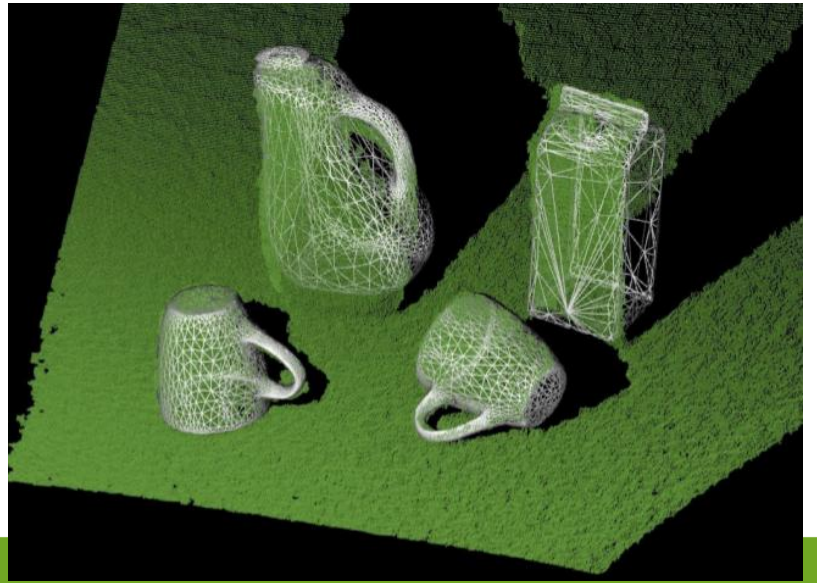
Viewpoint Feature Histogram (VFH)

- New feature: Viewpoint Feature Histogram (VFH):





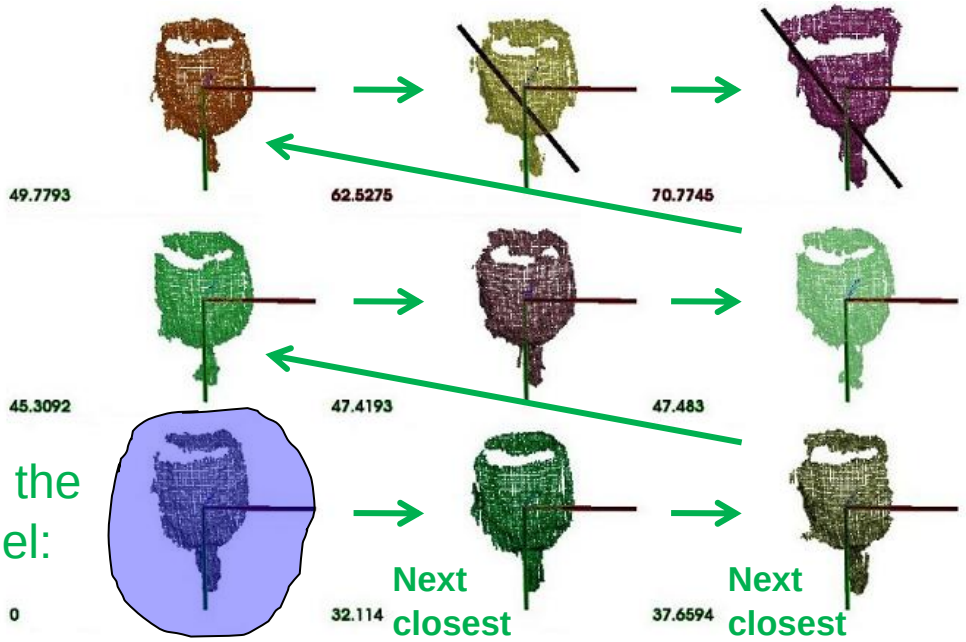
- Gives recognition and pose



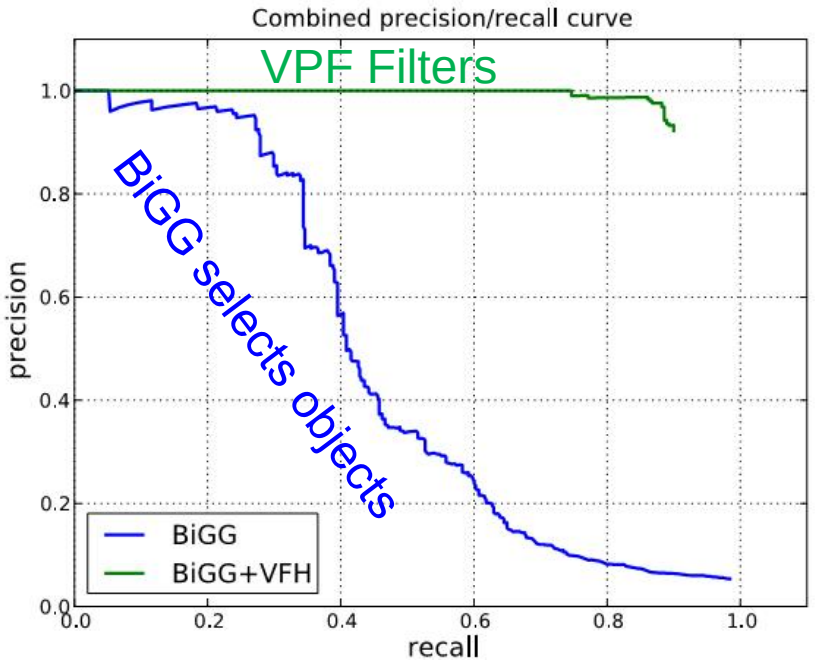
Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram, Rusu, Radu Bogdan, Bradski Gary, Thibaux Romain, and Hsu John , Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 10/2010, Taipei, Taiwan, (2010)

Willow Garage

BiGG Proposes the model
VPH Disposes



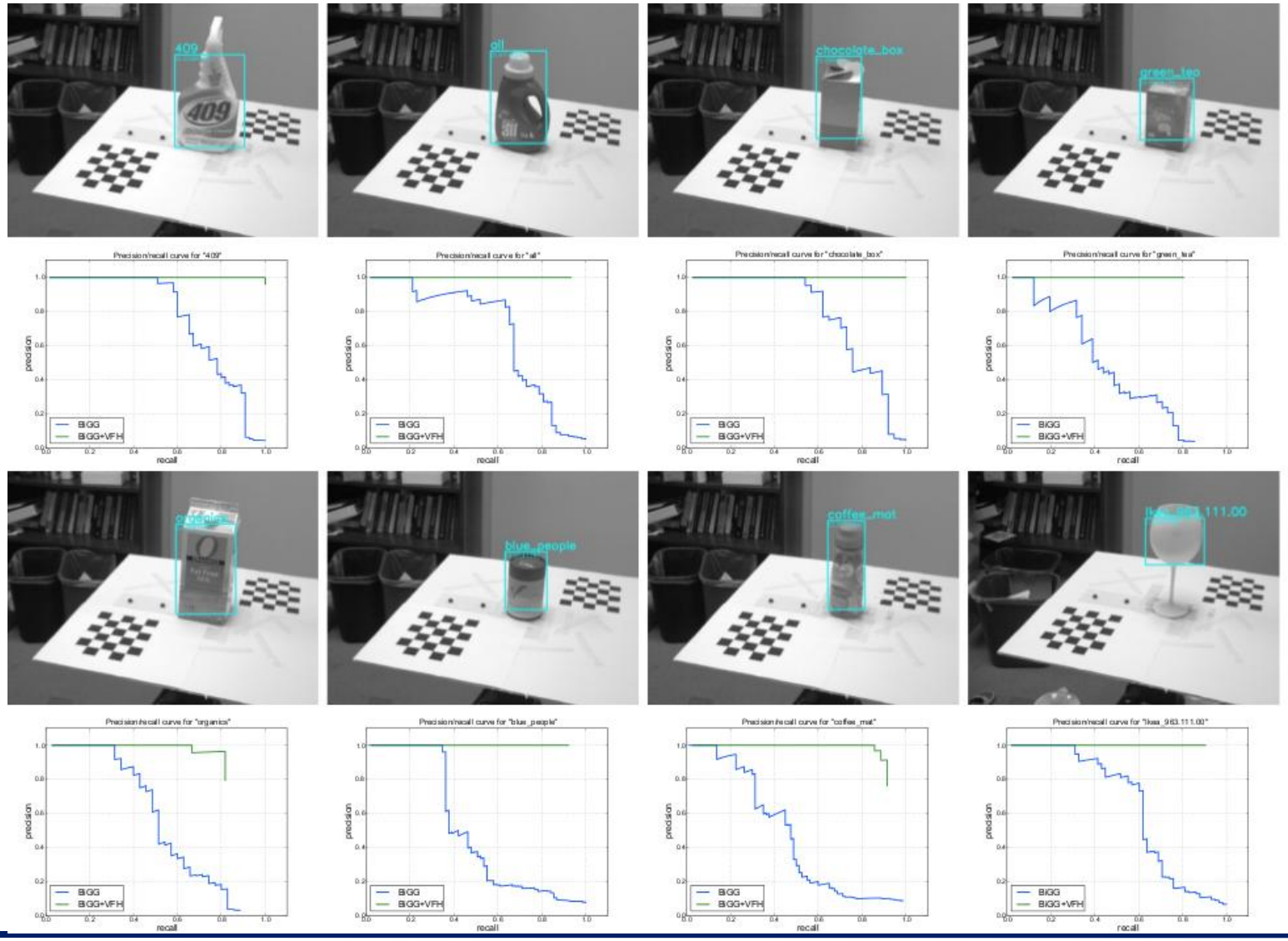
This is the model:



We get a fast, scalable and accurate classification and pose system

In Press: ICRA 2011

BIG + VFH

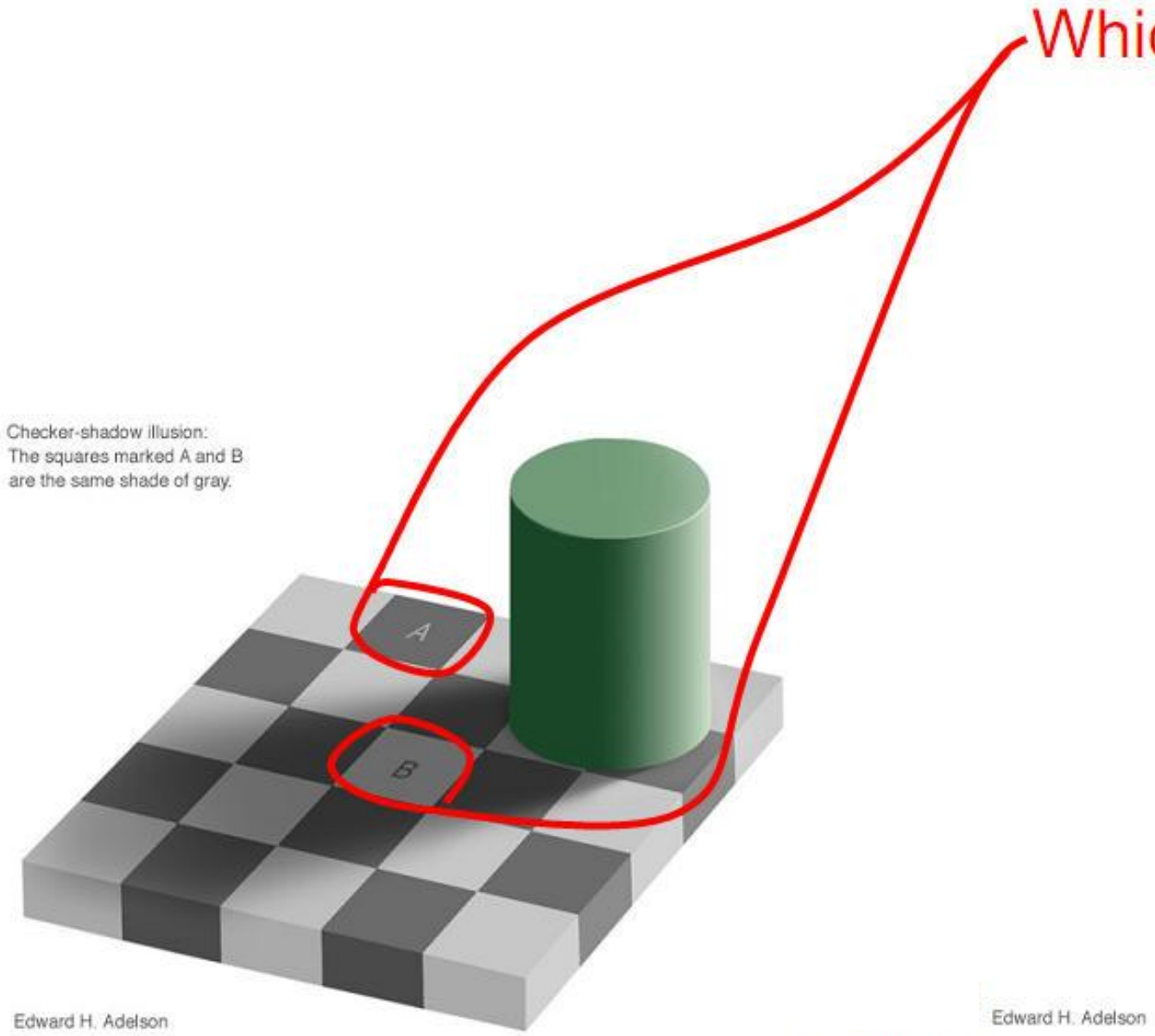

409
oil
chocolate_box
green_tea
Precision/recall curve for "409"
Precision/recall curve for "oil"
Precision/recall curve for "chocolate_box"
Precision/recall curve for "green_tea"
precision
recall
BIGG
BIGG+VFH
blue_people
coffee_mat
Precision/recall curve for "organics"
Precision/recall curve for "blue_people"
Precision/recall curve for "coffee_mat"

Outline

- *What's Willow Garage*
- *Perception is Hard*
- *Open Source Computer Vision Library (OpenCV)*
- *Point Cloud Library (PCL)*
- *Current Research Results*
- (if time) Speculations on Perception

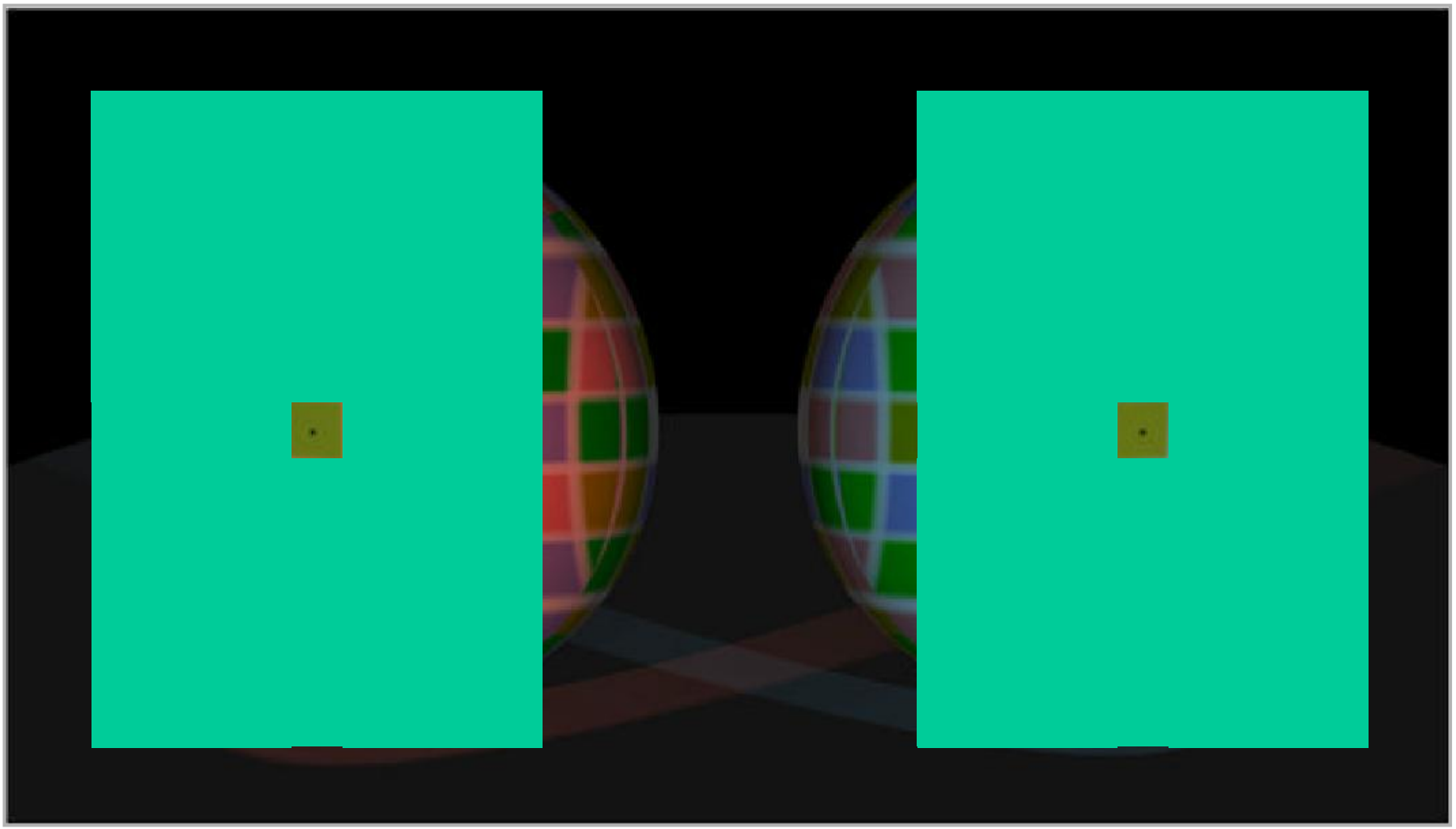
Willow Garage

Must deal with Lighting Changes …



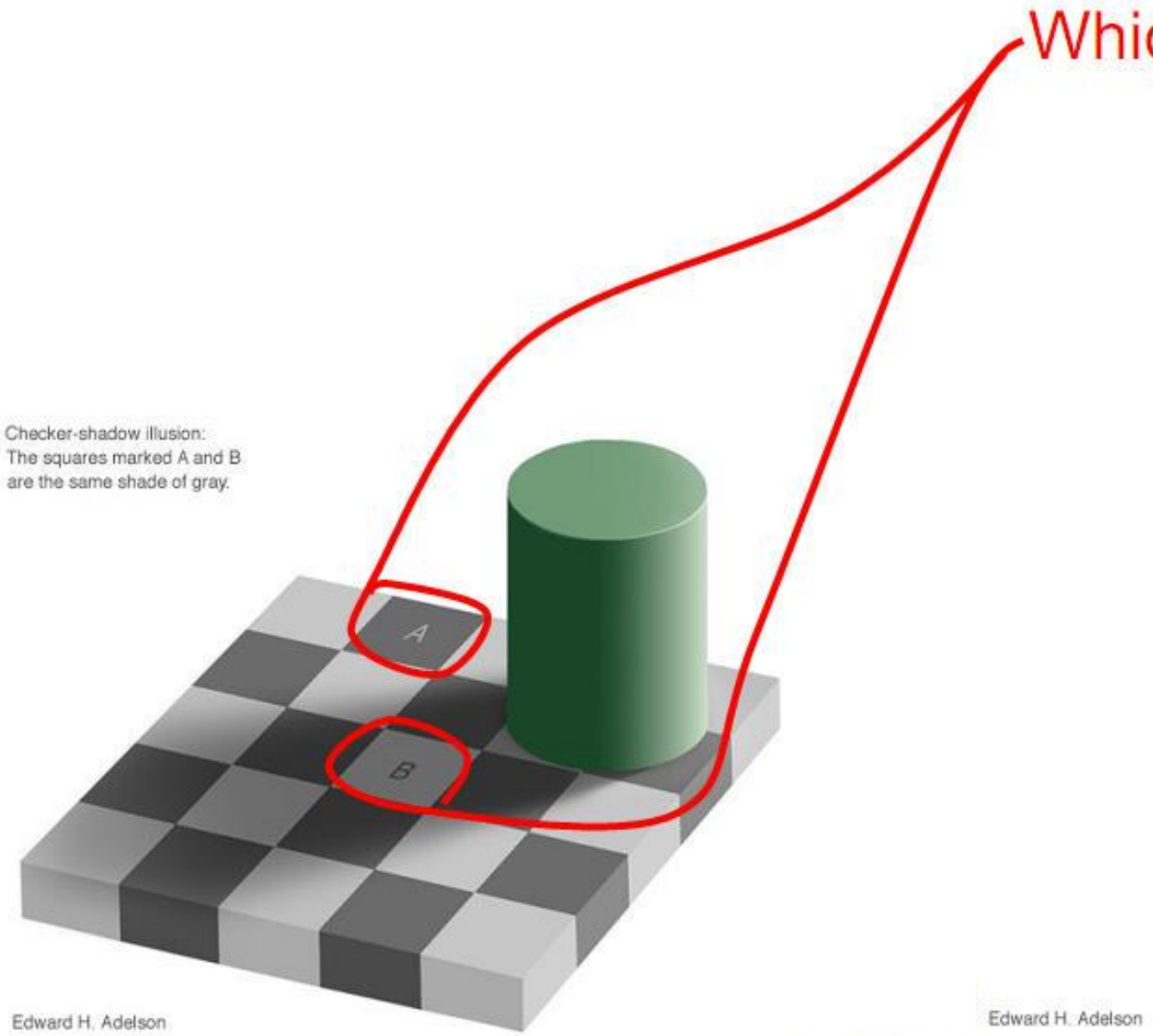
Edward H. Adelson

Color Changes with Lighting



Use context to stabilize colors

Must deal with Lighting Changes …



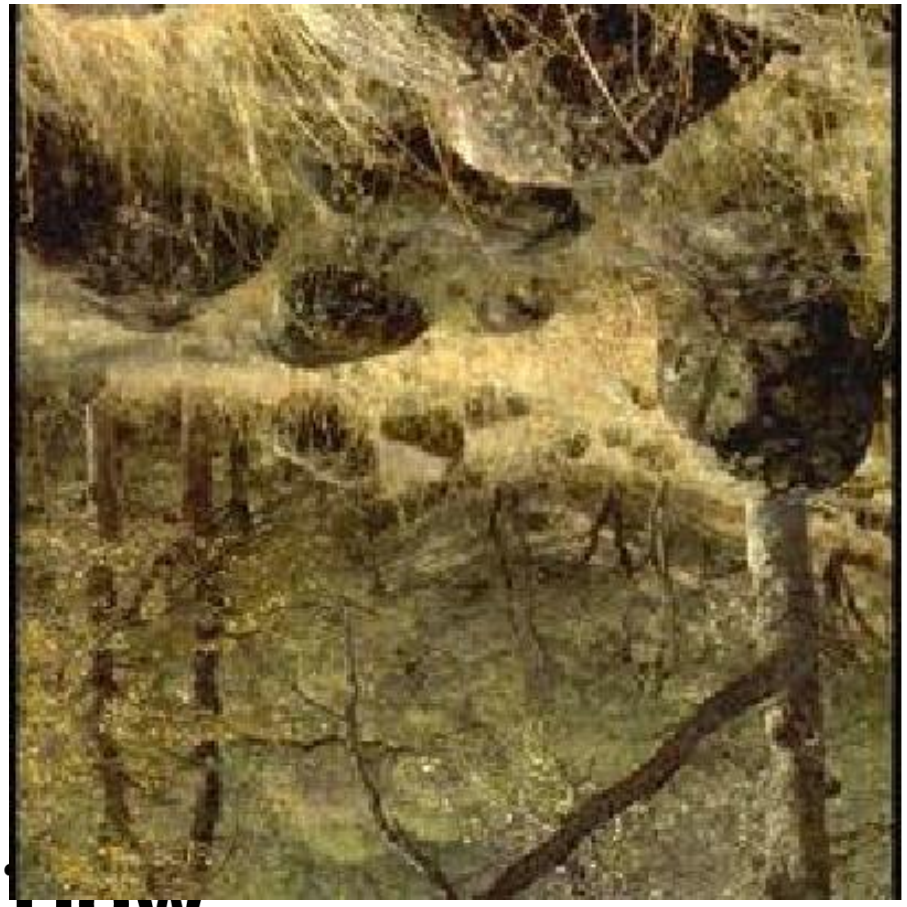
Checker-shadow illusion:
The squares marked A and B
are the same shade of gray.

Edward H. Adelson

Edward H. Adelson

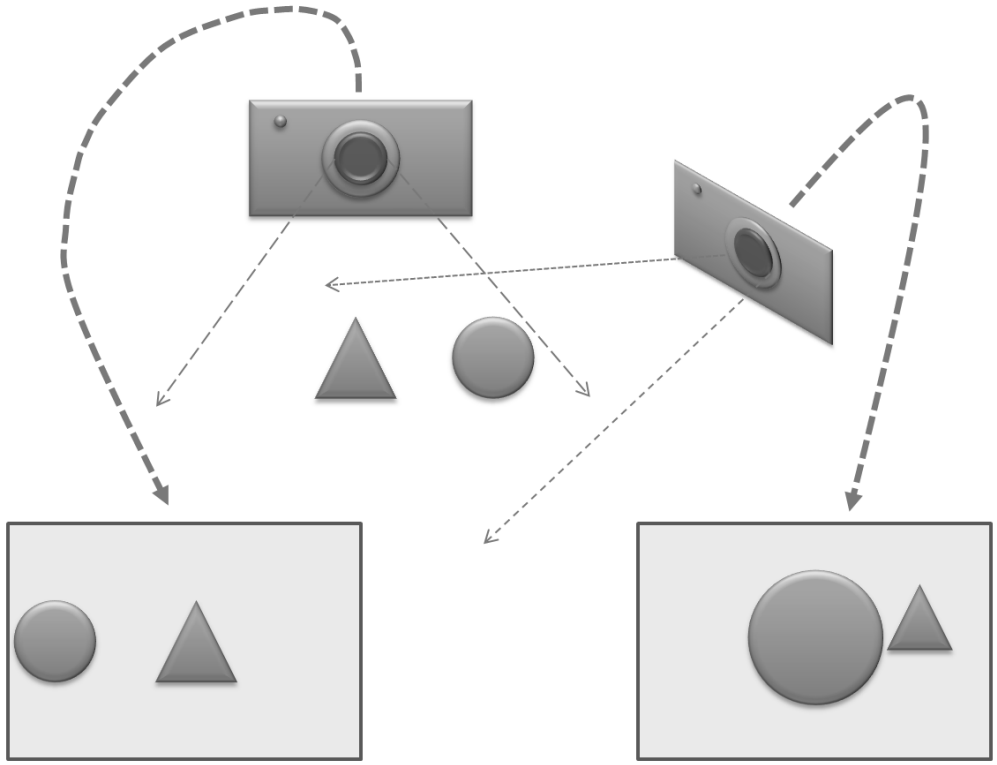
Lighting

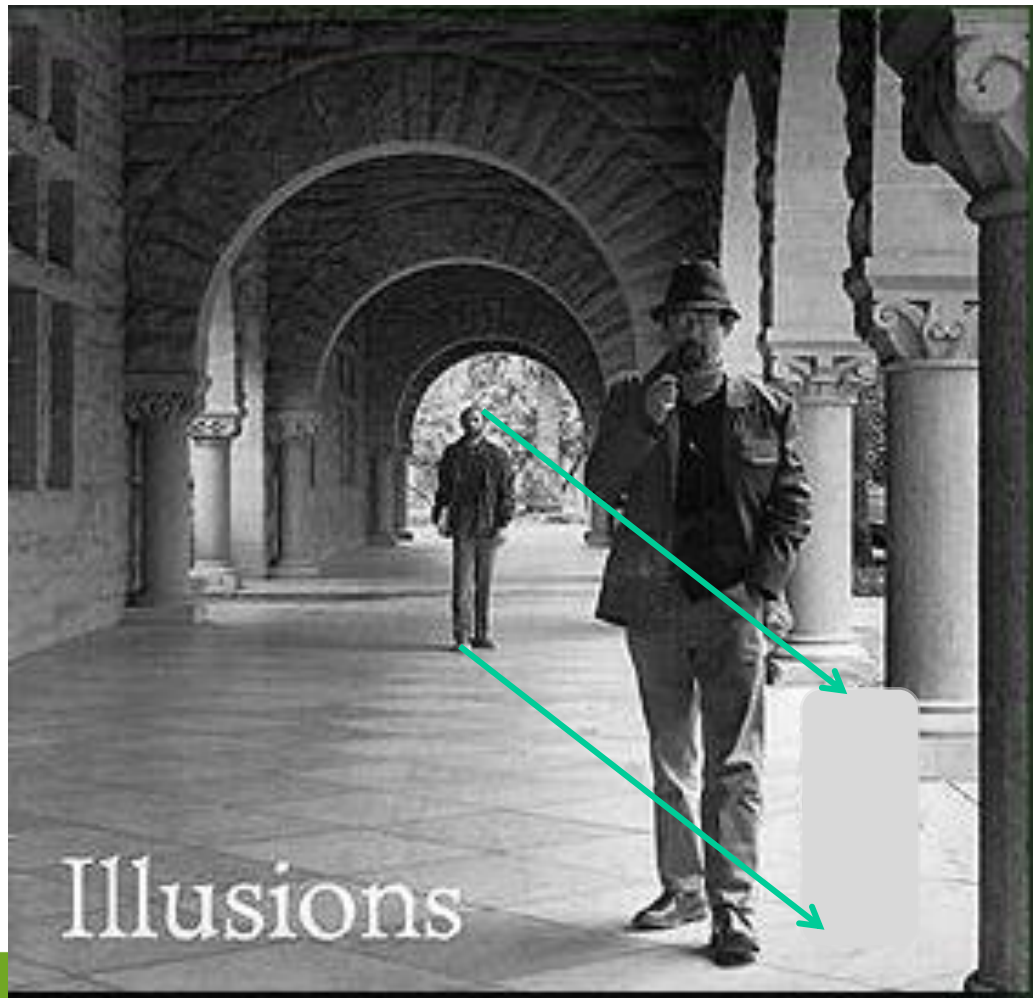
Perception of surfaces depends on lighting assumptions



The Brain Assumes 3D Geometry

Perception is ambiguous … depending on your point of view!

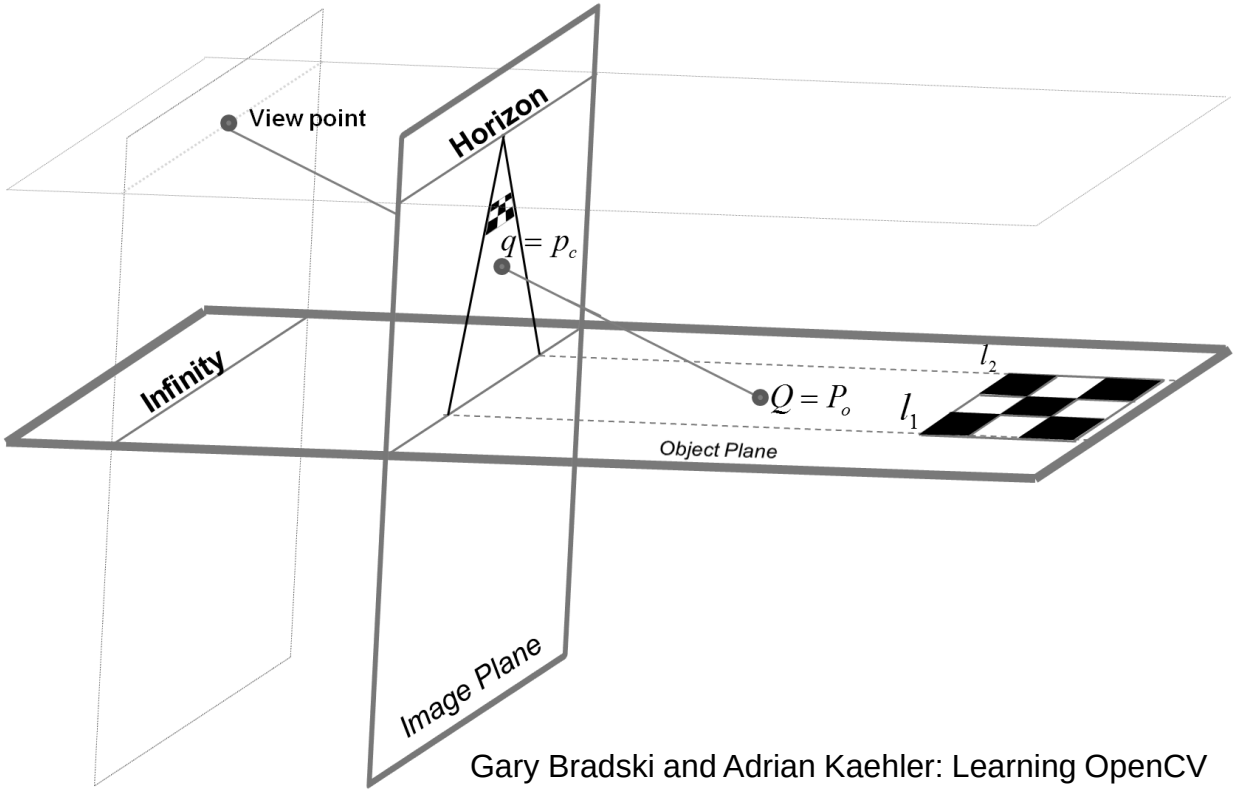




Willow Garage

Consequence of Projective Imaging: **Parallel lines meet**

- There exist vanishing points



Gary Bradski and Adrian Kaehler: Learning OpenCV

Marc Pollefeys

Consequences* for YOUR Perception

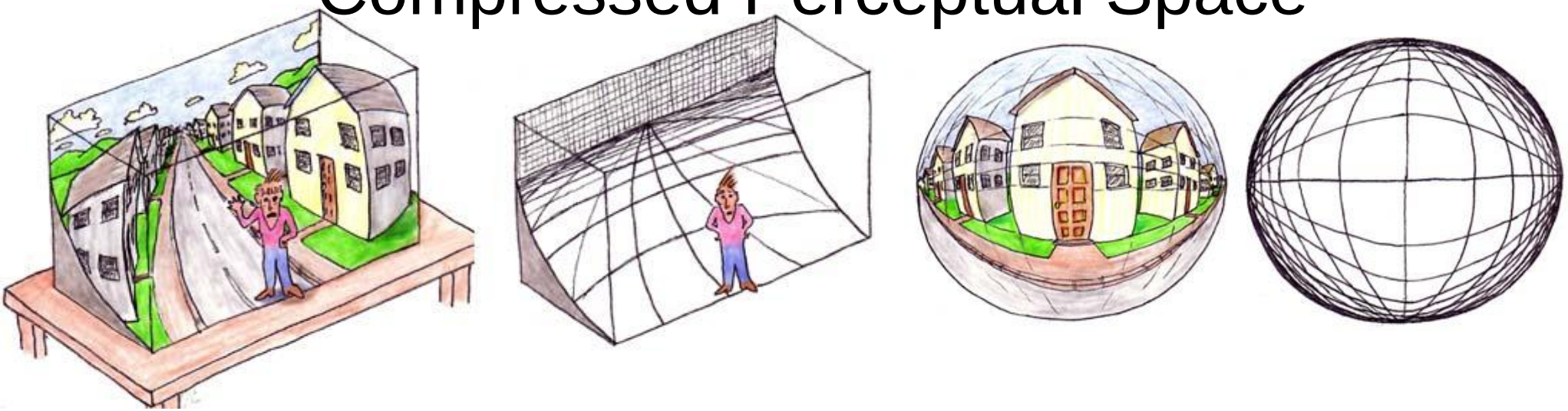
Visual Metrics are Strange

Same size things get smaller, we hardly notice…

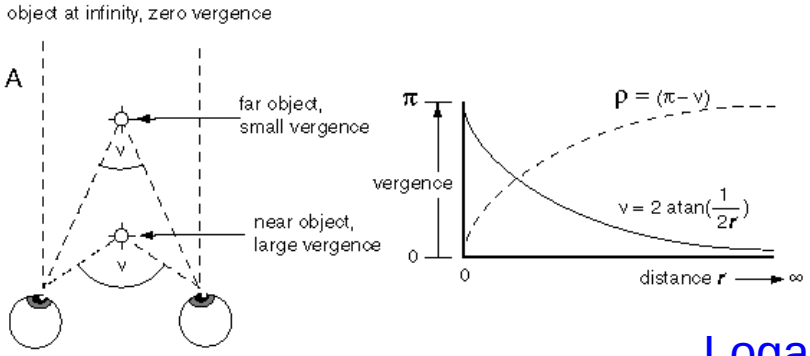
Parallel lines meet at a point…

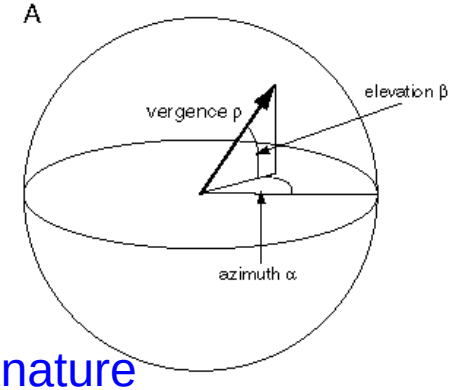
* A Cartoon Epistemology: http://cns-alumni.bu.edu/~slehar/cartoonepist/cartoonepist.html

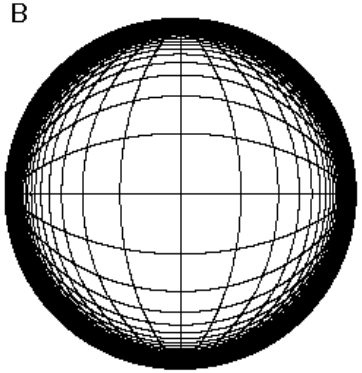
Vergence Implies a Logarithmically Compressed Perceptual Space



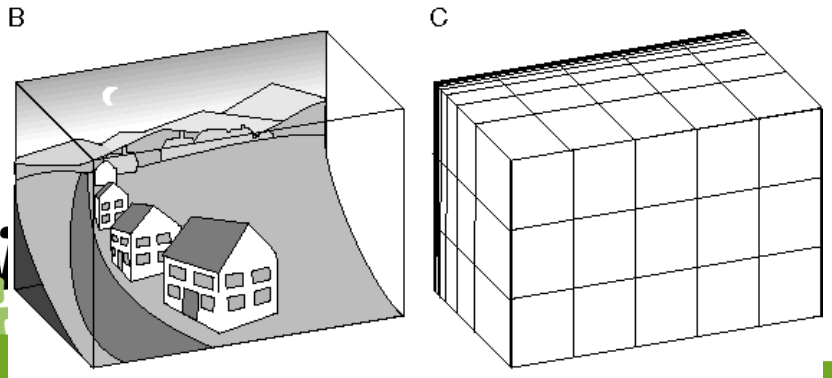
Perception must be mapped to a space variant grid

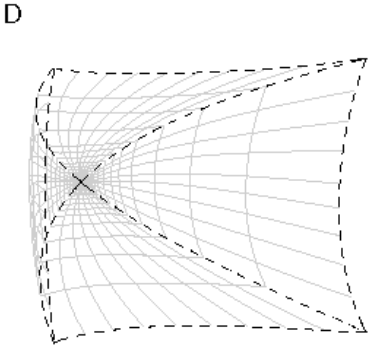






Logarithmic in nature





60

Questions?