Open Source Computer Vision Library *in Robotics*

Gary Bradski

Senior Scientist, Willow Garage;

Consulting Prof, Stanford University

Willow Garage OpenCV Team:

Vadim Pisarevsky: Team Lead

Victor Eruhimov: Senior Researcher

Alexey Latyshev

Maria Dimashova

Anatoly Baksheev

??

For the rest, cite:

Bradski, G.; Kaehler, A., "Learning OpenCV: Computer Vision with the OpenCV Library ", O'Reilly Press, 2008

http://www.amazon.com/Learning-OpenCV-Computer-Vision-Library/dp/0596516134

Outline

- Why OpenCV?
- What is OpenCV
- Use in Robotics at Willow Garage
- OpenCV Overview/Tour
- OpenCV Plans – Summer Release
- Questions?

Why is Vision Hard?

The difference between seeing and perception.

We perceive this:

What to do?

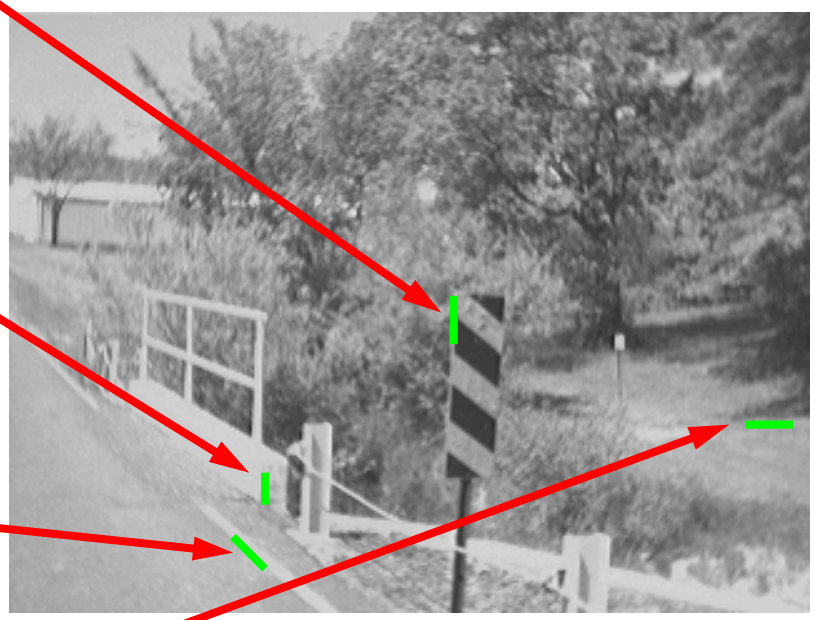
Maybe we should try to find edges ….

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

But, What's an Edge?

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



Slide credit: Christopher Rasmussen

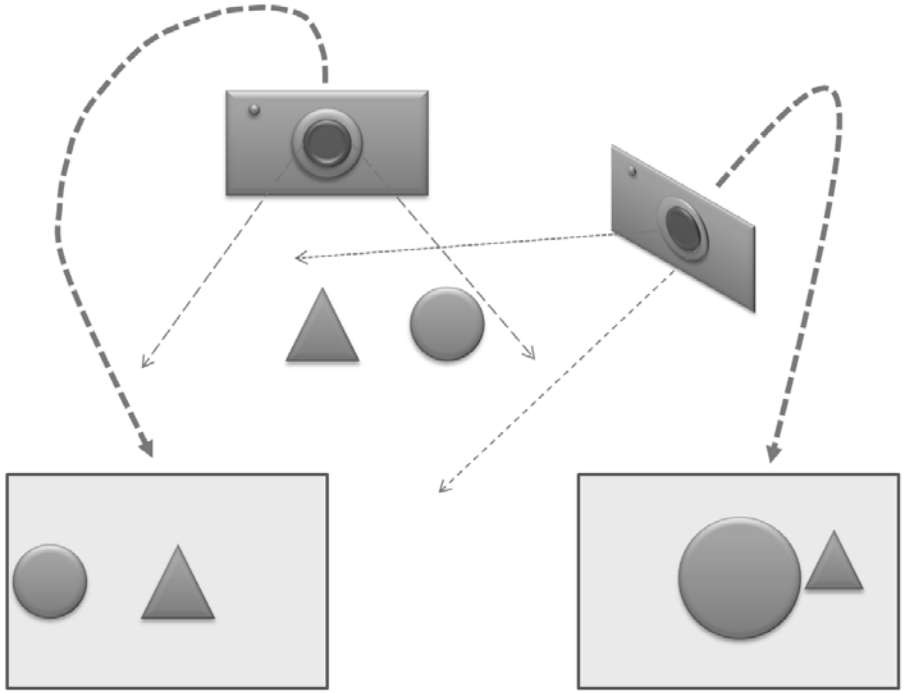
To Deal With the Confusion,
Your Brain has Rules...
That can be wrong

We even see invisible edges…

And surfaces …

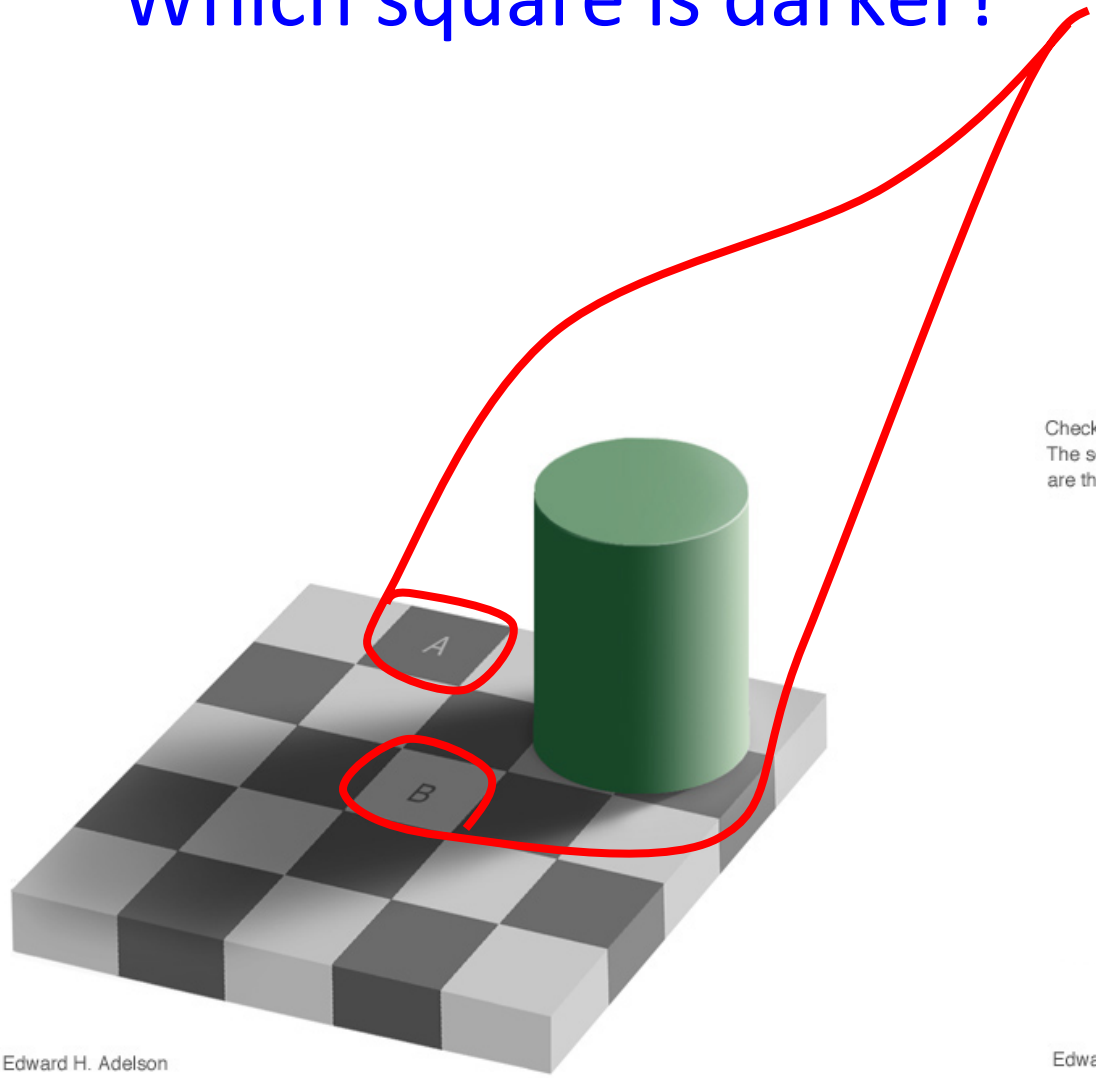
We need to deal with 3D Geometry

Perception is ambiguous … depending on your point of view!



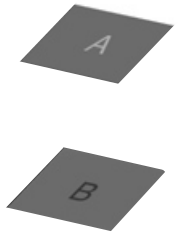
And Lighting in 3D

Which square is darker?



Edward H. Adelson

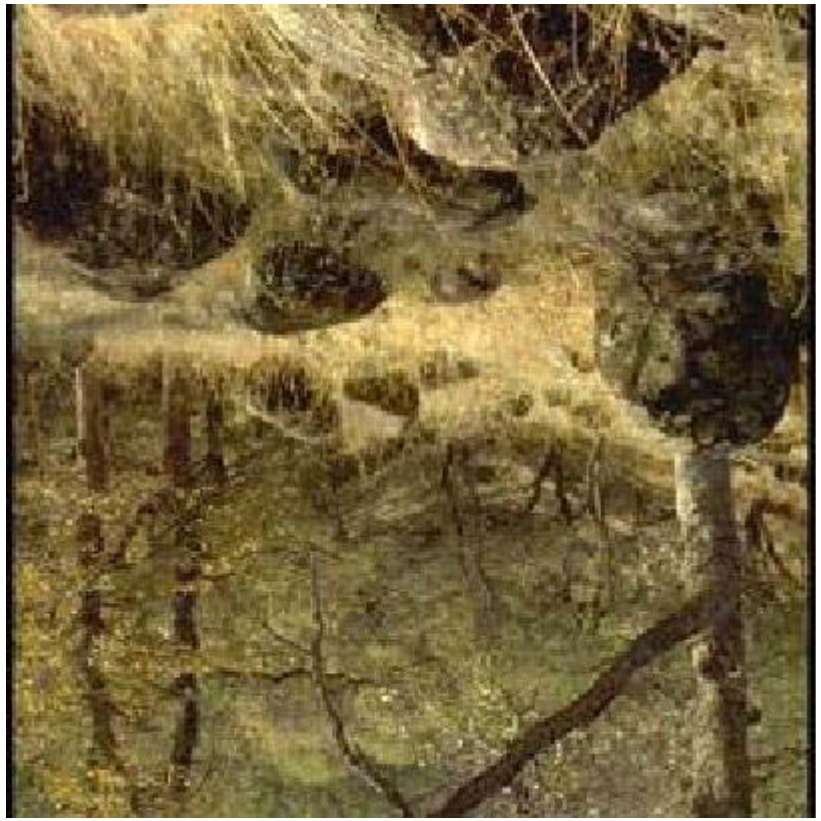
Checker-shadow illusion:
The squares marked A and B
are the same shade of gray.



Edward H. Adelson

Lighting is Ill-posed …

Perception of surfaces depends on lighting assumptions



Lighting -- Contrast

Which one is **male** and which one is **female**?

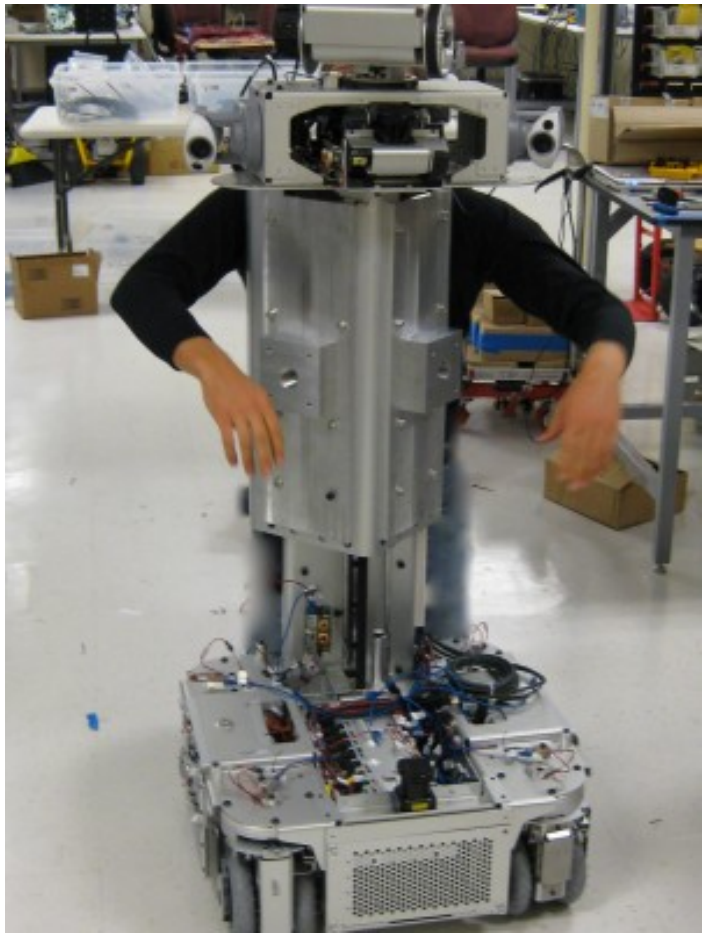
Illusion by: Richard Russell, Aarvard University

Consequences

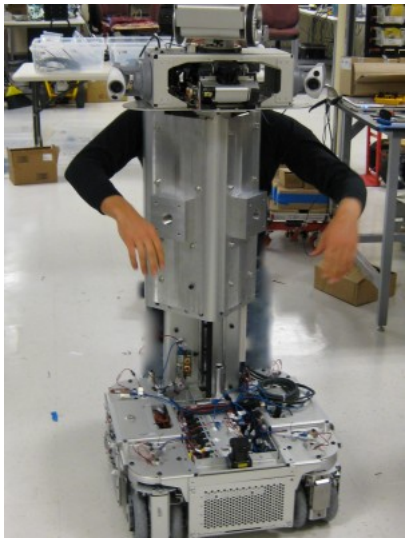
- Approaches to vision are
  - Fragmented
  - Complex
- People re-invent the infrastructure
- Groups without good infrastructure suffer

Towards Competent Perception

- In order to move towards competent perception,
- We need tools to bridge the gap

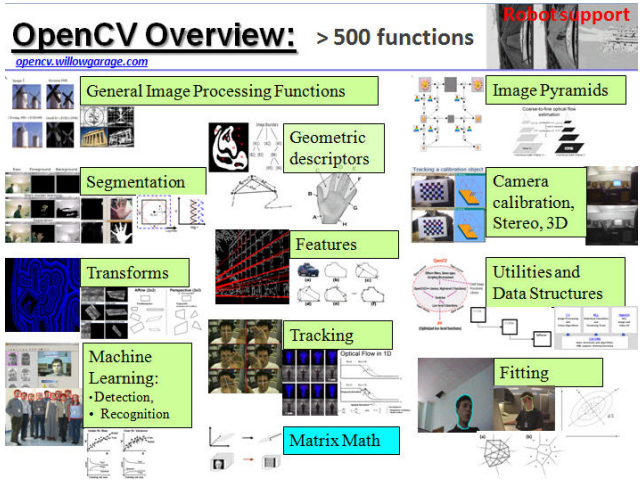


Solution



- Provide a free, open infrastructure
- Make available:
  - Basic algorithms, image processing
  - But also advanced techniques
- Some key applications (face detection)
  - More coming

OpenCV



Outline

- Why OpenCV?
- What is OpenCV
- Use in Robotics at Willow Garage
- OpenCV Overview/Tour
- OpenCV Plans – Summer Release
- Questions?

What is OpenCV?

OpenCV stands for the **Open** Source **C**omputer **V**ision Library.

- It was founded at Intel in 1999, went through some lean years after the .bust but is now under active development, now receiving ongoing support from **Willow Garage**.

OpenCV is **free** for commercial and research use.

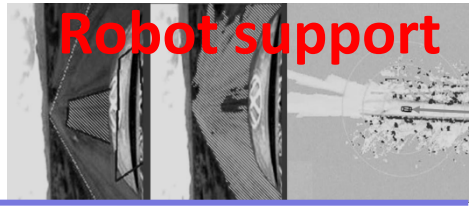
- It has a **BSD license**. The library runs across many platforms and actively supports Linux, Windows and Mac OS.

OpenCV was founded to advance the field of computer vision.

- It gives everyone a reliable, real time infrastructure to build on. It collects and makes available the most useful algorithms.

OpenCV Overview: > 500 functions

opencv.willowgarage.com



General Image Processing Functions

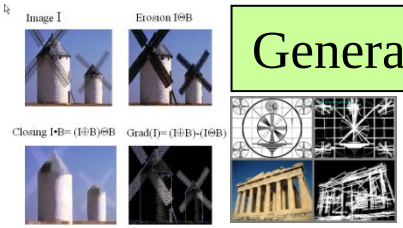
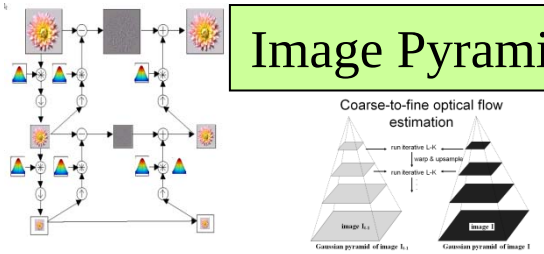
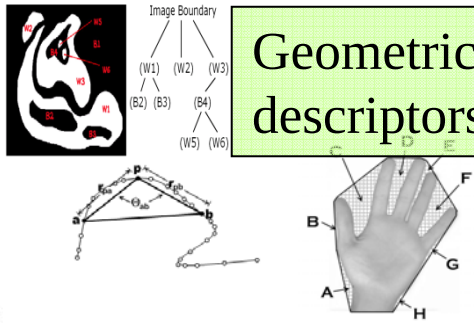


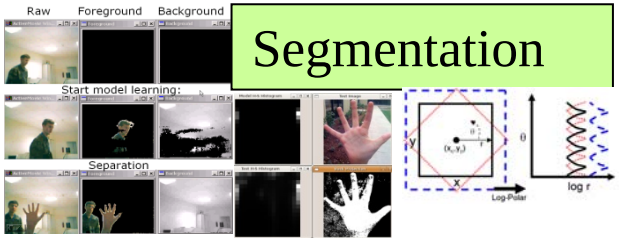
Image Pyramids



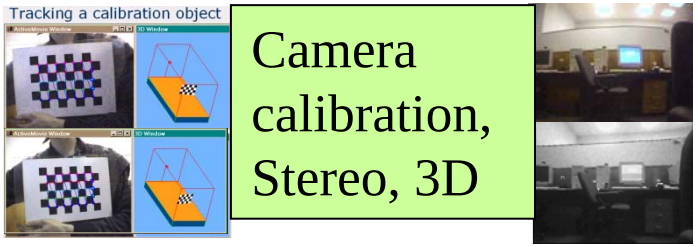
Geometric descriptors



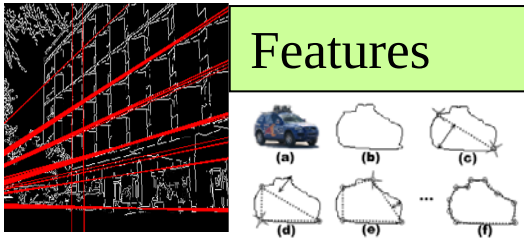
Segmentation



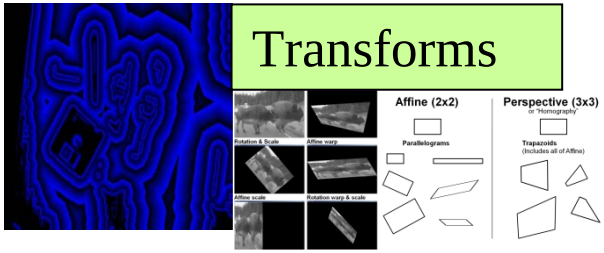
Camera calibration, Stereo, 3D



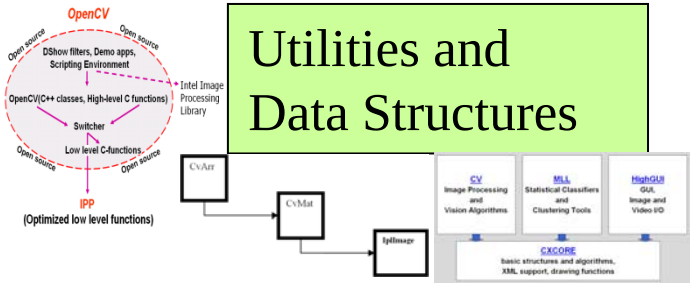
Features



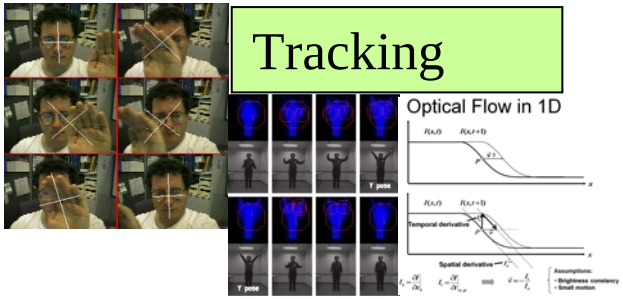
Transforms



Utilities and Data Structures

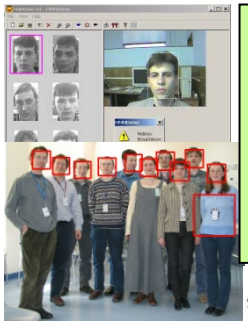


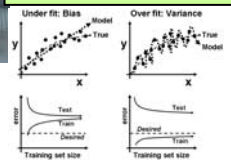
Tracking



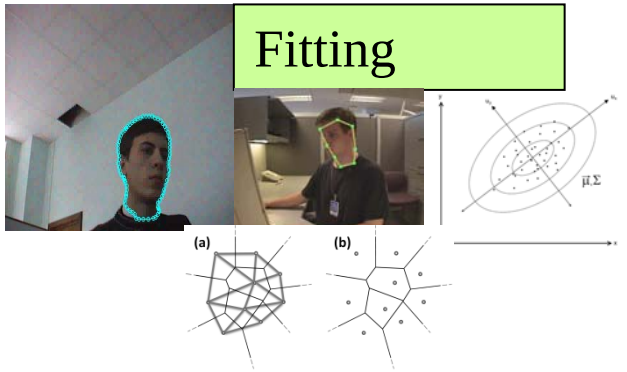
Machine Learning:

- Detection,
- Recognition

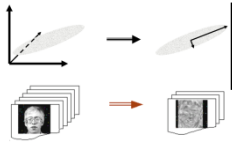




Fitting

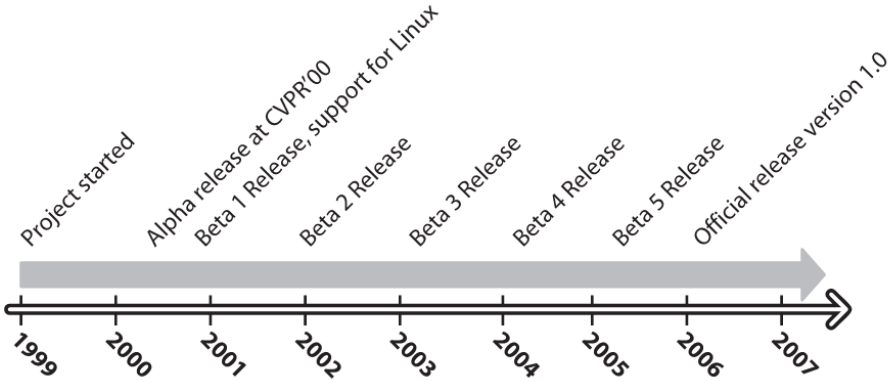


Matrix Math



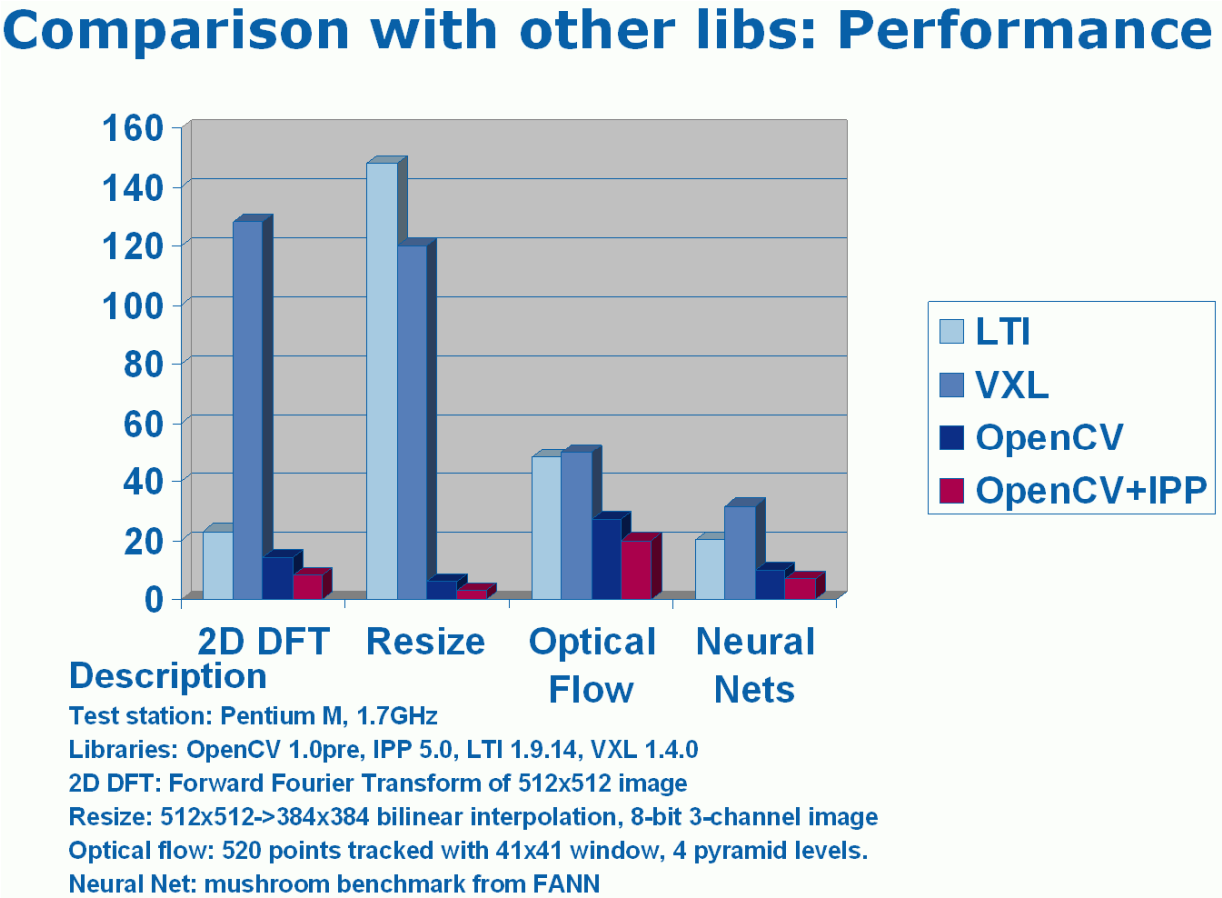
OpenCV History

- Motivation was to lower the bar to entry to computer vision, increase incentive for higher MIPS out in the market for Intel.
- Timeline:



- Willow Garage www.willowgarage.com now supports OpenCV (and a robot OS called ROS) as a means of accelerating peaceful uses of robotics and AI

OpenCV Tends Towards Real Time



License

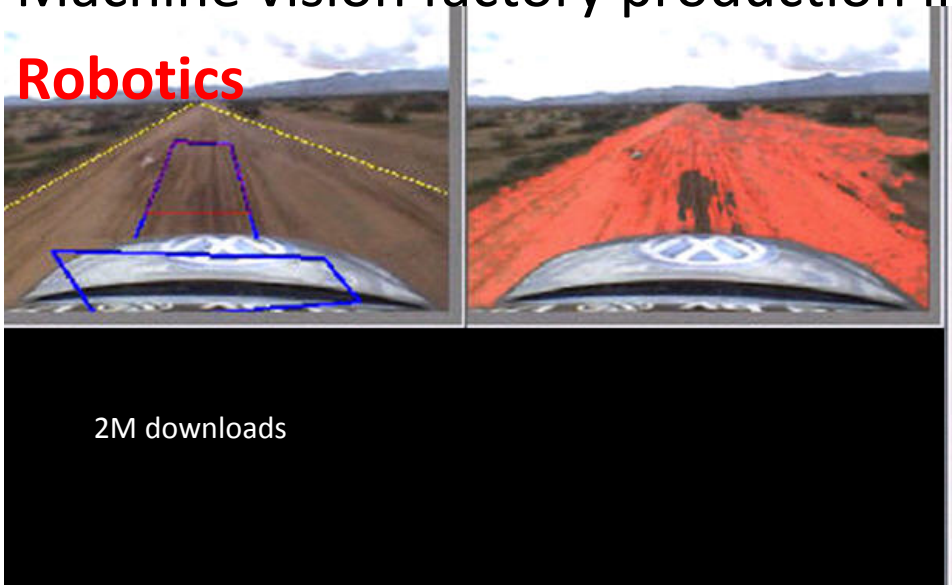
- Based on BSD license
- Free for commercial or research use
  - In whole or in part
  - Does not force your code to be open
  - You need not contribute back
    - We hope you will contribute back, recent contribution, C++ wrapper class used for Google Street Maps*

* Thanks to Daniel Filip

Where is OpenCV Used?

- Google Maps, Google street view, Google Earth, Books
- Academic and Industry Research
- Safety monitoring (Dam sites, mines, swimming pools)
- Security systems
- Image retrieval
- Video search
- Structure from motion in movies
- Machine vision factory production inspection systems
- **Robotics**

- *Well over 2M downloads*



Useful OpenCV Links

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

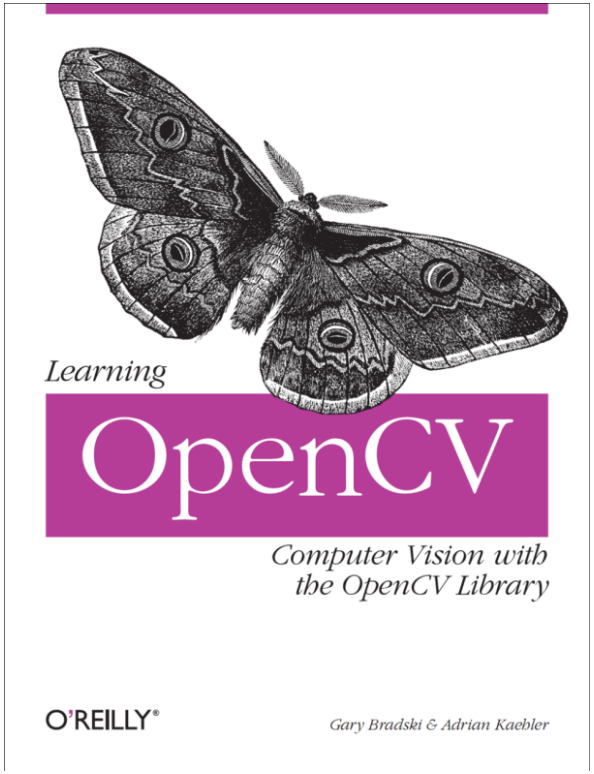
User Group:
http://tech.groups.yahoo.com/group/OpenCV/join

OpenCV Code Repository:
http://sourceforge.net/projects/opencvlibrary/

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/



Outline

- Why OpenCV?
- What is OpenCV
- Use in Robotics at Willow Garage
- OpenCV Overview/Tour
- OpenCV Plans – Summer Release
- Questions?

Willow Garage

- Aims to rapidly accelerate robotic capabilities
  - Strategy is to foster open HW, SW and open collaboration
    - ROS (Robot Operating System)
    - OpenCV
- Drive the work via milestones. Each one
  - proves capability
  - is useful in itself
  - can be shown to be solved

Willow Garage Milestones for PR2

- M1 (Navigation)
  - Navigate around an active office building for PI Kilometers a day for 2 days in a row without needing human intervention. **[DONE]**
- M2 (Manipulation/Marathon)
  - Navigate for 26 miles in a large busy office building
  - Plug itself in and open doors as needed. **[In Process]** **~**June
- M3 (Software)~August
  - Bring in groups who must
    - be able to achieve a substantial robotic task (say, clean a table)
    - In one week
    - From scatch
- M4 (Ship Robots)~End of Year

Opening Doors

Perception:

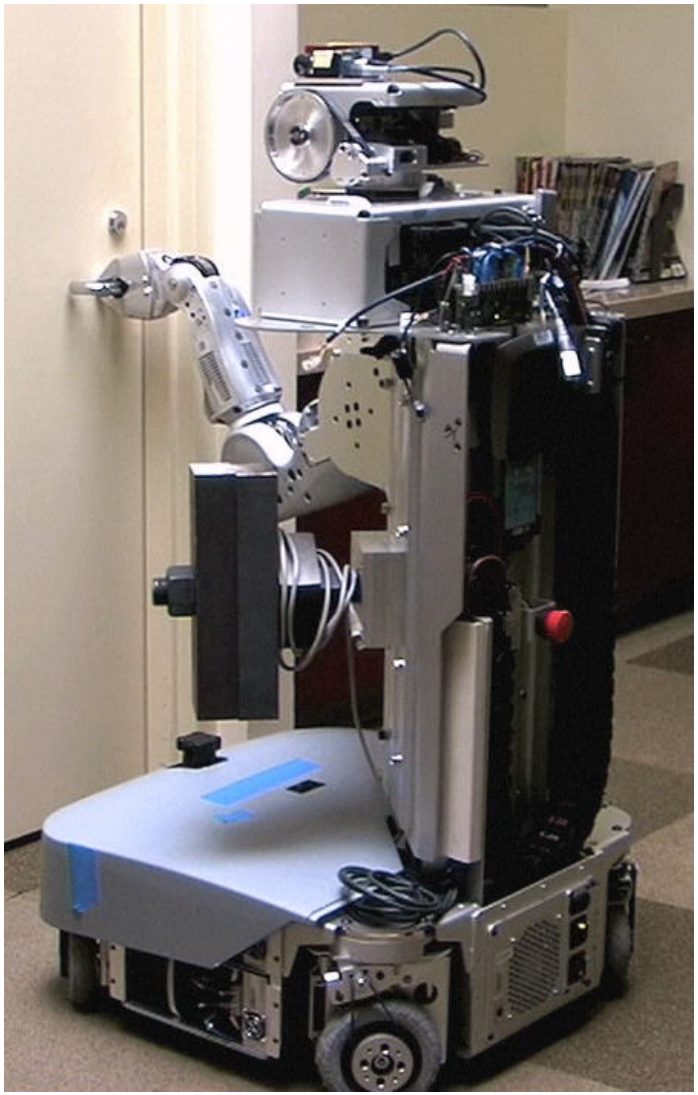
• Laser in base is used to establish plane of door.

•Panning laser finds high reflectance of handle.

•Stereo camera finds depth discontinuities at handle location.

• Computer vision Adaboost Cascade identifies handle location.

•Location is mapped to 3D location in hand coordinate system.



Plugging in

Perception:

•Stereo finds candidate outlet location from a distance (4 meters)

•Base laser is used to find the wall normal to allow the robot to drive to a spot 1 meter from the potential outlet.

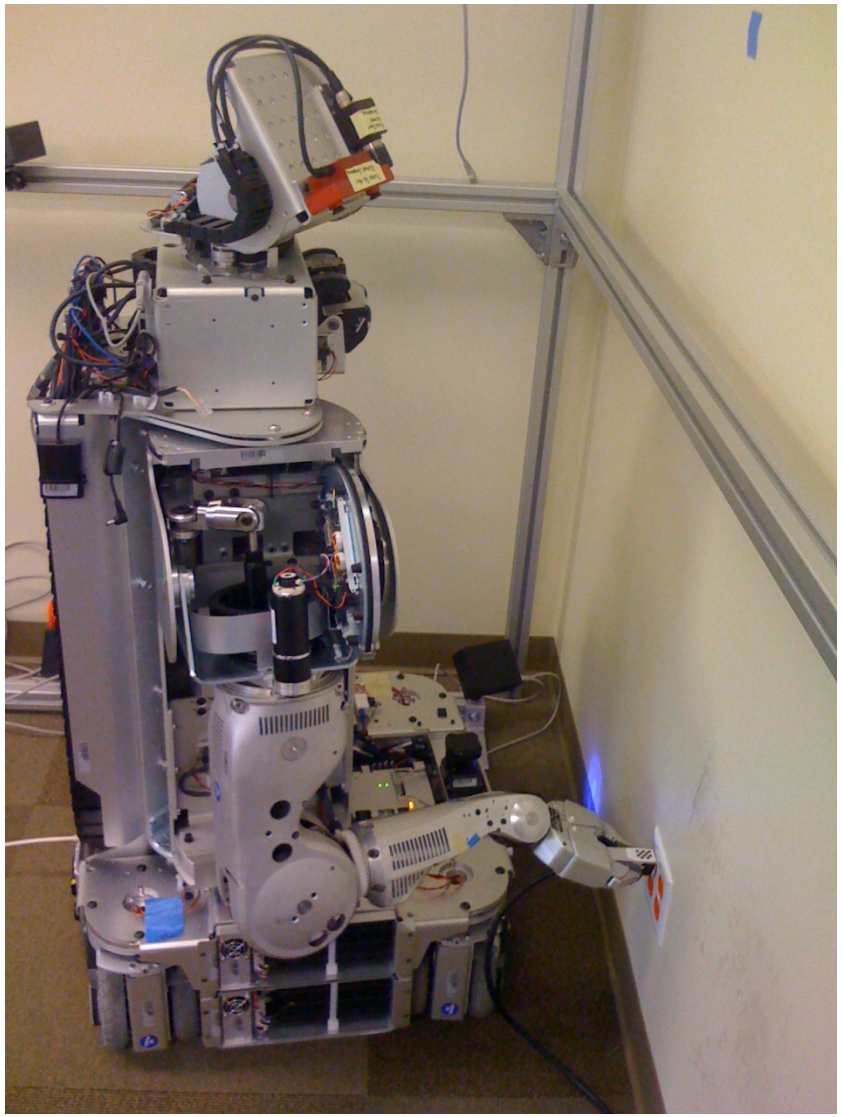
•Laser is used to compute homography of likely outlet locations and map them to cannonical locations.

•The outlet location is confirmed or denied.

•If found, the homography is made precise so that oultet holes are localized to less than 2mm accuracy.

•Vision localizes the plug to sub-milimeter accuracy.

•Visual servoing is used to plug in.



Computer Vision Datasets

- There are many data sets (http://elm.eeng.dcu.ie/~oconaire/cv_datasets.html):

•WallFlower dataset: For evaluating background modelling algorithms. Ground-truth foreground provided.
•Foreground/Background segmentation and Stereo dataset: from Microsoft Cambridge.
•VISOR: Video Surveillance Online Repositiory: Lots of videos and ground truth.
•3D Photography Dataset
•Multi-model, multi-camera meeting room dataset
•Advanced Video and Signal based Surveillance: a variety datasets for tracking and detection.
•Caltech image collections: object detection, segmentation and classification
•INRIA Datasets: Cars, people, horses, human actions, etc.
•CAVIAR surveillance Dataset
•Videos for Head Tracking
•Pedestrian dataset from MIT
•Shadow detection datasets
•Flash and non-Flash dataset
•Experiments on skin region detection and tracking: it includes a ground-truthed dataset
•MIT Face Dataset
•MIT Car Datasets
•MIT Street Scenes: CBCL StreetScenes Challenge Framework is a collection of images, annotations, software and performance measures for object detection [cars, pedestrians, bicycles, buildings, trees, skies, roads, sidewalks, and stores]
•LabelMe Dataset: Over 150,000 images with objects annotated and labelled.
•MuHAVi: Multicamera Human Action Video DataA large body of human action video data using 8 cameras. Includes manually annotated silhouette data.
•INRIA Xmas Motion Acquisition Sequences (IXMAS): Multiview dataset for view-invariant human action recognition.
•i-LIDS datasets: UK Government benchmark datasets for automated surveillance.

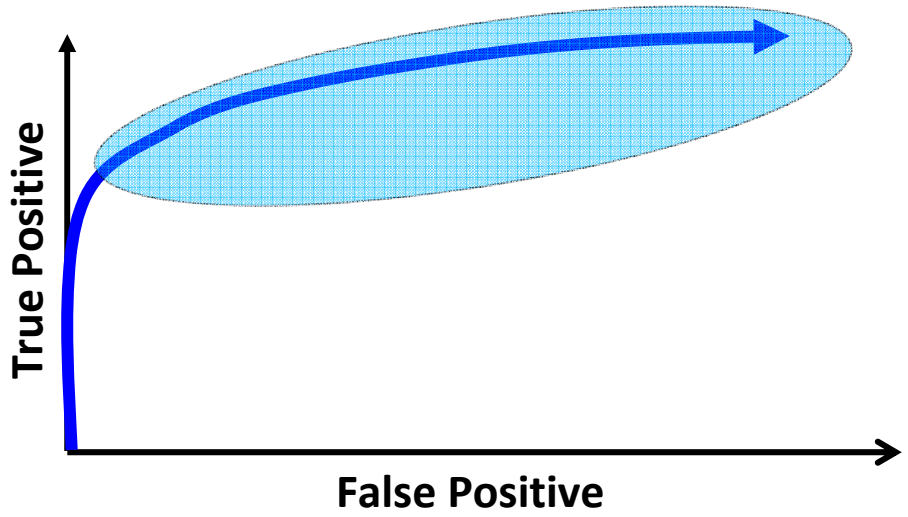
Dataset lists

•List of Databases: Includes multiple face datasets, texture datasets, etc.
•UIUC Datasets: Includes... Fifteen Scene Categories, 3D Object Recognition Stereo Dataset, 3D Photography Dataset, Visual Hull Datasets, Birds, Butterflies, Object Recognition Database, Texture Database and Video Sequences.
•OTCBVS Datasets: Several datasets that include non-visual data, such as thermal infrared and NIR.

Computer Vision challenges

- There are many data set challenges:
  - PASCAL VOC,
  - CalTech101
  - CalTech256 …
  - CVPR semantic robot vision challenge

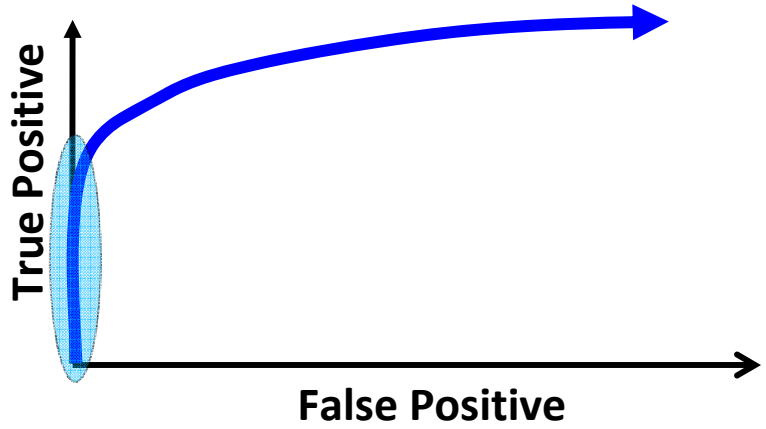
All of these explore the False Positive region of the ROC curve



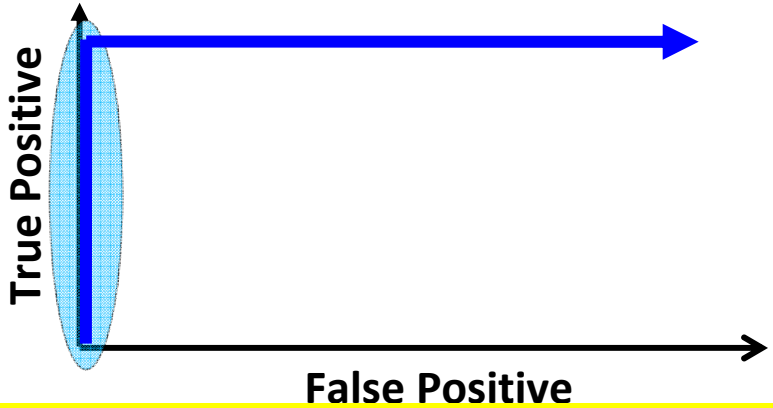
This is important work for the longer term…

Willow wants to help fill the gap in the shorter term

- We want to publically establish **solved** problems in vision
- This corresponds to exploring the True Positive part of the curve



- And push it to: “solved” for increasing classes of data:



Start with “easy” problems, advance from there

Solve It Perception Challenge

Ikea Manipulation Database:

- **Given 3D models of 20-50 mostly Lambertian items**
- **100's of labeled images of each item by itself and with others**
- **With corresponding dense stereo depth maps**
- **And the item itself (from your local Ikea)**

Can you get a robot to correctly identify 100% of the items in 3 lighting conditions on tables and shelves in a room?

Scanning:

Physical Ikea Database:

3D data:

Outline

- Why OpenCV?
- What is OpenCV
- Use in Robotics at Willow Garage
- OpenCV Overview/Tour
- OpenCV Plans – Summer Release
- Questions?

OpenCV Tour

- Installing OpenCV
- Useful structures and conventions
- I/O
- Function Overview
- Notable functions
  - ❖ Machine Learning

Installing OpenCV from Release Packages

You can get the latest official releases from Sourceforge:
http://sourceforge.net/project/showfiles.php?group_id=22870

Installation on Windows involves simply running the installer that comes with the package.

Installation on Linux is more involved since it has dependencies on other packages such as ffmpeg. You can find instructions on the OpenCV wiki:
http://opencv.willowgarage.com/wiki/

Installing the latest OpenCV from SVN

Building on Windows, Mac OS or Linux
is now unified by Cmake. Just go to:
http://opencv.willowgarage.com/wiki/InstallGuide
and follow the instructions.

On SVN you get the very latest code, perhaps with some bugs in the new routines. It is now updated daily.

Linux: Installing OpenCV from SVN

Installing OpenCV on Linux

Here is brief description on how to build the latest version of OpenCV. See INSTALL file shipped with OpenCV for more detail. The required packages are listed below:

* GCC 4.x or later. In Ubuntu/Debian it can be installed with

```
sudo apt-get install build-essential
```

* CMake 2.6 or higher
* Subversion (SVN) client
* GTK+ 2.x or higher, including headers (e.g. libgtk2.0-dev)
* pkgconfig
* libpng, zlib, libjpeg, libtiff, libjasper with development files (e.g. libjpeg-dev)
* Python 2.3 or later with developer packages (e.g. python-dev)
* SWIG 1.3.30 or later
* libavcodec etc. from ffmpeg 0.4.9-pre1 or later + headers. In Ubuntu 8.10 all the necessary ffmpeg files can be installed with:

```
sudo apt-get install libavformat-dev
```

* libdc1394 2.x + headers for video capturing from IEEE1394 cameras

Getting OpenCV source code from the repository

Enter your working directory (denoted as ~/<your_working_dir> below) and type:

```
cd ~/<your_working_dir>
svn co https://opencvlibrary.svn.sourceforge.net/svnroot/opencvlibrary/trunk
```

to get the current OpenCV snapshot, or

```
cd ~/<your_working_dir>
svn co
https://opencvlibrary.svn.sourceforge.net/svnroot/opencvlibrary/tags/latest_tested_snapshot
```

to get the latest tested OpenCV snapshot.

Compilation

To build OpenCV using cmake, type the following:

```
cd ~/<your_working_dir>/opencv  # the directory should contain CMakeLists.txt,
                                # INSTALL etc.
mkdir release # create the output directory
cd release
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D
BUILD_PYTHON_SUPPORT=ON ..
```

You will see the configuration results. Various build options can be adjusted interactively using CMake GUI. Then you build the library and install it to the target directory:

```
make
sudo make install
```

Path Configuration

In order to use the libraries in Linux, thier path should be specified. Library path can be specified in /etc/ld.so.conf.d/ by creating a file called 'opencv.conf' which contains the opencv library path (Default configuration is /usr/local/lib). Once the file is created, execute

```
sudo ldconfig -v
```

to make new set library pathes effective. Or you can also add the path to LD_LIBRARY_PATH:

```
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

Test OpenCV

You can run the correctness tests cxcoretest and cvtest for a quick sanity check:

```
cd ~/<your_working_dir>/opencv/release/bin
./cxcoretest
./cvtest -d ~/<your_working_dir>/opencv/tests/cv/testdata
```

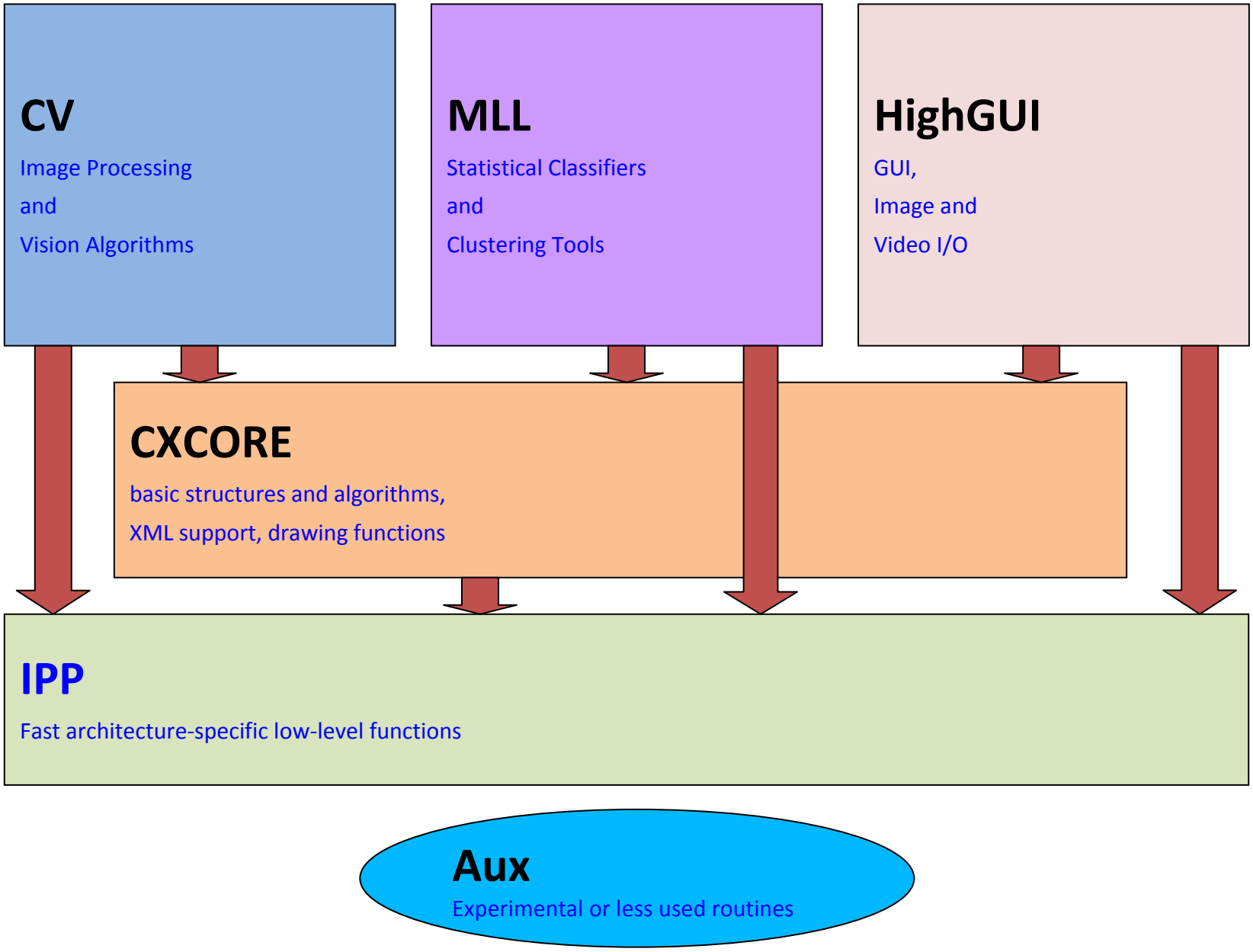
You can also build and run OpenCV samples:

```
cd ~/<your_working_dir>/opencv/samples/c
. build_all.sh
./delaunay
# try other samples (some of them require a camera) ...
```

OpenCV Tour

- Installing OpenCV
- Useful structures and conventions
- I/O
- Function Overview
- Notable functions
  - ❖ Machine Learning

OpenCV Structure



OpenCV Conventions

*** Function naming conventions:**

cvAlgorithm<Target><Mod>(...)

Algorithm = the core functionality (e.g. set, create)
Target = the target image area (e.g. contour, polygon)
Mod = optional modifiers (e.g. argument type)

*** Matrix data types:**

CV_<bit_depth>(S|U|F)C<number_of_channels>

S = Signed integer
U = Unsigned integer
F = Float

E.g.: CV_8UC1 means an 8-bit unsigned single-channel matrix,
CV_32FC2 means a 32-bit float matrix with two channels.

*** Image data types:**

IPL_DEPTH_<bit_depth>(S|U|F)

E.g.: IPL_DEPTH_8U means an 8-bit unsigned image.
IPL_DEPTH_32F means a 32-bit float image.

*** Header files:**

```
#include <cv.h>
#include <cvaux.h>
#include <highgui.h>
#include <cxcore.h>   // unnecessary - included in cv
#include <ml.h>       // machine learning library
```

*** Many data structures have in line constructors or macros**

- Structures start with upper case, inline constructors start with lower case
- For example struct **C**vPoint may be constructed by:
  - **c**vPoint(x,y);

- Inline usage example:

Draw a red line between two points:
Prototype:
void cvLine(CvArr img, CvPoint pt1, CvPoint pt2, CvScalar color, int thickness=1, int line_type=8, int shift=0);*
Call:
cvLine(Image,cvPoint(3,5), cvPoint(300,40), CV_RGB(255,0,0));

Modified fom (somewhat dated) tutorial at: http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

Highly Used Utility Data Structures

Structure:	**In Line Constructor:**
CvSize(int width, int height); CvSize2D32f(float width, float height);	CvSize foo = cvSize(int width, int height); CvSize2D32f foo = cvSize2D32f(float width, float height);
CvRect(int x, inty, int width, int height);	CvRect foo = cvRect(int x, int y, int width, int height);

CvScalar(double val[4]);	CvScalar foo = cvScalar(val0=0, val1=0,val2=0,val3=0);
	= cvScalarAll(double val); //set all
	=cvRealScalar(val0); //1st val is set, others = 0

CvPoint(int x, int y) ; CvPoint2D32f(float x, float y); CvPoint2D64f(double x,double y);	CvPoint foo = cvPoint(int x, int y); CvPoint2D32f foo = cvPoint2D32f(float x, float y); CvPoint2D32f foo = cvPointTo32f(cvPoint(int x, int y)); CvPoint2D64f foo = cvPoint2D64f(double x,double y);
CvPoint3D32f(float x, float y, float z); CvPoint3D64f(double x,double y,double z);	CvPoint3D32f foo=cvPoint3D32f(double x,double y,double z); CvPoint3D64f foo=cvPoint3D64f(double x,double y,double z);

Useful Defines

#define CV_RGB(r, g, b) cvScalar((b), (g), (r))

IPL_DEPTH_8U - unsigned 8-bit integers
IPL_DEPTH_8S - signed 8-bit integers
IPL_DEPTH_16U - unsigned 16-bit integers
IPL_DEPTH_16S - signed 16-bit integers
IPL_DEPTH_32S - signed 32-bit integers
IPL_DEPTH_32F - single precision floating-point numbers
IPL_DEPTH_64F - double precision floating-point numbers

define CV_SWAP(a,b,t) ((t) = (a), (a) = (b), (b) = (t))
#ifndef MIN
#define MIN(a,b) ((a) > (b) ? (b) : (a))
#endif
#ifndef MAX
#define MAX(a,b) ((a) < (b) ? (b) : (a))
#endif

#define CV_LOAD_IMAGE_UNCHANGED -1
#define CV_LOAD_IMAGE_GRAYSCALE 0
#define CV_LOAD_IMAGE_COLOR 1
#define CV_LOAD_IMAGE_ANYDEPTH 2
#define CV_LOAD_IMAGE_ANYCOLOR 4

cvFloor(double v)
cvCeil(double v);

#define CV_EVENT_MOUSEMOVE 0
#define CV_EVENT_LBUTTONDOWN 1
#define CV_EVENT_RBUTTONDOWN 2
#define CV_EVENT_MBUTTONDOWN 3
#define CV_EVENT_LBUTTONUP 4
#define CV_EVENT_RBUTTONUP 5
#define CV_EVENT_MBUTTONUP 6
#define CV_EVENT_LBUTTONDBLCLK 7
#define CV_EVENT_RBUTTONDBLCLK 8
#define CV_EVENT_MBUTTONDBLCLK 9

#define CV_EVENT_FLAG_LBUTTON 1
#define CV_EVENT_FLAG_RBUTTON 2
#define CV_EVENT_FLAG_MBUTTON 4
#define CV_EVENT_FLAG_CTRLKEY 8
#define CV_EVENT_FLAG_SHIFTKEY 16
#define CV_EVENT_FLAG_ALTKEY 32

Some Utilities

MATH

```
CV_INLINE  int  cvRound( double value );
CV_INLINE  int  cvFloor( double value );
CV_INLINE  int  cvCeil( double value );
#define cvInvSqrt(value) ((float)(1./sqrt(value)));
#define cvSqrt(value)  ((float)sqrt(value));
CV_INLINE int cvIsNaN( double value );
CV_INLINE int cvIsInf( double value )
```

RANDOM NUMBERS

```
typedef uint64 CvRNG;
CV_INLINE CvRNG cvRNG( int64 seed CV_DEFAULT(-1));
CV_INLINE unsigned cvRandInt( CvRNG* rng ); //0-1
CV_INLINE double cvRandReal( CvRNG* rng ); //0-1
```

ARITH METIC AND LOGIC

```
#define  CV_NOP(a)     (a)
#define  CV_ADD(a, b)   ((a) + (b))
#define  CV_SUB(a, b)   ((a) - (b))
#define  CV_MUL(a, b)   ((a) * (b))
#define  CV_AND(a, b)   ((a) & (b))
#define  CV_OR(a, b)    ((a) | (b))
#define  CV_XOR(a, b)   ((a) ^ (b))
#define  CV_ANDN(a, b)  (~(a) & (b))
#define  CV_ORN(a, b)   (~(a) | (b))
#define  CV_SQR(a)     ((a) * (a))

#define  CV_LT(a, b)    ((a) < (b))
#define  CV_LE(a, b)    ((a) <= (b))
#define  CV_EQ(a, b)    ((a) == (b))
#define  CV_NE(a, b)    ((a) != (b))
#define  CV_GT(a, b)    ((a) > (b))
#define  CV_GE(a, b)    ((a) >= (b))

#define  CV_NONZERO(a)     ((a) != 0)
#define  CV_NONZERO_FLT(a)  (((a)+(a)) != 0)
```

Image Data Structures IPL

Basic image structure inherited from old IPL library

Useful functions:

```
IplImage* cvCreateImage( vSize size, int depth,
                         int channels );
void cvReleaseImage( IplImage** image );

IplImage* cvCloneImage( const IplImage* image );

void cvCopy( const CvArr* src, CvArr* dst,
             const CvArr* mask=NULL );

void cvSetImageROI( IplImage* image, CvRect rect );
void cvResetImageROI( IplImage* image );
IplROI  cvRectToROI( CvRect rect, int coi );
```

IPL image header. **Only important fields are shown in blue**

```
typedef struct _IplImage
{
   int  nSize;       /* sizeof(IplImage) */
   int  ID;         /* version (=0)*/
   int  nChannels;    /* Most of OpenCV functions support 1,2,3 or 4 channels */
   int  alphaChannel;  /* ignored by OpenCV */
   int  depth;       /* pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16U,
                  IPL_DEPTH_16S, IPL_DEPTH_32S, IPL_DEPTH_32F and IPL_DEPTH_64F */
   char colorModel[4]; /* ignored by OpenCV */
   char channelSeq[4]; /* ditto */
   int  dataOrder;    /* 0 - interleaved color channels, 1 - separate color channels.
                         cvCreateImage can only create interleaved images */
   int  origin;      /* 0 - top-left origin,
                  1 - bottom-left origin (Windows bitmaps style) */
   int  align;       /* Alignment of image rows (4 or 8).
                  OpenCV ignores it and uses widthStep instead */
   int  width;       /* image width in pixels */
   int  height;      /* image height in pixels */
   struct _IplROI *roi;/* image ROI. when it is not NULL, this specifies image region to process */
   struct _IplImage *maskROI;  /* must be NULL in OpenCV */
   void  *imageId;                  /* ditto */
   struct _IplTileInfo *tileInfo;   /* ditto */
   int  imageSize;    /* image data size in bytes */
   char *imageData;  /* pointer to aligned image data */
   int  widthStep;   /* size of aligned image row in bytes */
   int  BorderMode[4]; /* border completion mode, ignored by OpenCV */
   int  BorderConst[4]; /* ditto */
   char *imageDataOrigin; /* pointer to a very origin of image data
                     (not necessarily aligned) -
                     it is needed for correct image deallocation */
} IplImage;
```

Image Data Structures Matrix

Useful Functions:

```
CvMat* cvCreateMat( int rows, int cols, int type );

CvMat* cvCreateMat( int rows, int cols, int type );

CvMat* cvInitMatHeader(
        CvMat* mat, int rows, int cols, int type,
        void* data=NULL, int step=CV_AUTOSTEP );
Lighter weight:
CvMat cvMat( int rows, int cols,
                int type, void* data=NULL );
CvMat cvMat( int rows, int cols,
                int type, void* data=NULL );

CvMat* cvCreateMatHeader( int rows, int cols,
                                  int type );
void cvCreateData( CvArr* arr );
void cvReleaseData( CvArr* arr );
```

typedef struct CvMat

```
{
   int type; /* CvMat signature (CV_MAT_MAGIC_VAL), element type and flags */
   int step; /* full row length in bytes */
   int* refcount; /* underlying data reference counter */
   union
   {
      uchar* ptr;
      short* s;
      int* i;
      float* fl;
      double* db;
   } data; /* data pointers */
#ifdef __cplusplus
   union
   {
      int rows;
      int height;
   };
   union
   {
      int cols;
      int width;
   };
#else
   int rows; /* number of rows */
   int cols; /* number of columns */
#endif
```

} **CvMat**;

Matric or Image

- Most functions can take either, this is denoted by
- CvArr

  Can be IplImage*, CvMat* or even CvSeq*. The particular array

  // type is determined at runtime by analyzing the first 4

  // bytes of the header of the actual array.

OpenCV Tour

- Installing OpenCV
- Useful structures and conventions
- I/O
- Function Overview
- Notable functions
  - ❖ Machine Learning

I/O Windows for Display

 # Create and position a window:

```
cvNamedWindow("win1", CV_WINDOW_AUTOSIZE);
cvMoveWindow("win1", 100, 100); // offset from the UL corner of the screen
```

 # Load an image:

```
IplImage* img=0;
img=cvLoadImage(fileName);
if(!img) printf("Could not load image file: %s\n",fileName);
```

 # Display an image:

```
cvShowImage("win1",img);
```

Can display a color or grayscale byte/float-image. A byte image is assumed to have values in the range $[0..255]$. A float image is assumed to have values in the range $[0..1]$. A color image is assumed to have data in BGR order.

 # Close a window:

```
cvDestroyWindow("win1");
```

 # Resize a window:

```
cvResizeWindow("win1",100,100); // new width/heigh in pixels
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

I/O Mouse

Handle mouse events:

*** Define a mouse handler:**

```
void mouseHandler(int event, int x, int y, int flags, void* param)
{
  switch(event){
    case CV_EVENT_LBUTTONDOWN:
      if(flags & CV_EVENT_FLAG_CTRLKEY)
        printf("Left button down with CTRL pressed\n");
      break;

    case CV_EVENT_LBUTTONUP:
      printf("Left button up\n");
      break;
  }
}
```

x,y: pixel coordinates are with respect to the UL corner

event:
CV_EVENT_LBUTTONDOWN, CV_EVENT_RBUTTONDOWN,
CV_EVENT_MBUTTONDOWN, CV_EVENT_LBUTTONUP,
CV_EVENT_RBUTTONUP, CV_EVENT_MBUTTONUP,
CV_EVENT_LBUTTONDBLCLK, CV_EVENT_RBUTTONDBLCLK,
CV_EVENT_MBUTTONDBLCLK, CV_EVENT_MOUSEMOVE:

flags:
CV_EVENT_FLAG_CTRLKEY, CV_EVENT_FLAG_SHIFTKEY,
CV_EVENT_FLAG_ALTKEY, CV_EVENT_FLAG_LBUTTON,
CV_EVENT_FLAG_RBUTTON, CV_EVENT_FLAG_MBUTTON

Register the handler:

```
mouseParam=5;
cvSetMouseCallback("win1",mouseHandler,&mouseParam);
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

I/O Keyboard

Handle keyboard events:

*** The keyboard does not have an event handler.**

*** Get keyboard input without blocking:**

```
int key;
key=cvWaitKey(10); // wait 10ms for input
```

*** Get keyboard input with blocking:**

```
int key;
key=cvWaitKey(0); // wait indefinitely for input
```

*** The main keyboard event loop:**

```
while(1){
  key=cvWaitKey(10);
  if(key==27) break;

  switch(key){
    case 'h':
      ...
      break;
    case 'i':
      ...
      break;
  }
}
```

•In Linux, you may have to watch out for higher order bits. If so, use:

```
Int key;
 key = cvWaitKey(10) & 0xFF;
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

I/O Slider or, “Trackbar”

Handle trackbar events:

*** Define a trackbar handler:**

```
void trackbarHandler(int pos)
{
  printf("Trackbar position: %d\n",pos);
}
```

*** Register the handler:**

```
int trackbarVal=25;
int maxVal=100;
cvCreateTrackbar("bar1", "win1", &trackbarVal ,maxVal , trackbarHandler);
```

*** Get the current trackbar position:**

```
int pos = cvGetTrackbarPos("bar1","win1");
```

*** Set the trackbar position:**

```
cvSetTrackbarPos("bar1", "win1", 25);
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

I/O Images Allocating and Release

 # Allocate an image:

IplImage* cvCreateImage(CvSize size, int depth, int channels);

size:

cvSize(width,height);

depth:

pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16U, IPL_DEPTH_16S, IPL_DEPTH_32S, IPL_DEPTH_32F, IPL_DEPTH_64F

channels:

Number of channels per pixel. Can be 1, 2, 3 or 4. The channels are interleaved. The usual data layout of a color image is
b0 g0 r0 b1 g1 r1 ...

Examples:

```
// Allocate a 1-channel byte image
IplImage* img1=cvCreateImage(cvSize(640,480),IPL_DEPTH_8U,1);

// Allocate a 3-channel float image
IplImage* img2=cvCreateImage(cvSize(640,480),IPL_DEPTH_32F,3);
```

 # Release an image:

```
IplImage* img=cvCreateImage(cvSize(640,480),IPL_DEPTH_8U,1);
cvReleaseImage(&img);
```

 # Clone an image:

```
IplImage* img1=cvCreateImage(cvSize(640,480),IPL_DEPTH_8U,1);
IplImage* img2;
img2=cvCloneImage(img1);
```

 # Set/get the region of interest:

```
void  cvSetImageROI(IplImage* image, CvRect rect);
void  cvResetImageROI(IplImage* image);
vRect cvGetImageROI(const IplImage* image);
```

All OpenCV functions support ROI.

 # Set/get the channel of interest:

```
void cvSetImageCOI(IplImage* image, int coi); // 0=all
int cvGetImageCOI(const IplImage* image);
```

The majority of OpenCV functions do NOT support COI.

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

Drawing into an Image

 # Draw a box:
// draw a box with red lines of width 1 between (100,100) and (200,200)
cvRectangle(img, cvPoint(100,100), cvPoint(200,200), cvScalar(255,0,0), 1);

 # Draw a circle:
// draw a circle at (100,100) with a radius of 20. Use green lines of width 1
cvCircle(img, cvPoint(100,100), 20, cvScalar(0,255,0), 1);

 # Draw a line segment:
// draw a green line of width 1 between (100,100) and (200,200)
cvLine(img, cvPoint(100,100), cvPoint(200,200), cvScalar(0,255,0), 1);

 # Draw a set of polylines:

```
CvPoint  curve1[]={10,10,  10,100,  100,100,  100,10};
CvPoint  curve2[]={30,30,  30,130,  130,130,  130,30,  150,10};
CvPoint* curveArr[2]={curve1, curve2};
int      nCurvePts[2]={4,5};
int      nCurves=2;
int      isCurveClosed=1;
int      lineWidth=1;

cvPolyLine(img,curveArr,nCurvePts,nCurves,isCurveClosed,cvScalar(0,255,255),lineWidth);
```

 # Draw a set of filled polygons:

```
cvFillPoly(img,curveArr,nCurvePts,nCurves,cvScalar(0,255,255));
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

Writing Text into an Image

Add text:

```
CvFont font;
double hScale=1.0;
double vScale=1.0;
int    lineWidth=1;
cvInitFont(&font,CV_FONT_HERSHEY_SIMPLEX|CV_FONT_ITALIC,
hScale,vScale,0,lineWidth);

cvPutText (img,"My comment",cvPoint(200,400), &font, cvScalar(255,255,0));
```

Other possible fonts:

```
CV_FONT_HERSHEY_SIMPLEX, CV_FONT_HERSHEY_PLAIN,
CV_FONT_HERSHEY_DUPLEX, CV_FONT_HERSHEY_COMPLEX,
CV_FONT_HERSHEY_TRIPLEX, CV_FONT_HERSHEY_COMPLEX_SMALL,
CV_FONT_HERSHEY_SCRIPT_SIMPLEX, CV_FONT_HERSHEY_SCRIPT_COMPLEX,
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

I/O Reading and Writing Images

 # Reading an image from a file:

```
IplImage* img=0;
img=cvLoadImage(fileName);
if(!img) printf("Could not load image file: %s\n",fileName);
```

Supported image formats:

BMP, DIB, JPEG, JPG, JPE, PNG, PBM, PGM, PPM, SR, RAS, TIFF, TIF

By default, the loaded image is forced to be a 3-channel color image. This default can be modified by using:

```
img=cvLoadImage(fileName,flag);
```

flag: >0 the loaded image is forced to be a 3-channel color image
=0 the loaded image is forced to be a 1 channel grayscale image
<0 the loaded image is loaded as is (with number of channels in the file).

 # Writing an image to a file:

```
if(!cvSaveImage(outFileName,img)) printf("Could not save: %s\n",outFileName);
```

The output file format is determined based on the file name extension.

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

I/O Image Conversion

Image conversion

*** Convert to a grayscale or color byte-image**:

```
cvConvertImage(src, dst, flags=0);
```

src = float/byte grayscale/color image
dst = byte grayscale/color image
flags = CV_CVTIMG_FLIP (flip vertically)
CV_CVTIMG_SWAP_RB (swap the R and B channels)

*** Convert a color image to grayscale:**

Using the OpenCV conversion:

```
cvCvtColor(cimg,gimg,CV_BGR2GRAY); // cimg -> gimg
```

Using a direct conversion:

```
for(i=0;i<cimg->height;i++) for(j=0;j<cimg->width;j++)
  gimgA[i][j]= (uchar)(cimgA[i][j].b*0.114 +
                       cimgA[i][j].g*0.587 +
                       cimgA[i][j].r*0.299);
```

*** Convert between color spaces:**

```
cvCvtColor(src,dst,code); // src -> dst
```

code = CV_<X>2<Y>
<X>/<Y> = RGB, BGR, GRAY, HSV, YCrCb, XYZ, Lab, Luv, HLS

e.g.: CV_BGR2GRAY, CV_BGR2HSV, CV_BGR2Lab

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

I/O Accessing Image Elements

 # Assume that you need to access the k-th channel of the pixel at the i-row and j-th column.
The row index i is in the range $[0,\mbox{height}-1]$. The column index j is in the range $[0,\mbox{width}-1]$. The channel index k is in the range $[0,\mbox{nchannels}-1]$.

 # Indirect access: (General, but inefficient, access to any type image)

*** For a single-channel byte image:**

```
IplImage* img=cvCreateImage(cvSize(640,480),IPL_DEPTH_8U,1);
CvScalar s;
s=cvGet2D(img,i,j); // get the (i,j) pixel value
printf("intensity=%f\n",s.val[0]);
s.val[0]=111;
cvSet2D(img,i,j,s); // set the (i,j) pixel value
```

*** For a multi-channel float (or byte) image:**

```
IplImage* img=cvCreateImage(cvSize(640,480),IPL_DEPTH_32F,3);
CvScalar s;
s=cvGet2D(img,i,j); // get the (i,j) pixel value
printf("B=%f, G=%f, R=%f\n",s.val[0],s.val[1],s.val[2]);
s.val[0]=111;
s.val[1]=111;
s.val[2]=111;
cvSet2D(img,i,j,s); // set the (i,j) pixel value
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

I/O Accessing Image Elements

Direct access: (Efficient access, but error prone)

*** For a single-channel byte image:**

```
IplImage* img=cvCreateImage(cvSize(640,480),IPL_DEPTH_8U,1);
((uchar *)(img->imageData + i*img->widthStep))[j]=111;
```

*** For a multi-channel byte image:**

```
IplImage* img=cvCreateImage(cvSize(640,480),IPL_DEPTH_8U,3);
((uchar *)(img->imageData + i*img->widthStep))[j*img->nChannels + 0]=111; // B
((uchar *)(img->imageData + i*img->widthStep))[j*img->nChannels + 1]=112; // G
((uchar *)(img->imageData + i*img->widthStep))[j*img->nChannels + 2]=113; // R
```

*** For a multi-channel float image:**

```
IplImage* img=cvCreateImage(cvSize(640,480),IPL_DEPTH_32F,3);
((float *)(img->imageData + i*img->widthStep))[j*img->nChannels + 0]=111; // B
((float *)(img->imageData + i*img->widthStep))[j*img->nChannels + 1]=112; // G
((float *)(img->imageData + i*img->widthStep))[j*img->nChannels + 2]=113; // R
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

I/O Accessing Image Elements

 # Direct access using a pointer: (Simplified and efficient access under limiting assumptions)

*** For a single-channel byte image:**

```
IplImage* img  = cvCreateImage(cvSize(640,480),IPL_DEPTH_8U,1);
int height     = img->height;
int width      = img->width;
int step       = img->widthStep/sizeof(uchar);
uchar* data    = (uchar *)img->imageData;
data[i*step+j] = 111;
```

*** For a multi-channel byte image:**

```
IplImage* img  = cvCreateImage(cvSize(640,480),IPL_DEPTH_8U,3);
int height     = img->height;
int width      = img->width;
int step       = img->widthStep/sizeof(uchar);
int channels   = img->nChannels;
uchar* data    = (uchar *)img->imageData;
data[i*step+j*channels+k] = 111;
```

*** For a multi-channel float image (assuming a 4-byte alignment):**

```
IplImage* img  = cvCreateImage(cvSize(640,480),IPL_DEPTH_32F,3);
int height     = img->height;
int width      = img->width;
int step       = img->widthStep/sizeof(float);
int channels   = img->nChannels;
float * data   = (float *)img->imageData;
data[i*step+j*channels+k] = 111;
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

I/O Array Allocation and Release

Allocating and releasing matrices

*** General:**

o OpenCV has a C interface to matrix operations. There are many alternatives that have a C++ interface (which is more convenient) and are as efficient as OpenCV.

o Vectors are obtained in OpenCV as matrices having one of their dimensions as 1.

o Matrices are stored row by row where each row has a 4 byte alignment.

*** Allocate a matrix:**

```
CvMat* cvCreateMat(int rows, int cols, int type);
```

type: Type of the matrix elements. Specified in form CV_<bit_depth>(S|U|F)C<number_of_channels>. E.g.: CV_8UC1 means an 8-bit unsigned single-channel matrix, CV_32SC2 means a 32-bit signed matrix with two channels.

Example:

```
CvMat* M = cvCreateMat(4,4,CV_32FC1);
```

*** Release a matrix:**

```
CvMat* M = cvCreateMat(4,4,CV_32FC1);
cvReleaseMat(&M);
```

*** Clone a matrix:**

```
CvMat* M1 = cvCreateMat(4,4,CV_32FC1);
CvMat* M2;
M2=cvCloneMat(M1);
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

I/O Array Allocation and Release

*** Initialize a matrix:**

```
double a[] = { 1,  2,  3,  4,
               5,  6,  7,  8,
               9, 10, 11, 12 };

CvMat Ma=cvMat(3, 4, CV_64FC1, a);
```

Alternatively:

```
CvMat Ma;
cvInitMatHeader(&Ma, 3, 4, CV_64FC1, a);
```

*** Initialize a matrix to identity:**

```
CvMat* M = cvCreateMat(4,4,CV_32FC1);
cvSetIdentity(M);
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

I/O Accessing Array Elements

Accessing matrix elements

*** Assume that you need to access the (i,j) cell of a 2D float matrix.**
*** Indirect matrix element access:**

```
cvmSet(M,i,j,2.0); // Set M(i,j)
t = cvmGet(M,i,j); // Get M(i,j)
```

*** Direct matrix element access assuming a 4-byte alignment:**

```
CvMat* M    = cvCreateMat(4,4,CV_32FC1);
int n       = M->cols;
float *data = M->data.fl;

data[i*n+j] = 3.0;
```

*** Direct matrix element access assuming possible alignment gaps:**

```
CvMat* M    = cvCreateMat(4,4,CV_32FC1);
int   step  = M->step/sizeof(float);
float *data = M->data.fl;

(data+i*step)[j] = 3.0;
```

*** Direct matrix element access of an initialized matrix:**

```
double a[16];
CvMat Ma = cvMat(3, 4, CV_64FC1, a);
a[i*4+j] = 2.0; // Ma(i,j)=2.0;
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

Working with Video

Capturing a frame from a video sequence

*** OpenCV supports capturing images from a camera or a video file (AVI).**

*** Initializing capture from a camera:**

```
CvCapture* capture = cvCaptureFromCAM(0); // capture from video device #0
```

*** Initializing capture from a file:**

```
CvCapture* capture = cvCaptureFromAVI("infile.avi");
```

*** Capturing a frame:**

```
IplImage* img = 0;
if(!cvGrabFrame(capture)){          // capture a frame
  printf("Could not grab a frame\n\7");
  exit(0);
}
img=cvRetrieveFrame(capture);       // retrieve the captured frame
```

To obtain images from several cameras simultaneously, first grab an image from each camera. Retrieve the captured images after the grabbing is complete.

*** Releasing the capture source:**

```
cvReleaseCapture(&capture);
```

Note that the image captured by the device is allocated/released by the capture function. There is no need to release it explicitly.

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

Working with Video

Getting/setting frame information

*** Get capture device properties:**

```
cvQueryFrame(capture); // this call is necessary to get correct
                       // capture properties
int frameH    = (int) cvGetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT);
int frameW    = (int) cvGetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH);
int fps       = (int) cvGetCaptureProperty(capture, CV_CAP_PROP_FPS);
int numFrames = (int) cvGetCaptureProperty(capture,  CV_CAP_PROP_FRAME_COUNT);
```

The total frame count is relevant for video files only. It does not seem to be working properly.

*** Get frame information:**

```
float posMsec   =      cvGetCaptureProperty(capture, CV_CAP_PROP_POS_MSEC);
int posFrames   = (int) cvGetCaptureProperty(capture, CV_CAP_PROP_POS_FRAMES);
float posRatio  =      cvGetCaptureProperty(capture, CV_CAP_PROP_POS_AVI_RATIO);
```

Get the position of the captured frame in [msec] with respect to the first frame, or get its index where the first frame starts with an index of 0. The relative position (ratio) is 0 in the first frame and 1 in the last frame. This ratio is valid only for capturing images from a file.

*** Set the index of the first frame to capture:**

```
// start capturing from a relative position of 0.9 of a video file
cvSetCaptureProperty(capture, CV_CAP_PROP_POS_AVI_RATIO, (double)0.9);
```

This only applies for capturing from a file. It does not seem to be working properly.

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

Working with Video

Saving a video file

*** Initializing a video writer:**

```
CvVideoWriter *writer = 0;
int isColor = 1;
int fps     = 25;  // or 30
int frameW  = 640; // 744 for firewire cameras
int frameH  = 480; // 480 for firewire cameras
writer=cvCreateVideoWriter("out.avi",CV_FOURCC('P','I','M','1'),
                    fps,cvSize(frameW,frameH),isColor);
```

Other possible codec codes:

```
CV_FOURCC('P','I','M','1')    = MPEG-1 codec
CV_FOURCC('M','J','P','G')    = motion-jpeg codec (does not work well)
CV_FOURCC('M', 'P', '4', '2') = MPEG-4.2 codec
CV_FOURCC('D', 'I', 'V', '3') = MPEG-4.3 codec
CV_FOURCC('D', 'I', 'V', 'X') = MPEG-4 codec
CV_FOURCC('U', '2', '6', '3') = H263 codec
CV_FOURCC('I', '2', '6', '3') = H263I codec
CV_FOURCC('F', 'L', 'V', '1') = FLV1 codec
```

A codec code of -1 will open a codec selection window (in windows).

*** Writing the video file:**

```
IplImage* img = 0;
int nFrames = 50;
for(i=0;i<nFrames;i++){
  cvGrabFrame(capture);        // capture a frame
  img=cvRetrieveFrame(capture);  // retrieve the captured frame
  cvWriteFrame(writer,img);      // add the frame to the file
}
```

To view the captured frames during capture, add the following in the loop:

```
cvShowImage("mainWin", img);
key=cvWaitKey(20);          // wait 20 ms
```

Note that without the 20[msec] delay the captured sequence is not displayed properly.

*** Releasing the video writer:**

```
cvReleaseVideoWriter(&writer);
```

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

OpenCV Tour

- Installing OpenCV
- Useful structures and conventions
- I/O
- Function Overview
- Notable functions
  - ❖ Machine Learning

Matrix and Vector Operations

Matrix/vector operations

*** Matrix-matrix operations:**

```
CvMat *Ma, *Mb, *Mc;
cvAdd(Ma, Mb, Mc);     // Ma+Mb   -> Mc
cvSub(Ma, Mb, Mc);     // Ma-Mb   -> Mc
cvMatMul(Ma, Mb, Mc);  // Ma*Mb   -> Mc
```

*** Elementwise matrix operations:**

```
CvMat *Ma, *Mb, *Mc;
cvMul(Ma, Mb, Mc);     // Ma.*Mb  -> Mc
cvDiv(Ma, Mb, Mc);     // Ma./Mb  -> Mc
cvAddS(Ma, cvScalar(-10.0), Mc); // Ma.-10 -> Mc
```

*** Vector products:**

```
double va[] = {1, 2, 3};
double vb[] = {0, 0, 1};
double vc[3];

CvMat Va=cvMat(3, 1, CV_64FC1, va);
CvMat Vb=cvMat(3, 1, CV_64FC1, vb);
CvMat Vc=cvMat(3, 1, CV_64FC1, vc);

double res=cvDotProduct(&Va,&Vb); // dot product:   Va . Vb -> res
cvCrossProduct(&Va, &Vb, &Vc);    // cross product: Va x Vb -> Vc
```

Note that Va, Vb, Vc, must be 3 element vectors in a cross product.

http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

Matrix and Vector Operations

 # Single matrix operations:

```
CvMat *Ma, *Mb;
cvTranspose(Ma, Mb);      // transpose(Ma) -> Mb (cannot transpose onto self)
CvScalar t = cvTrace(Ma); // trace(Ma) -> t.val[0]
double d = cvDet(Ma);     // det(Ma) -> d
cvInvert(Ma, Mb);         // inv(Ma) -> Mb
```

 # Inhomogeneous linear system solver:

```
CvMat* A  = cvCreateMat(3,3,CV_32FC1);
CvMat* x  = cvCreateMat(3,1,CV_32FC1);
CvMat* b  = cvCreateMat(3,1,CV_32FC1);
cvSolve(&A, &b, &x);    // solve (Ax=b) for x
```

 # Eigen analysis (of a symmetric matrix):

```
CvMat* A  = cvCreateMat(3,3,CV_32FC1);
CvMat* E  = cvCreateMat(3,3,CV_32FC1);
CvMat* l  = cvCreateMat(3,1,CV_32FC1);
cvEigenVV(&A, &E, &l);  // l = eigenvalues of A (descending order)
                        // E = corresponding eigenvectors (rows)
```

 # Singular value decomposition:

```
CvMat* A  = cvCreateMat(3,3,CV_32FC1);
CvMat* U  = cvCreateMat(3,3,CV_32FC1);
CvMat* D  = cvCreateMat(3,3,CV_32FC1);
CvMat* V  = cvCreateMat(3,3,CV_32FC1);
cvSVD(A, D, U, V, CV_SVD_U_T|CV_SVD_V_T); // A = U D V^T
```

The flags cause U and V to be returned transposed (does not work well without the transpose flags).

samples/c

In ...\opencv_incomp\samples\c

bgfg_codebook.cpp	- Use of a image value codebook for background detection for collectin objects
bgfg_segm.cpp	- Use of a background learning engine
blobtrack.cpp	- Engine for blob tracking in images
calibration.cpp	- Camera Calibration
camshiftdemo.c	- Use of meanshift in simple color tracking
contours.c	- Demonstrates how to compute and use object contours
convert_cascade.c	- Change the window size in a recognition cascade
convexhull.c	- Find the convex hull of an object
delaunay.c	- Triangulate a 2D point cloud
demhist.c	- Show how to use histograms for recognition
dft.c	- Discrete fourier transform
distrans.c	- distance map from edges in an image
drawing.c	- Various drawing functions
edge.c	- Edge detection
facedetect.c	- Face detection by classifier cascade
ffilldemo.c	- Flood filling demo
find_obj.cpp	- Demo use of SURF features
fitellipse.c	- Robust elipse fitting
houghlines.c	- Line detection
image.cpp	- Shows use of new image class, CvImage();
inpaint.cpp	- Texture infill to repair imagery
kalman.c	- Kalman filter for trackign
kmeans.c	- K-Means
laplace.c	- Convolve image with laplacian.
letter_recog.cpp	- Example of using machine learning Boosting, Backpropagation (MLP) and Random forests
lkdemo.c	- Lukas-Canada optical flow
minarea.c	- For a cloud of points in 2D, find min bounding box and circle. Shows use of Cv_SEQ
morphology.c	- Demonstrates Erode, Dilate, Open, Close
motempl.c	- Demonstrates motion templates (orthogonal optical flow given silhouettes)
mushroom.cpp	- Demonstrates use of decision trees (CART) for recognition
pyramid_segmentation.c	- Color segmentation in pyramid
squares.c	- Uses contour processing to find squares in an image
stereo_calib.cpp	- Stereo calibration, recognition and disparity map computation
watershed.cpp	- Watershed transform demo.

Book Examples

ch2_ex2_1.cpp	Load image from disk
ch2_ex2_2.cpp	Play video from disk
ch2_ex2_3.cpp	Add a slider control
ch2_ex2_4.cpp	Load, smooth and dsiplay image
ch2_ex2_5.cpp	Pyramid down sampling
ch2_ex2_6.cpp	CvCanny edge detection
ch2_ex2_7.cpp	Pyramid down and Canny edge
ch2_ex2_8.cpp	Above program simplified
ch2_ex2_9.cpp	Play video from camera or file
ch2_ex2_10.cpp	Read and write video, do Logpolar
ch3_ex3_1.txt	Matrix structure
ch3_ex3_2.txt	Matrix creation and release
ch3_ex3_3.cpp	Create matrix from data list
ch3_ex3_4.cpp	Accessing matrix data CV_MAT_ELEM()
ch3_ex3_5.cpp	Setting matrix CV_MAT_ELEM_PTR()
ch3_ex3_6.txt	Pointer access to matrix data
ch3_ex3_7.txt	Image and Matrix Element access functions
ch3_ex3_8.txt	Setting matrix or image elements
ch3_ex3_9.cpp	Summing all elements in 3 channel matrix
ch3_ex3_10.txt	IplImage Header
ch3_ex3_11.cpp	Use of widthstep
ch3_ex3_12.cpp	Use of image ROI
ch3_ex3_13.cpp	Implementing an ROI using widthstep
ch3_ex3_14.cpp	Alpha blending example
ch3_ex3_15.cpp	Saving and loading a CvMat
ch3_ex3_16.txt	File storage demo
ch3_ex3_17.cpp	Writing configuration files as XML
ch3_ex3_19.cpp	Reading an XML file
ch3_ex3_20.cpp	How to check if IPP acceleration is on

Book Examples

ch4_ex4_1.cpp	Use a mouse to draw boxes
ch4_ex4_2.cpp	Use a trackbar as a button
ch4_ex4_3.cpp	Finding the video codec
ch5_ex5_1.cpp	Using CvSeq
ch5_ex5_2.cpp	cvThreshold example
ch5_ex5_3.cpp	Combining image planes
ch5_ex5_4.cpp	Adaptive threshiolding
ch6_ex6_1.cpp	cvHoughCircles example
ch6_ex6_2.cpp	Affine transform
ch6_ex6_3.cpp	Perspective transform
ch6_ex6_4.cpp	Log-Polar conversion
ch6_ex6_5.cpp	2D Fourier Transform
ch7_ex7_1.cpp	Using histograms
ch7_ex7_2.txt	Earth Mover's Distance interface
ch7_ex7_3_expanded.cpp	Earth Mover's Distance set up
ch7_ex7_4.txt	Using Earth Mover's Distance
ch7_ex7_5.cpp	Template matching /Cross Corr.
ch7_ex7_5_HistBackProj.cpp	Back projection of histograms
ch8_ex8_1.txt	CvSeq structure
ch8_ex2.cpp	Contour structure
ch8_ex8_2.cpp	Finding contours
ch8_ex8_3.cpp	Drawing contours

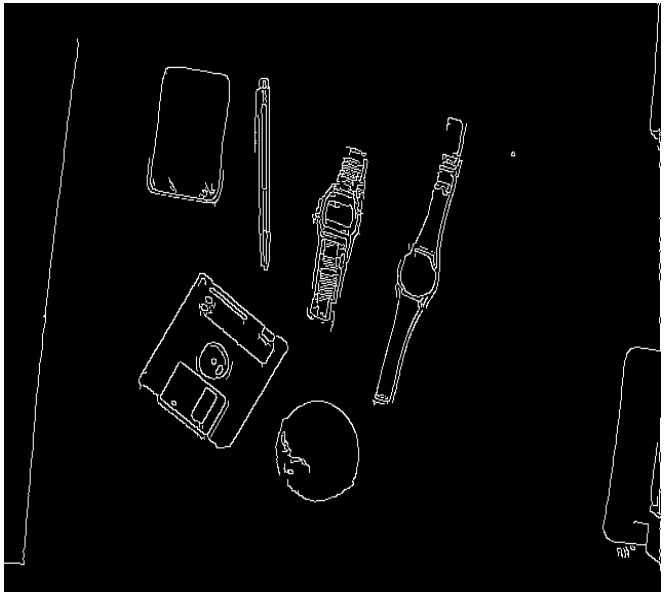
Book Examples

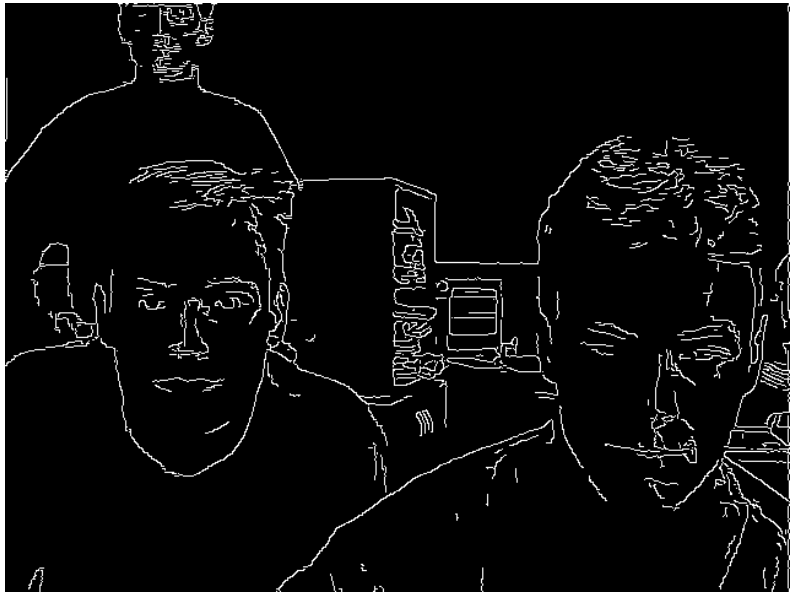
ch9_ex9_1.cpp	Sampling from a line in an image
ch9_watershed.cpp	Image segmentation using Watershed transform
ch9_AvgBackground.cpp	Background model using an average image
ch9_backgroundAVG.cpp	Background averaging using a codebook compared to just an average
ch9_backgroundDiff.cpp	Use the codebook method for doing background differencing
ch9_ClearStaleCB_Entries.cpp	Refine codebook to eliminate stale entries
cv_yuv_codebook.cpp	Core code used to design OpenCV codebook
ch10_ex10_1.cpp	Optical flow using Lucas-Kanade in an image pyramid
ch10_ex10_1b_Horn_Schunck.cpp	Optical flow based on Horn-Schunck block matching
ch10_ex10_2.cpp	Kalman filter example code
ch10_motempl.cpp	Using motion templates for segmenting motion.
ch11_ex11_1.cpp	Camera calibration using automatic chessboard finding using a camera
ch11_ex11_1_fromdisk.cpp	Doing the same, but read from disk
ch11_chessboards.txt	List of included chessboards for calibration from disk example
ch12_ex12_1.cpp	Creating a bird's eye view of a scene using homography
ch12_ex12_2.cpp	Computing the Fundamental matrix using RANSAC
ch12_ex12_3.cpp	Stereo calibration, rectification and correspondence
ch12_ex12_4.cpp	2D robust line fitting
ch12_list.txt	List of included stereo L+R image pair data
ch13_dtree.cpp	Example of using a decision tree
ch13_ex13_1.cpp	Using k-means
ch13_ex13_2.cpp	Creating and training a decision tree
ch13_ex13_3.cpp	Training using statistical boosting
ch13_ex13_4.cpp	Face detection using Viola-Jones
cvx_defs.cpp	Some defines for use with codebook segmentatio

OpenCV Tour

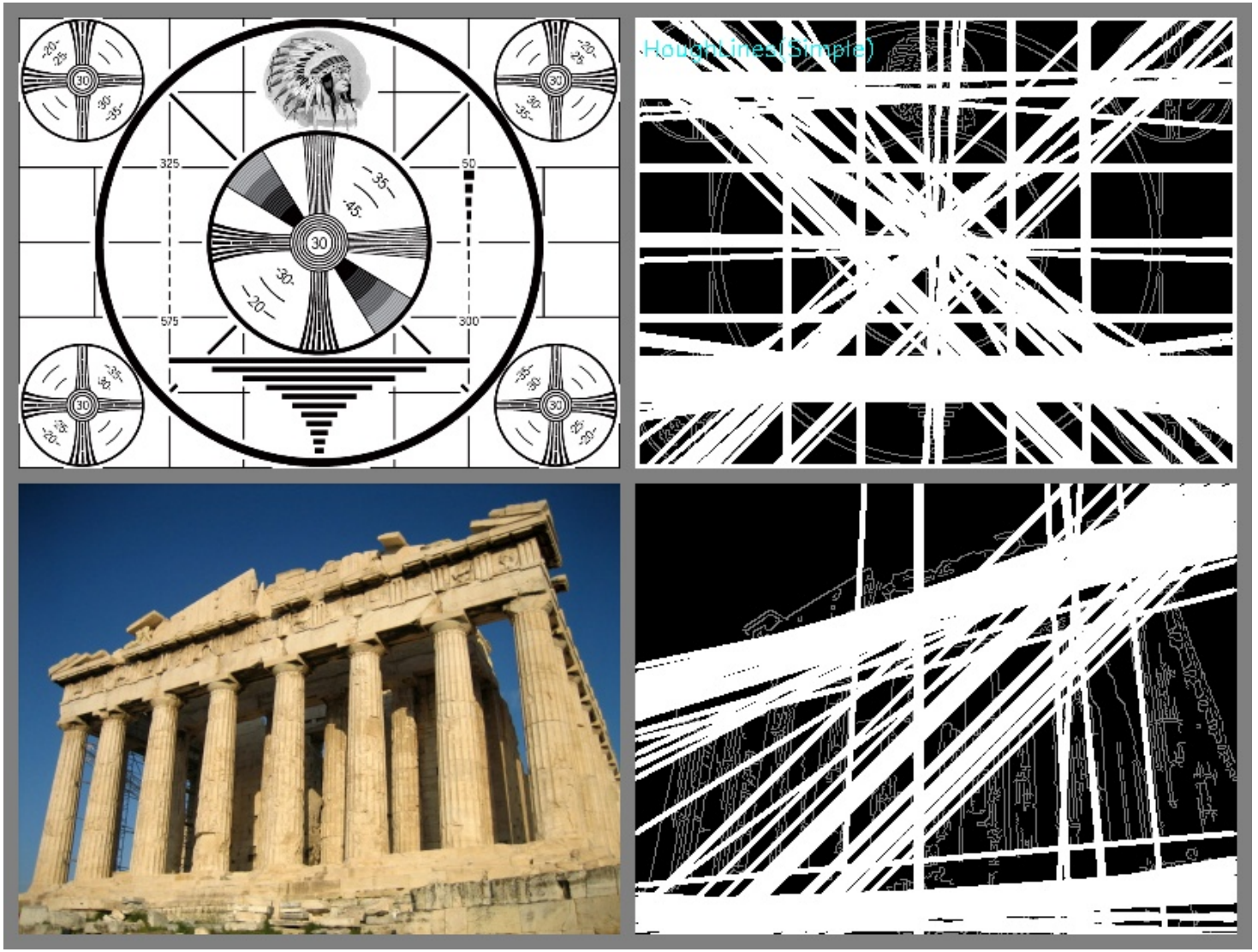
- Installing OpenCV
- Useful structures and conventions
- I/O
- Function Overview
- Notable functions
  - ❖ Machine Learning

Canny Edge Detector

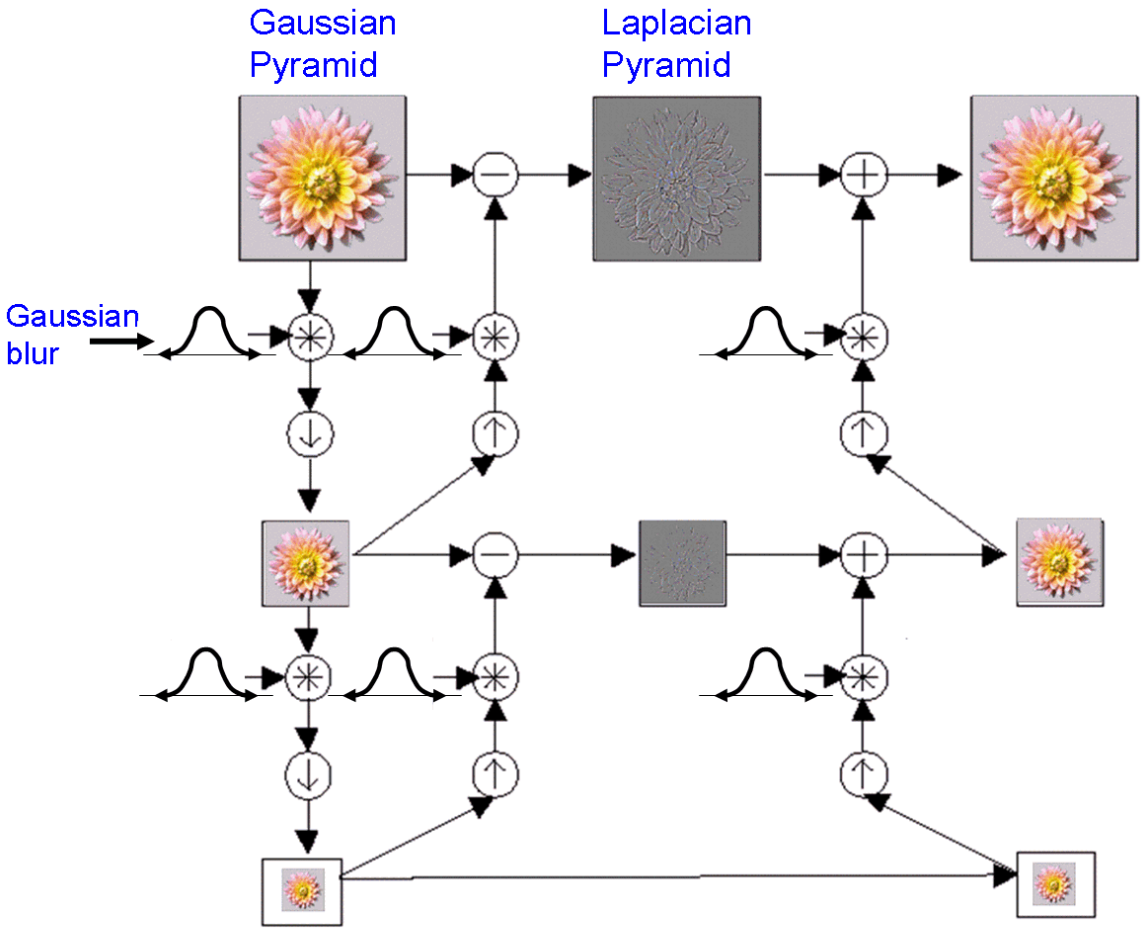




Hough Transform



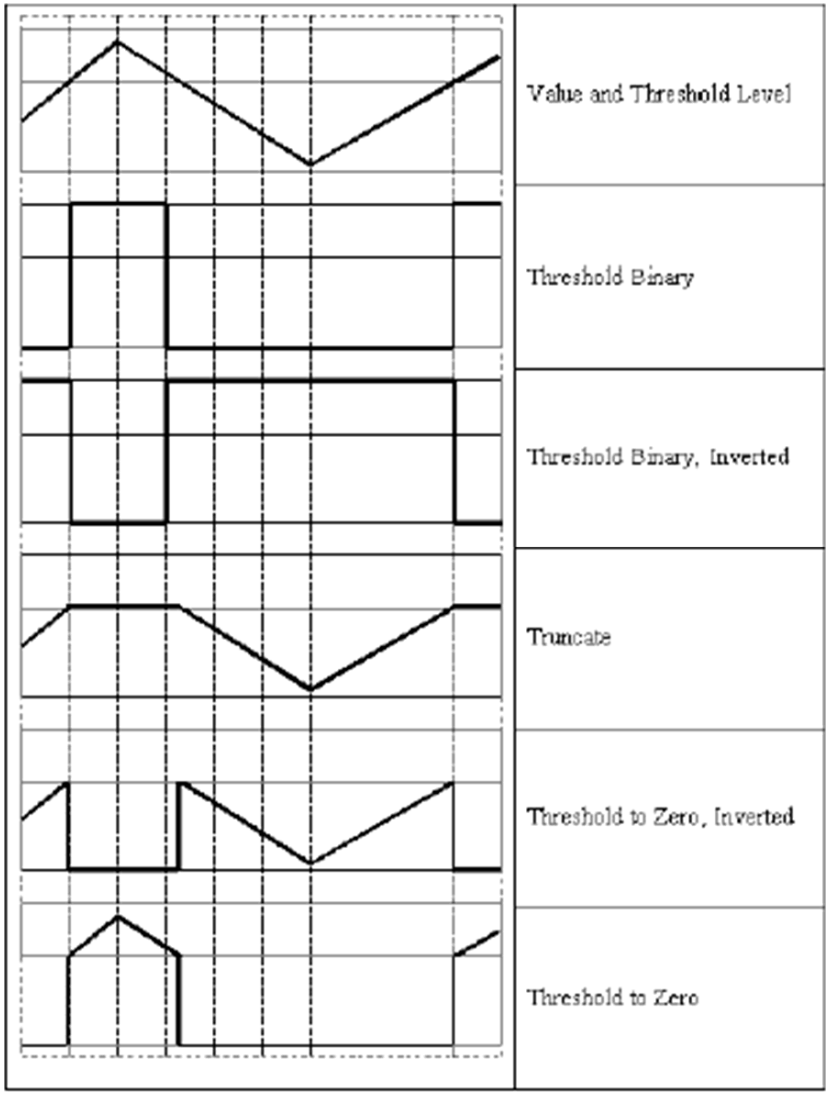
Scale Space



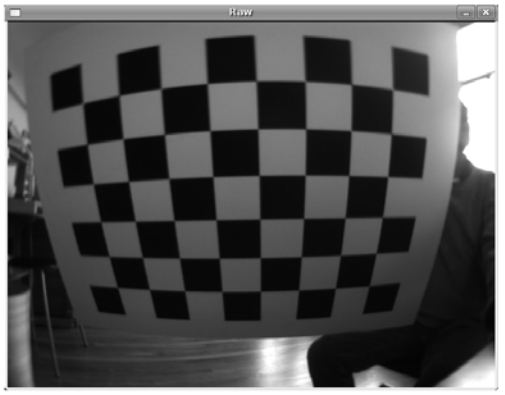

```
void cvPyrDown(
        IplImage*  src,
        IplImage*  dst,
        IplFilter      filter = IPL_GAUSSIAN_5x5);
```

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

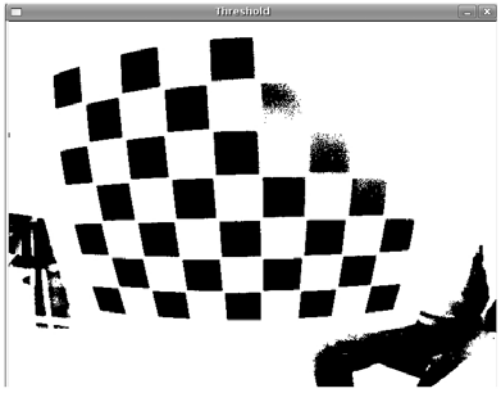
Thresholds



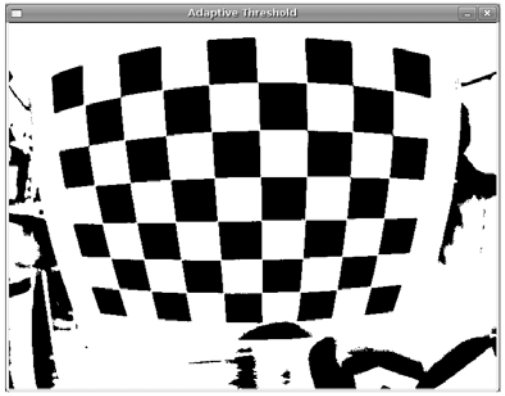
Source Image:



Binary Threshold:



Adaptive Binary Threshold:



Contours

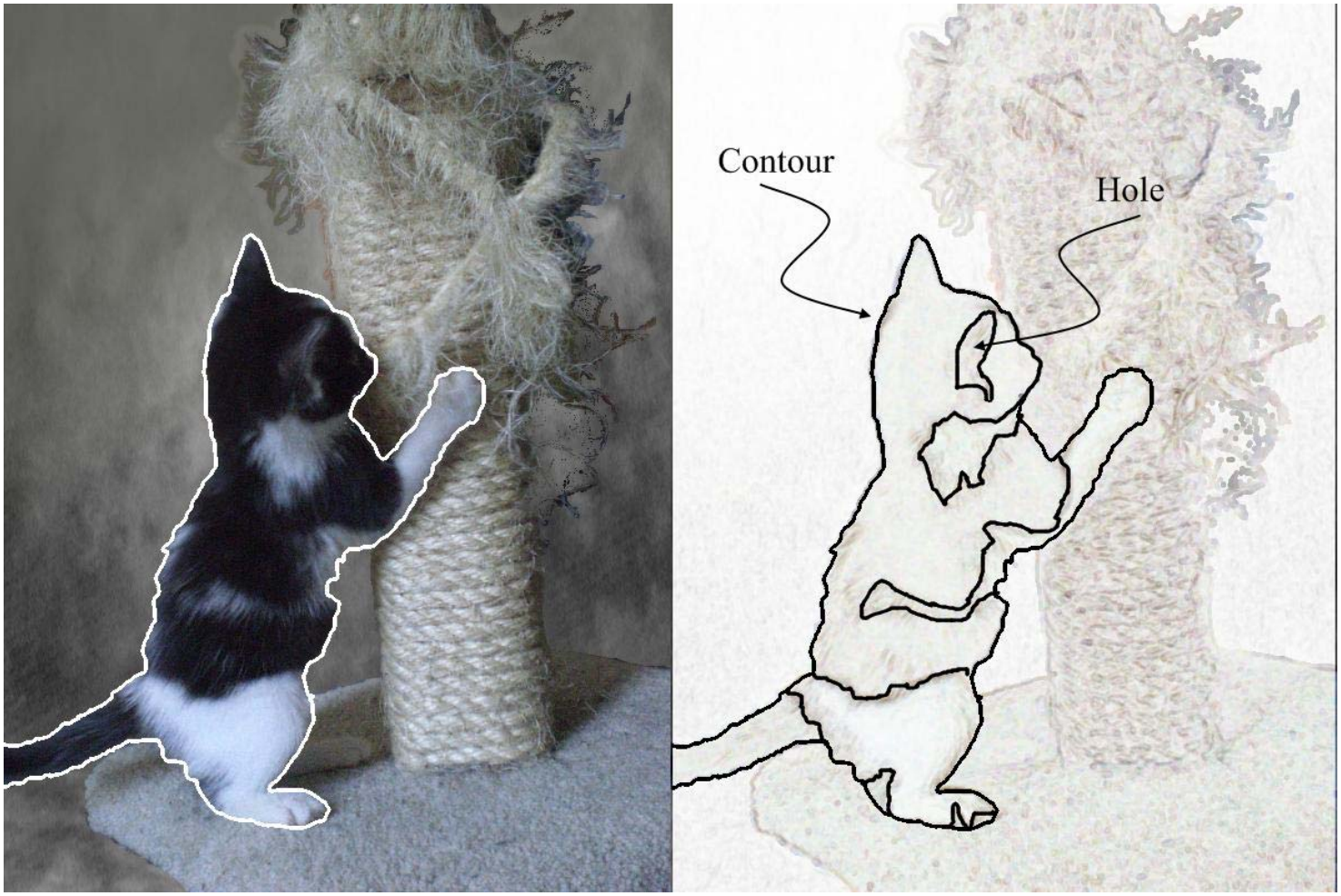


Image textures

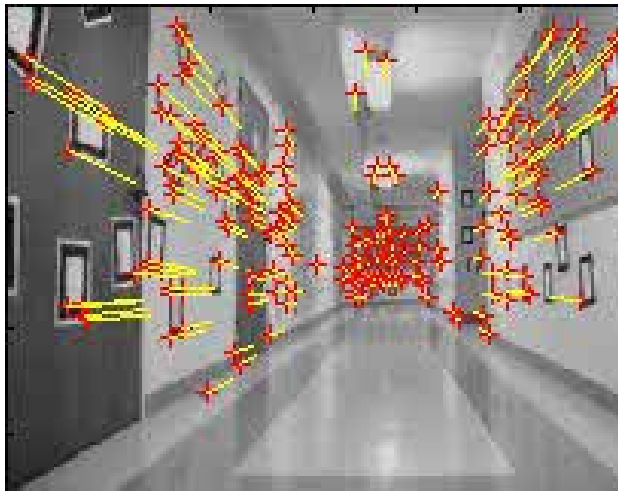
- Inpainting:
- Removes damage to images, in this case, it removes the text.

Optical Flow

```
// 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)

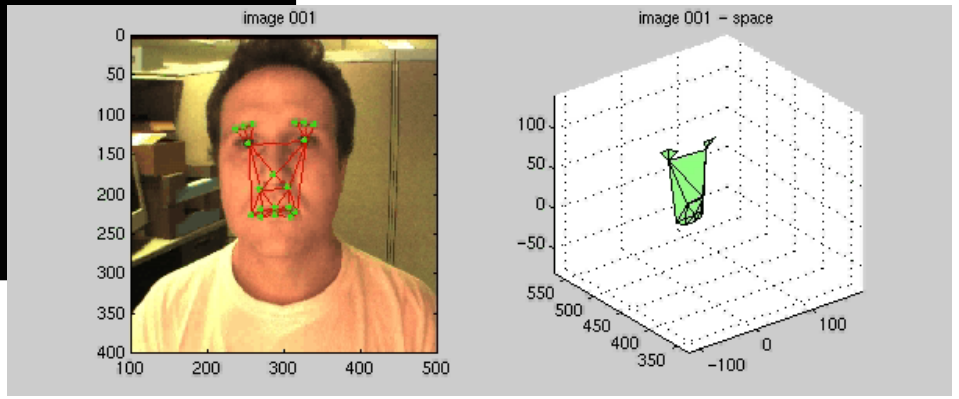


$$I(x+dx, y+dy, t+dt) = I(x, y, t);$$

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

$$G \cdot \partial X = b,$$

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

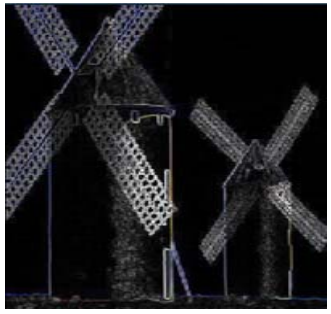


Morphological Operations Examples

- Morphology - applying Min-Max. Filters and its combinations



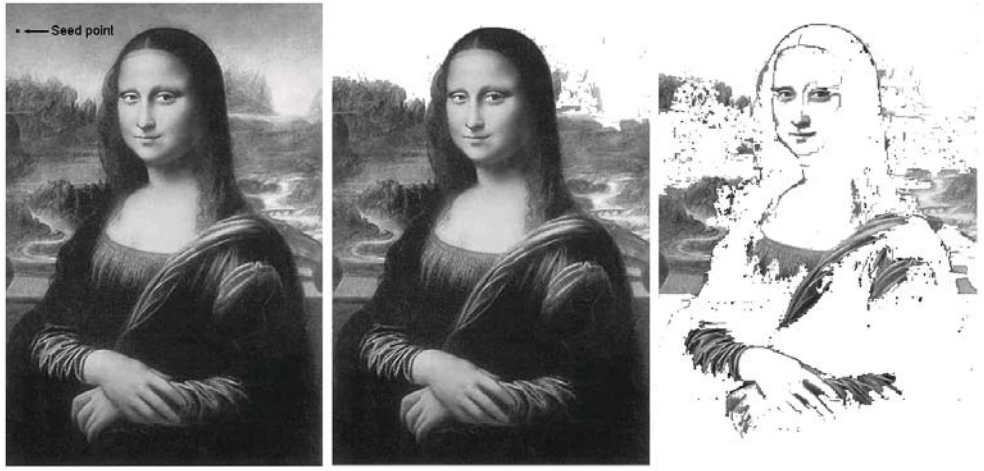


Distance Transform

- Distance field from edges of objects

Flood Filling



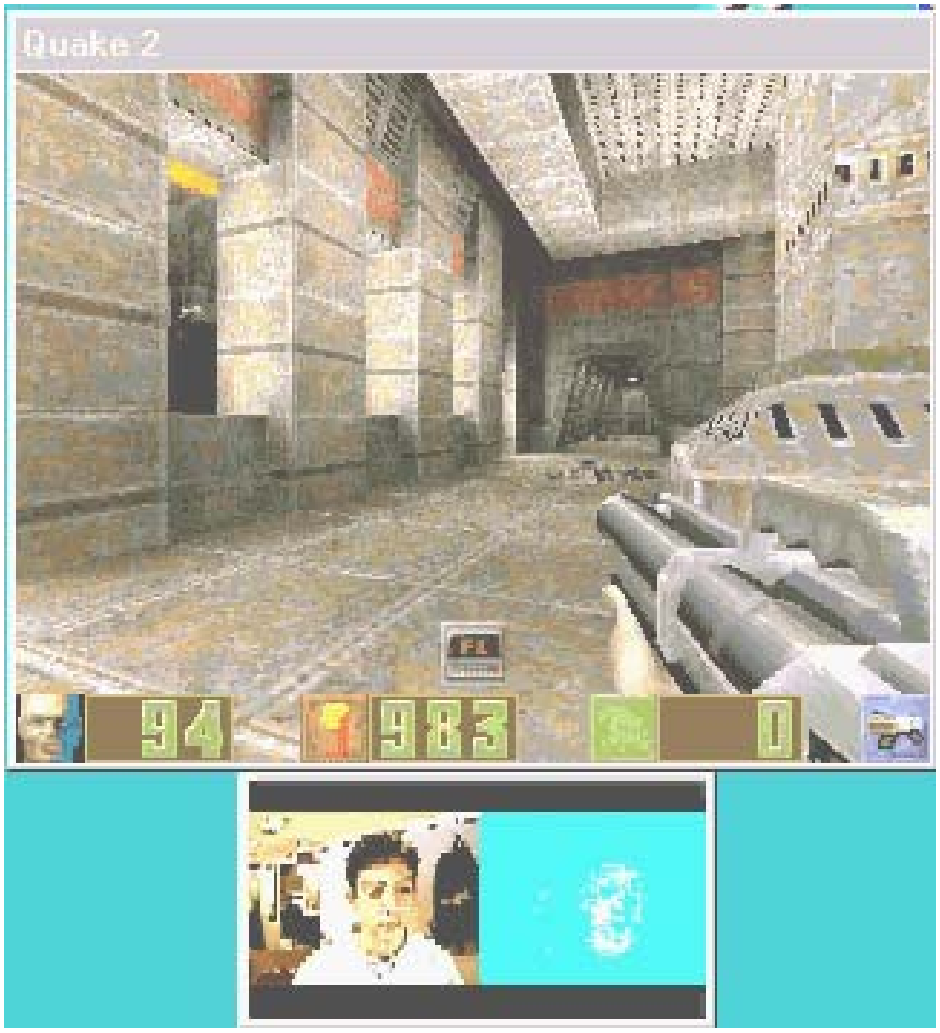
Original image

Tolerance interval ± 5

Tolerance interval ± 6

Tracking with CAMSHIFT

- Control game with head

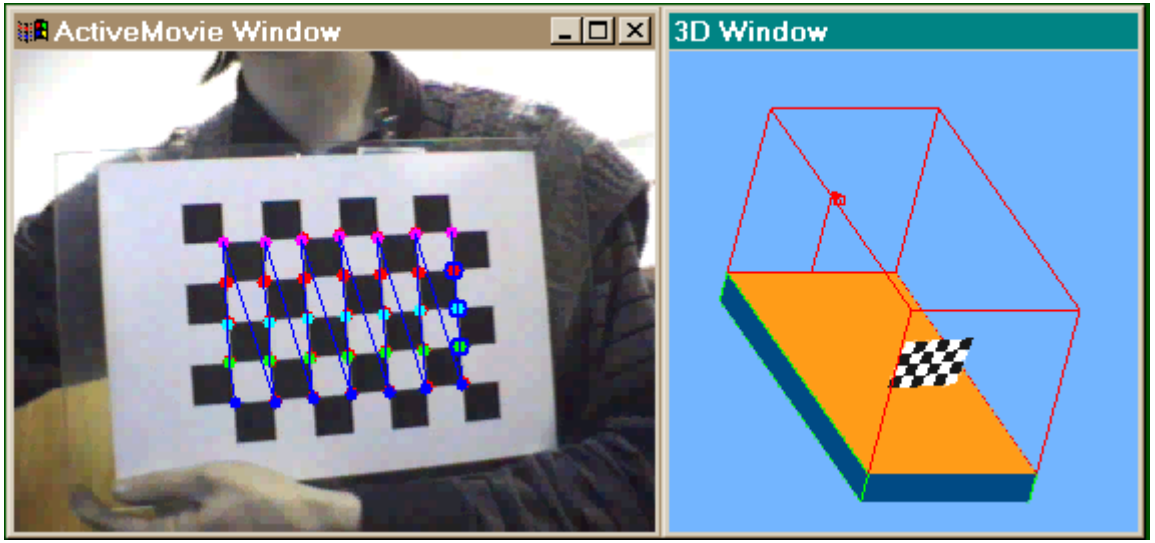


Camera Calibration

Now, camera calibration can be done by holding checkerboard in front of the camera for a few seconds.

And after that you'll get:

3D view of checkerboard



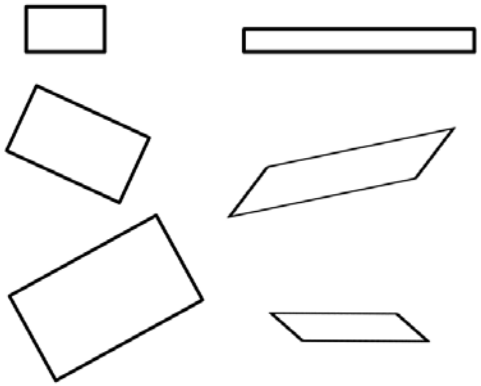
Un-distorted image

(My work with Vadim Pisarevski)

Projections

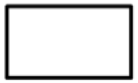
Affine (2x2)

Parallelograms



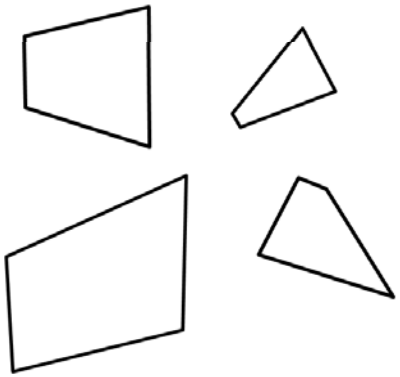
Perspective (3x3)

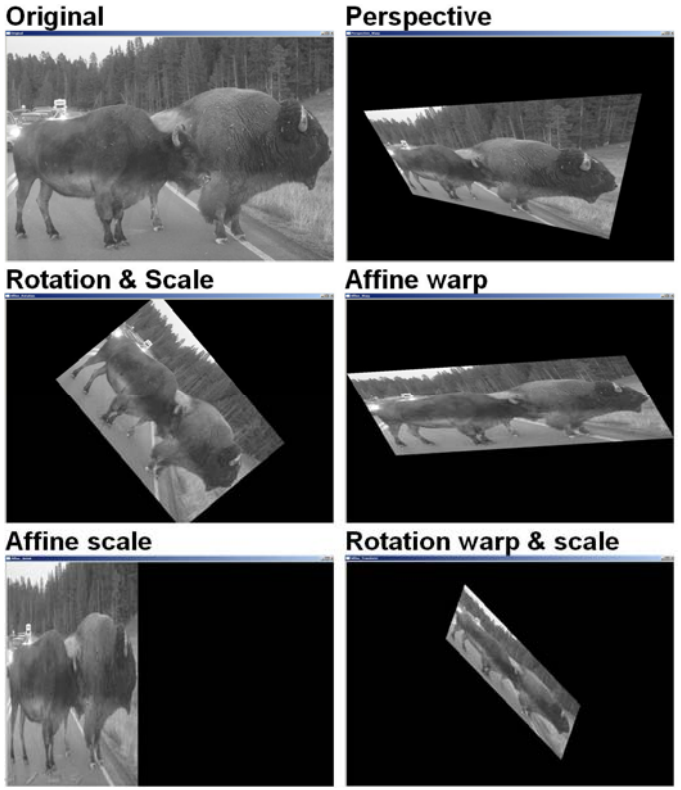
or “Homography”



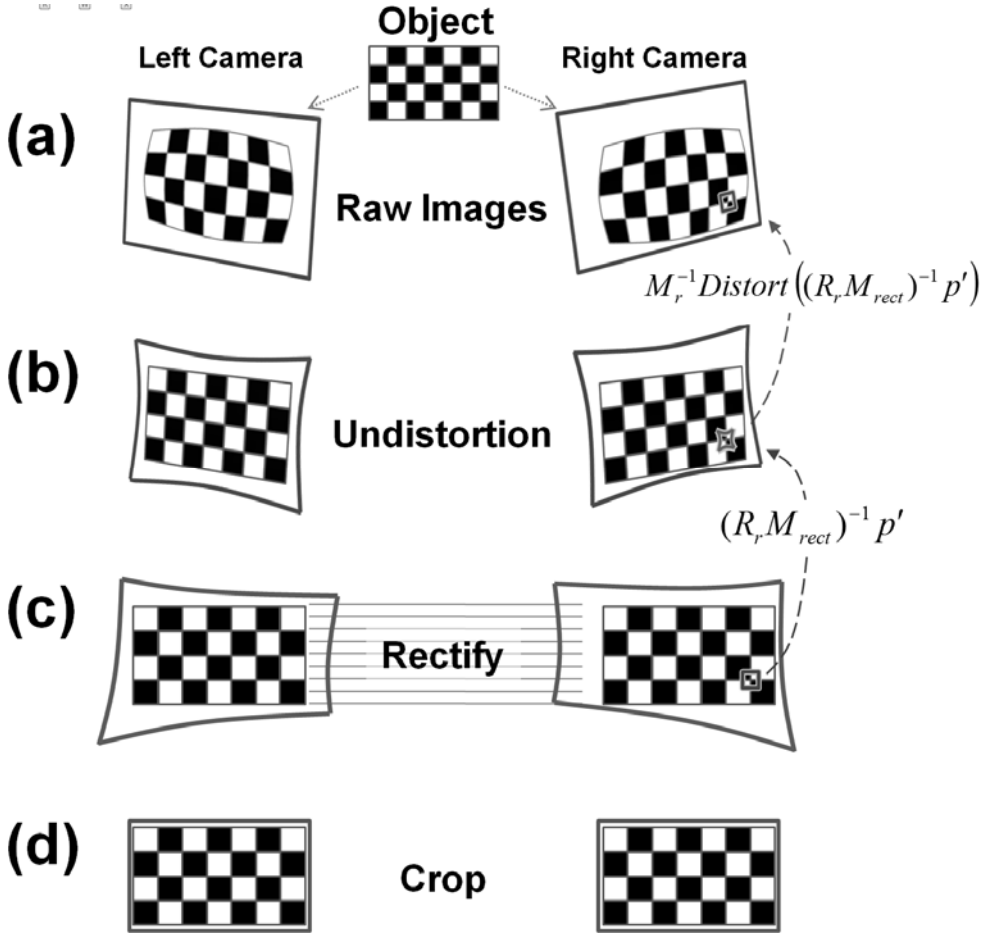
Trapazoids

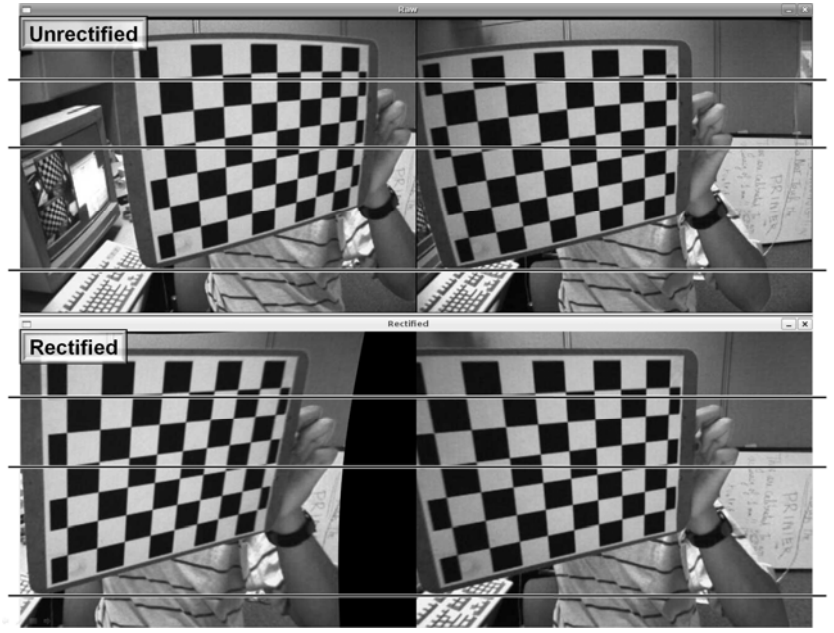
(Includes all of Affine)





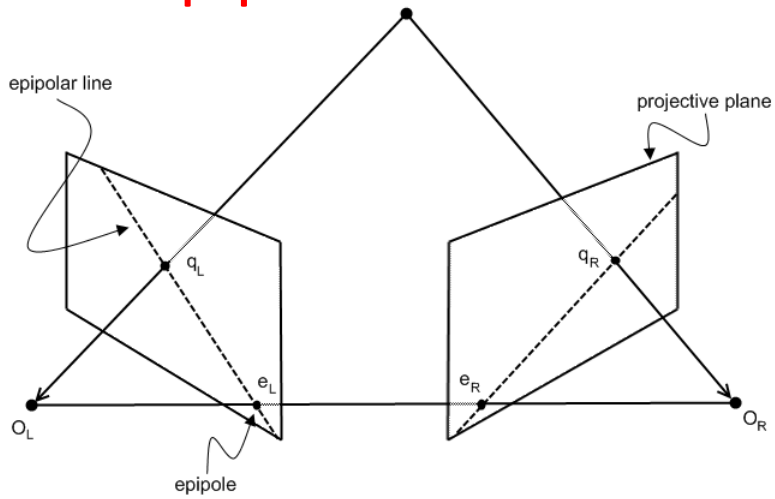
Stereo Calibration



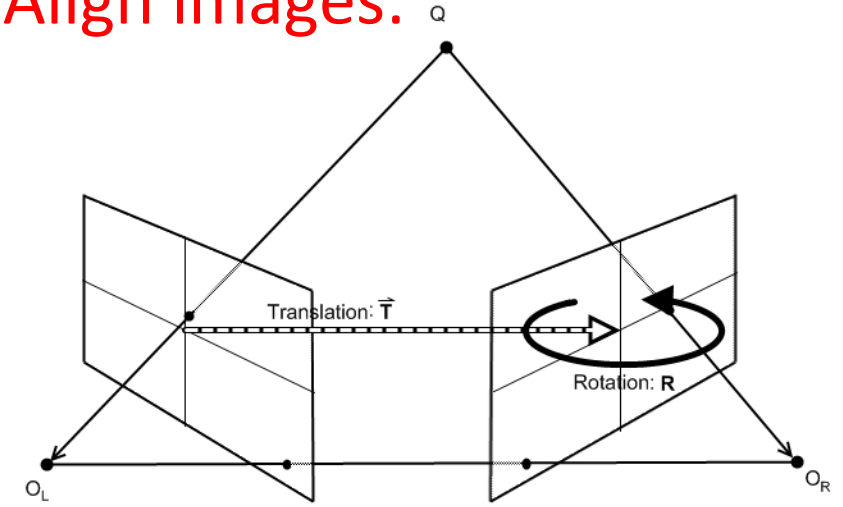


3D Stereo Vision

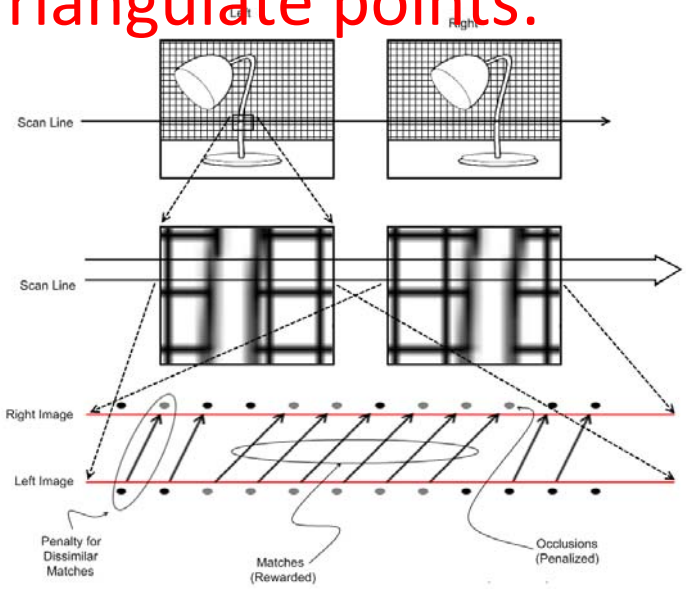
- Find Epipolar lines:



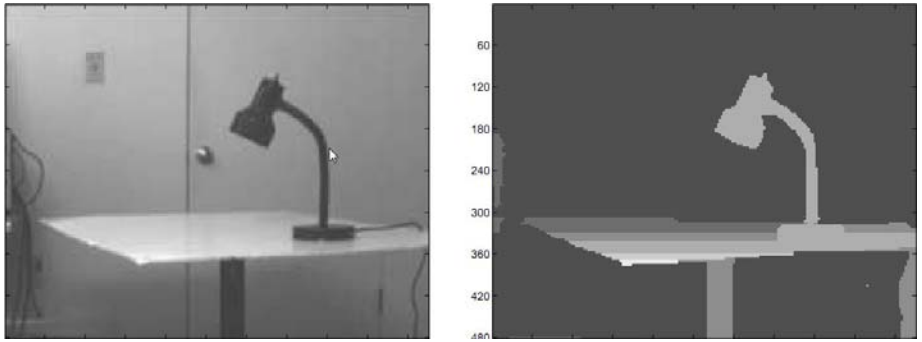
- Align images:



- Triangulate points:

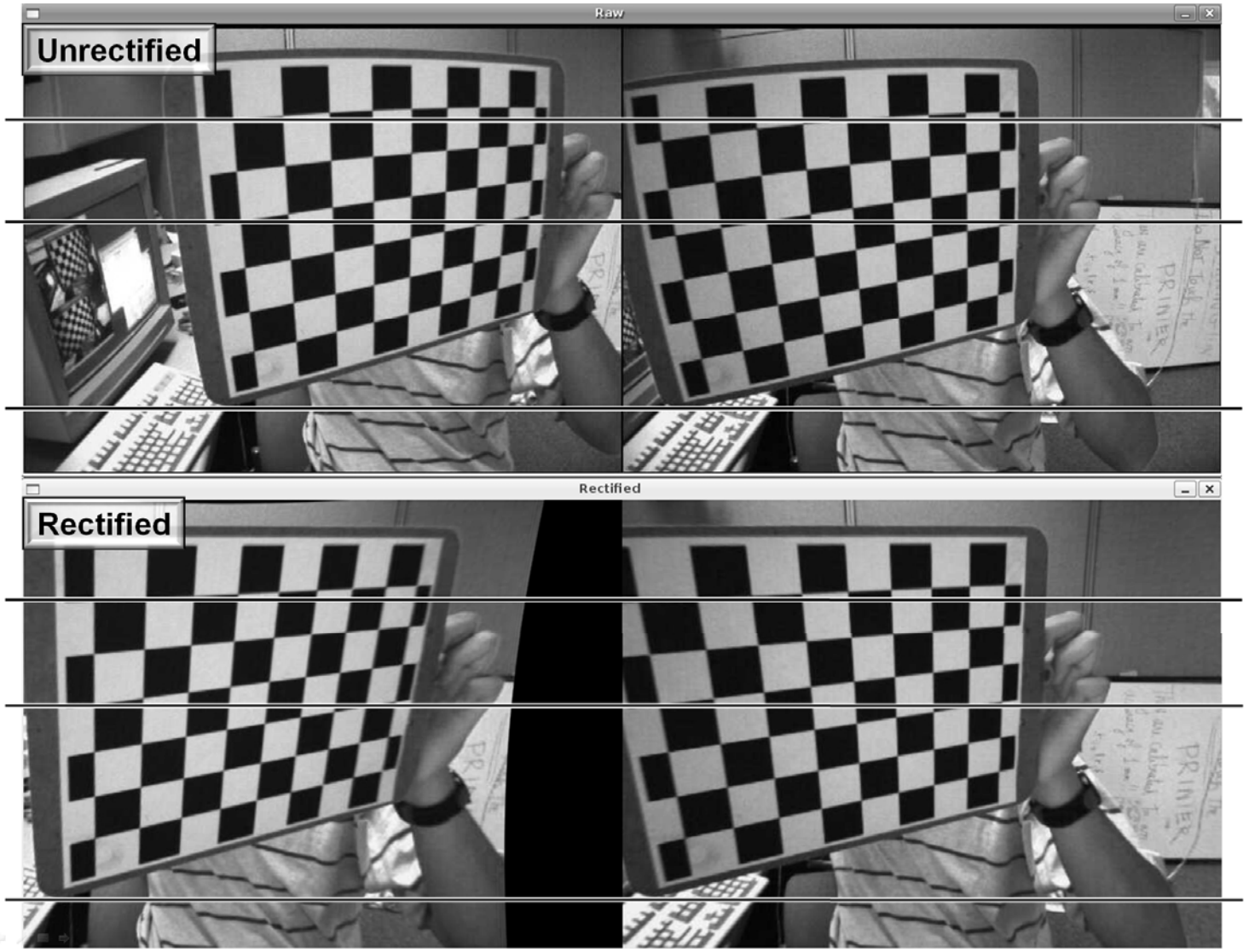


- Depth:



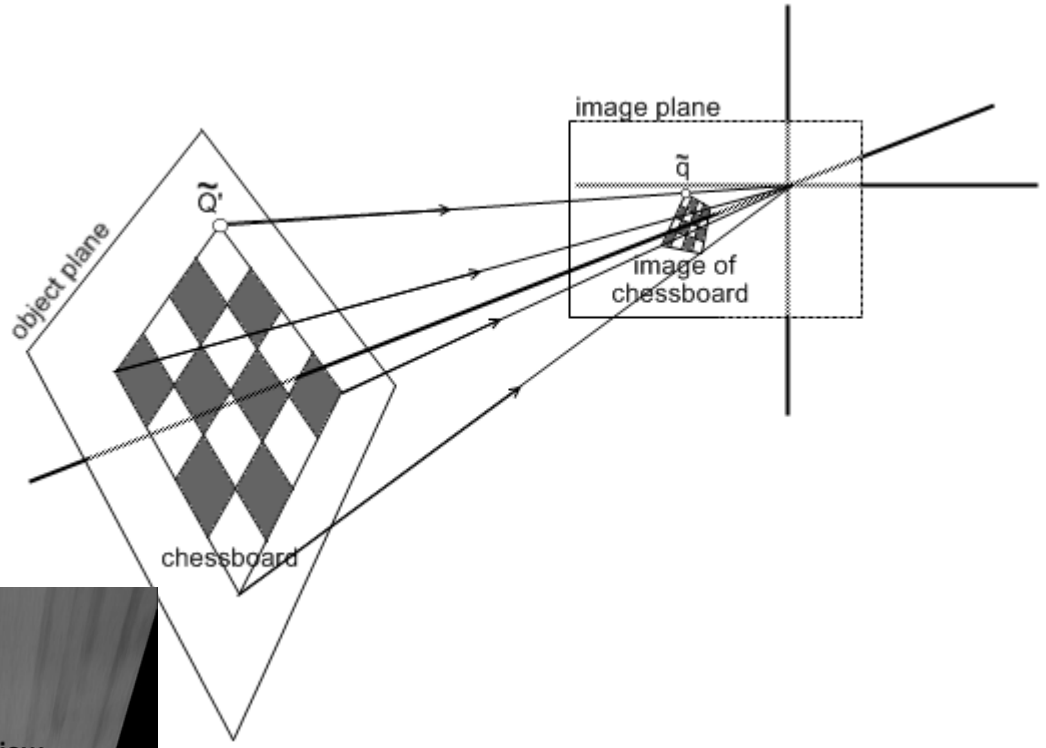
3D Stereo Vision

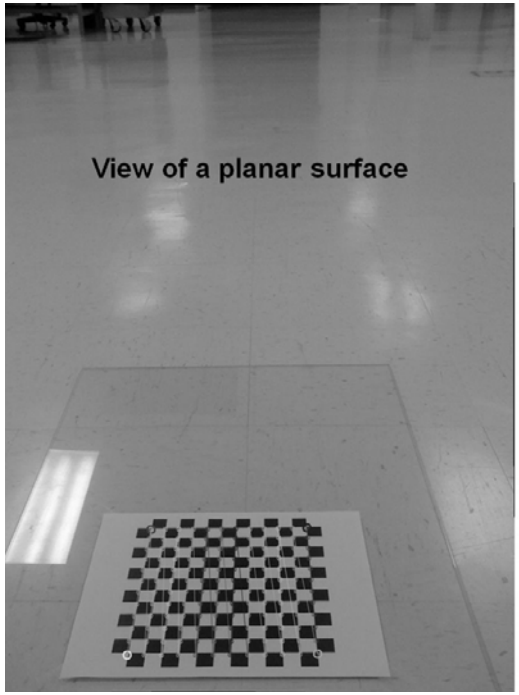
- State of the art stereo pipeline:

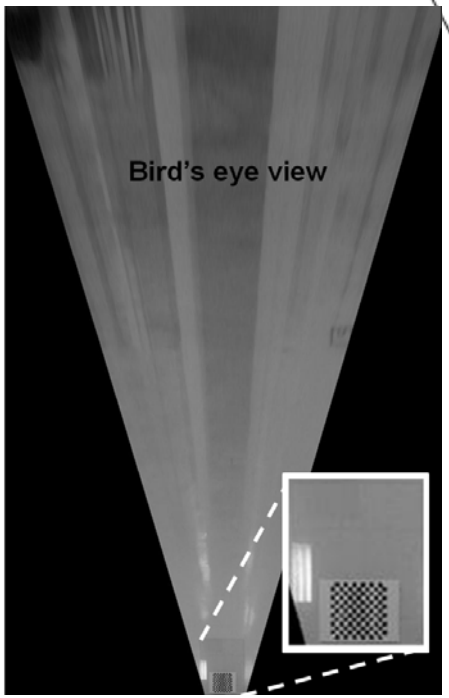


3D Stereo Vision

- Projections:

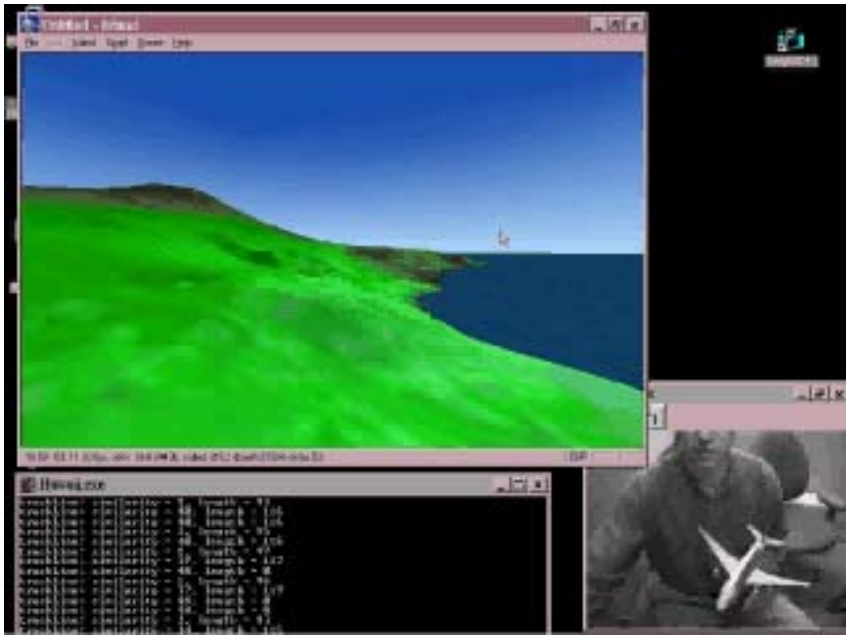


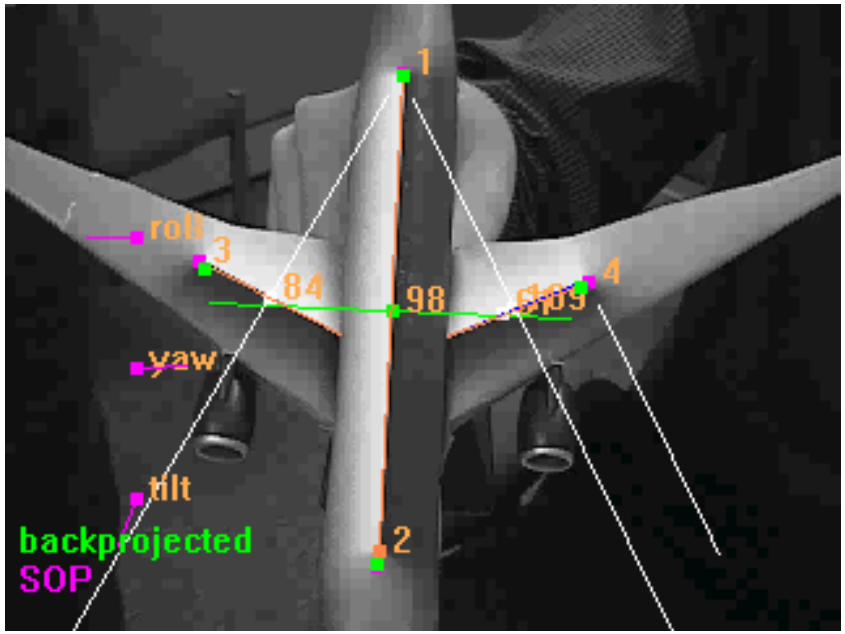




3D tracking

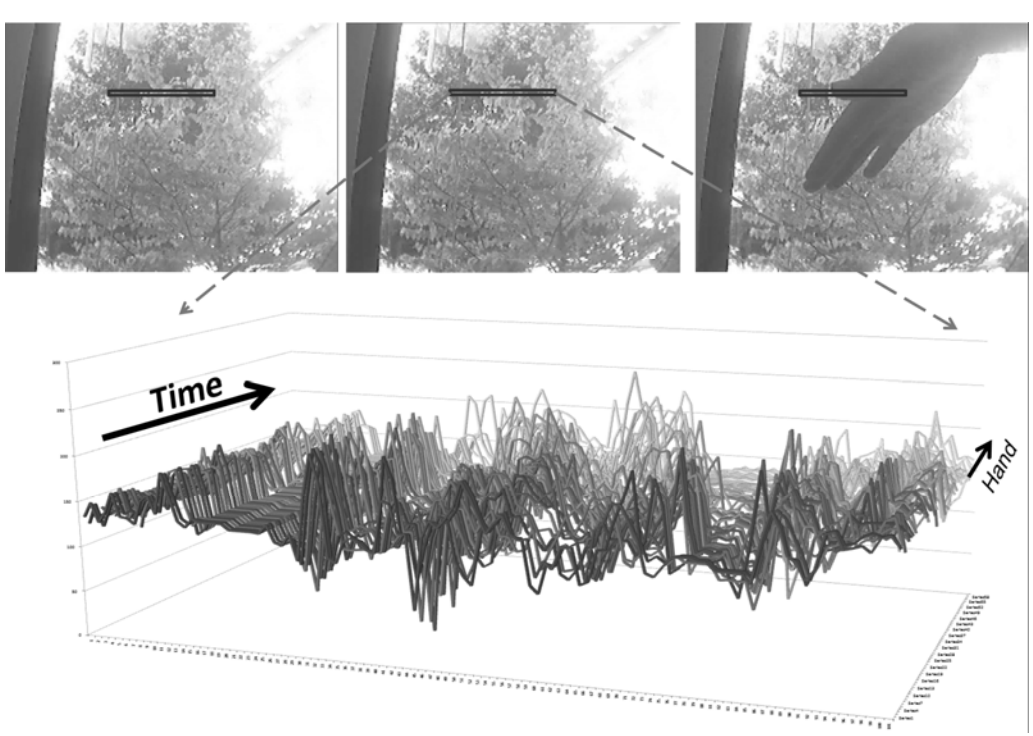
- Camera Calibration
- View Morphing
- POSIT

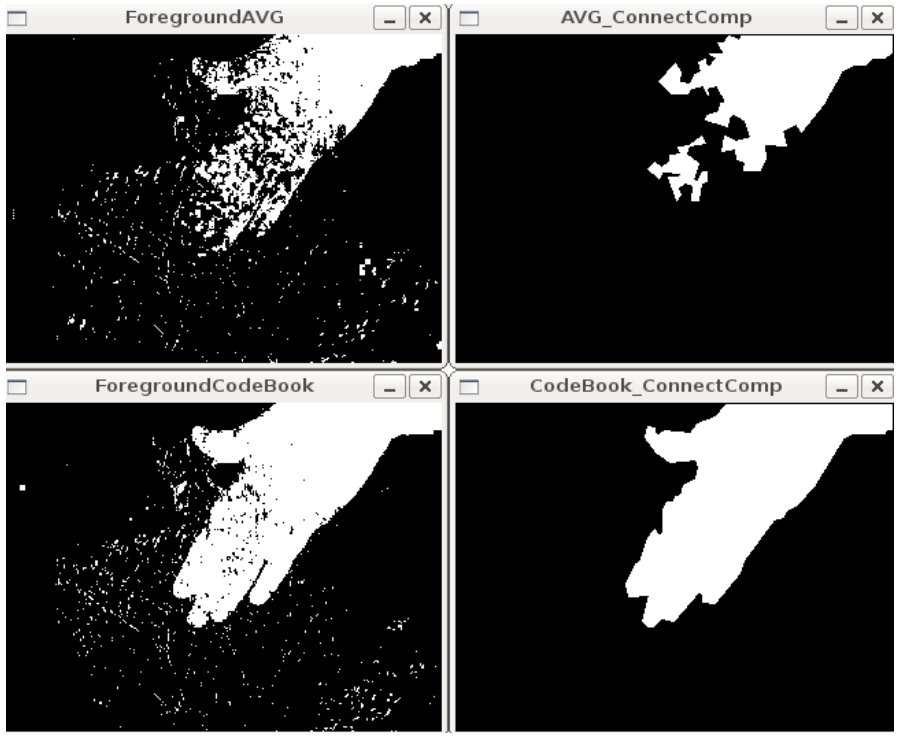




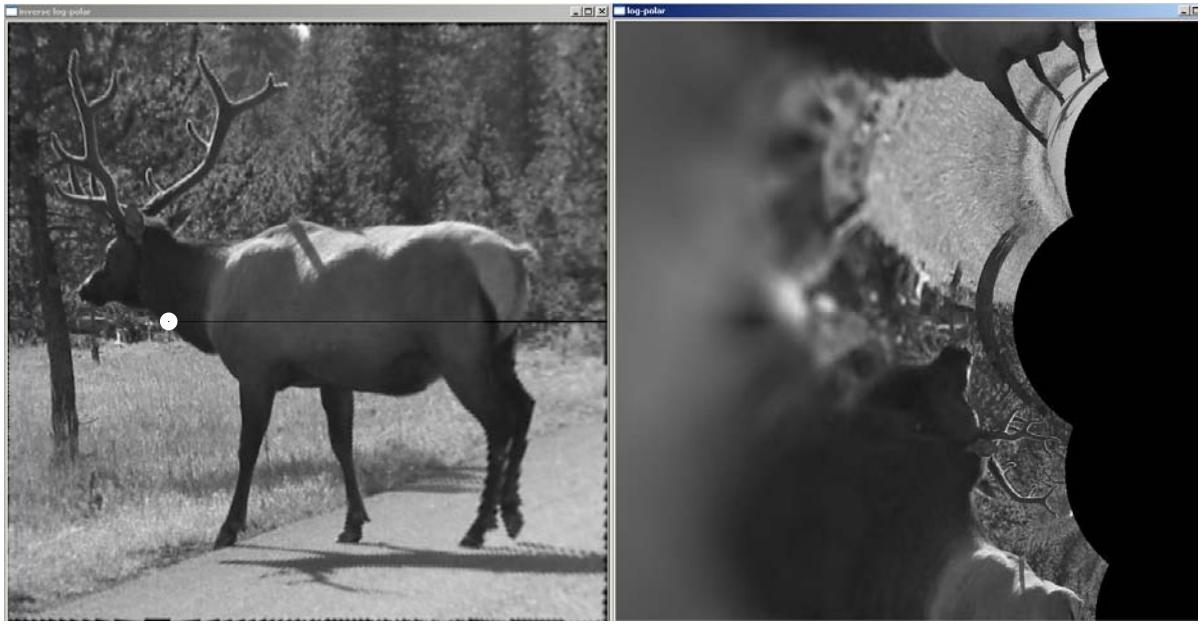
Utilities

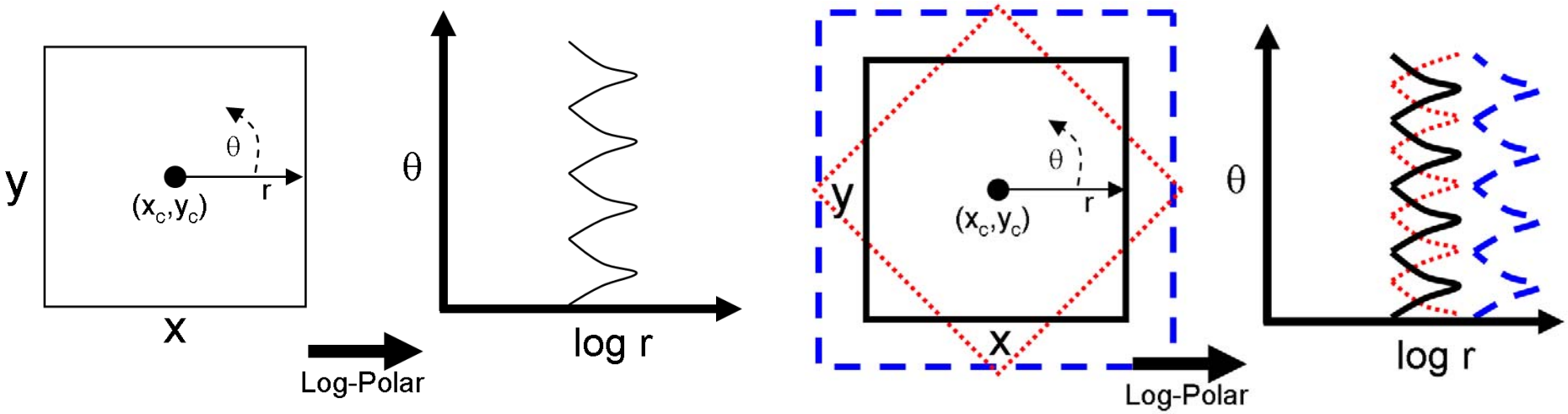
- Background segmentation object labeler
- Object selection labeler



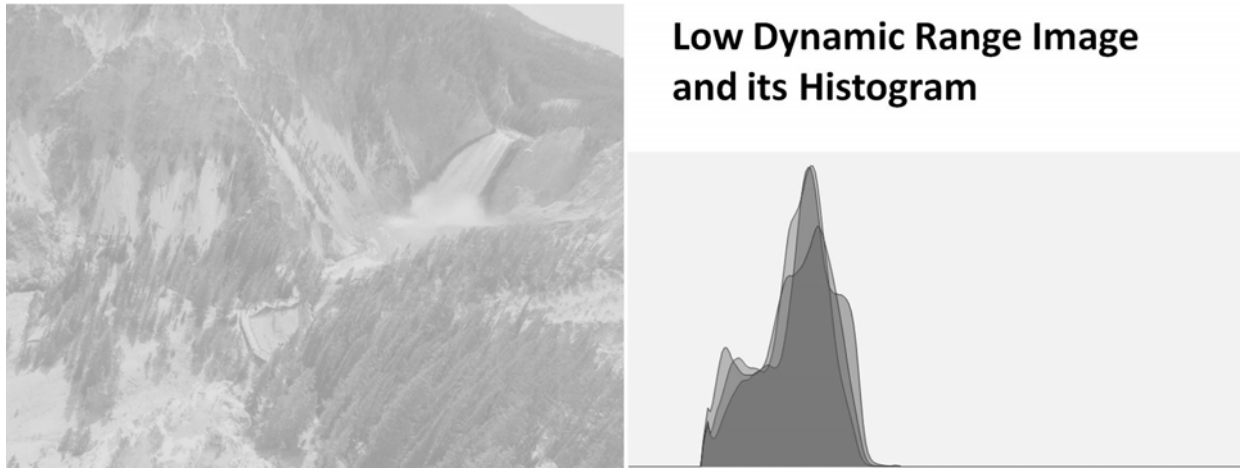


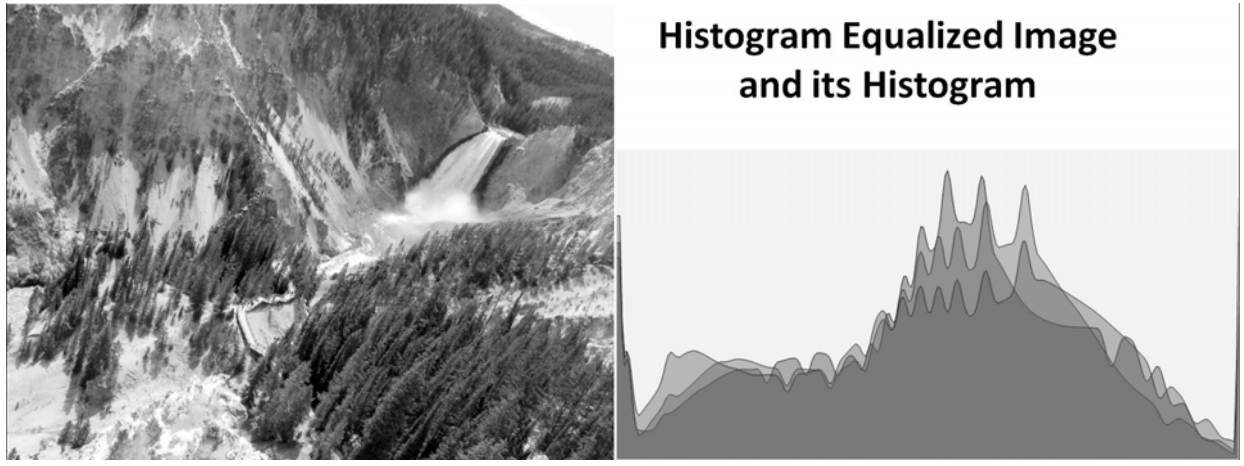
Space Variant vision: Log-Polar Transform





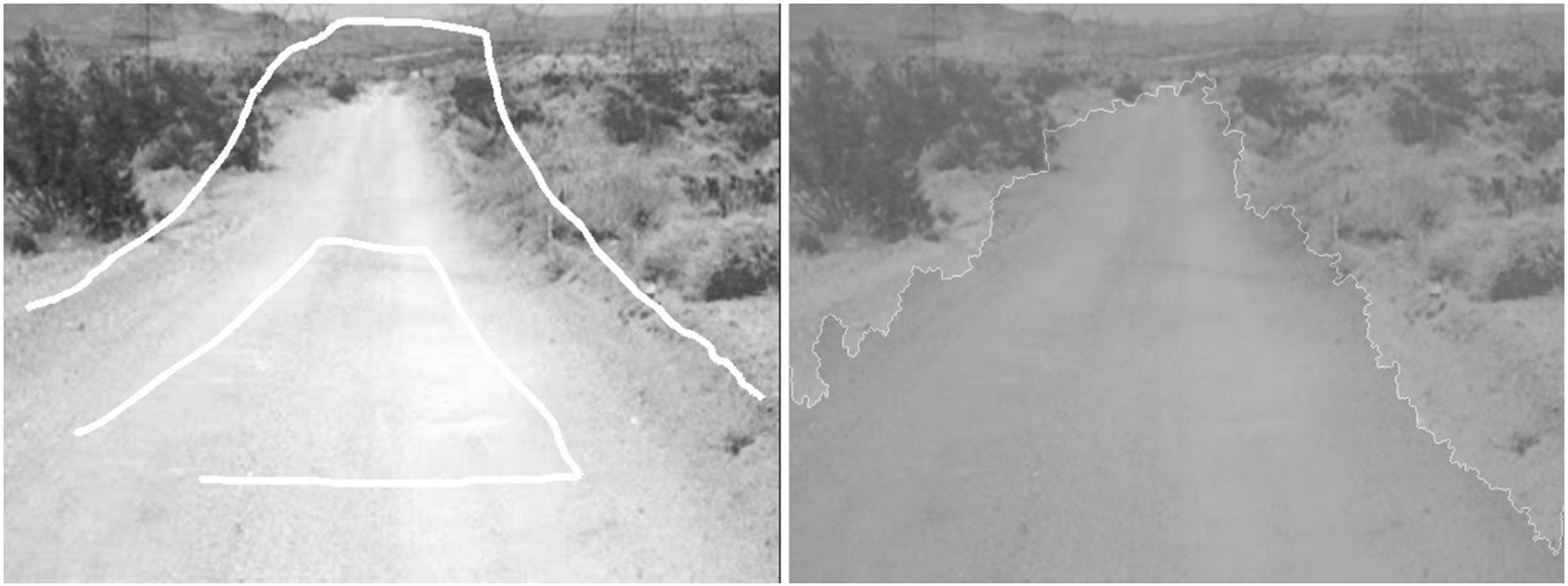
Histogram Equalization



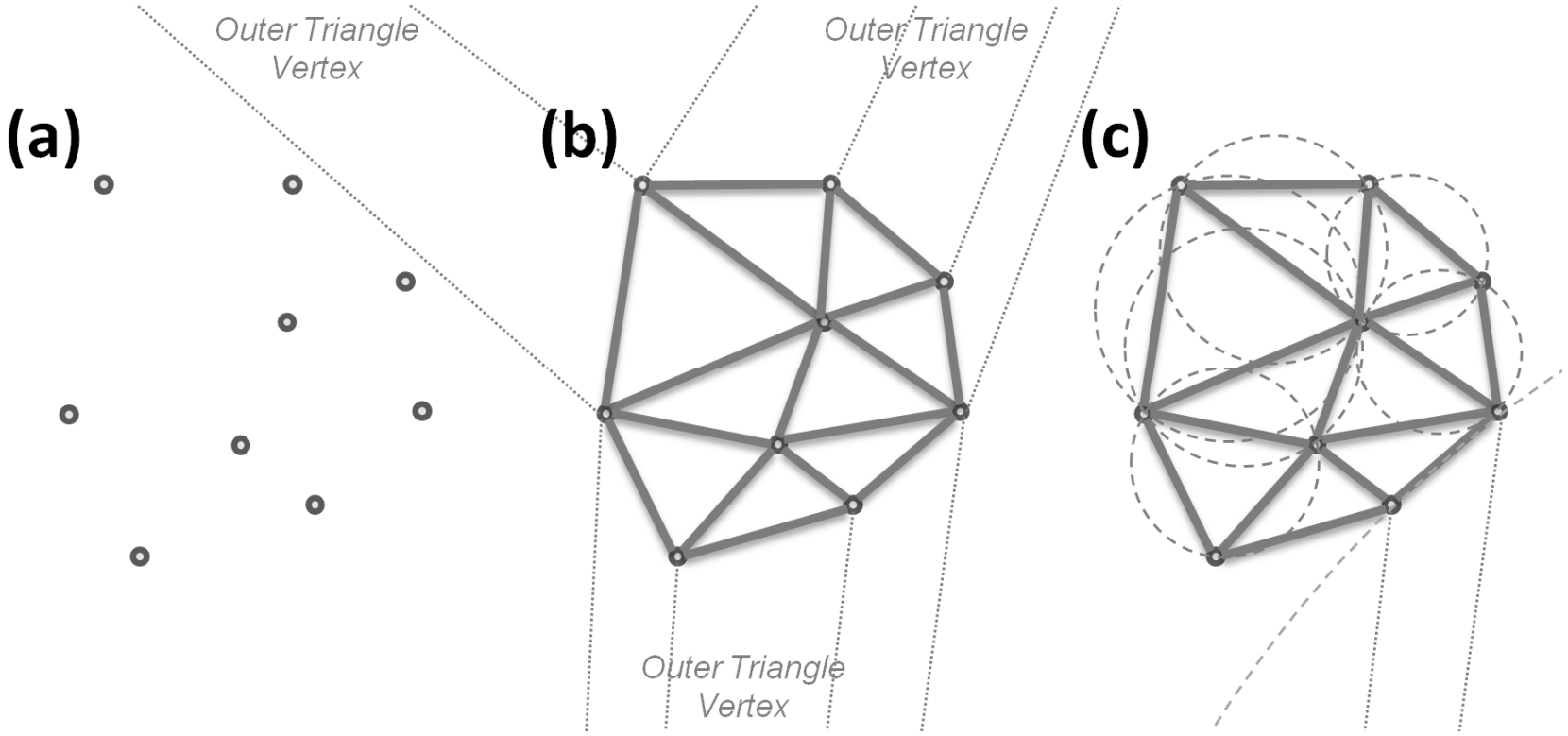


Segmentation

- Pyramid, mean-shift, graph-cut
- Here: Watershed



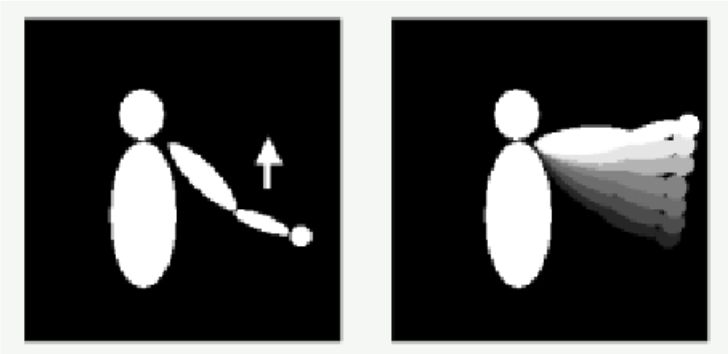
Delaunay Triangulation, Voronoi Tessellation



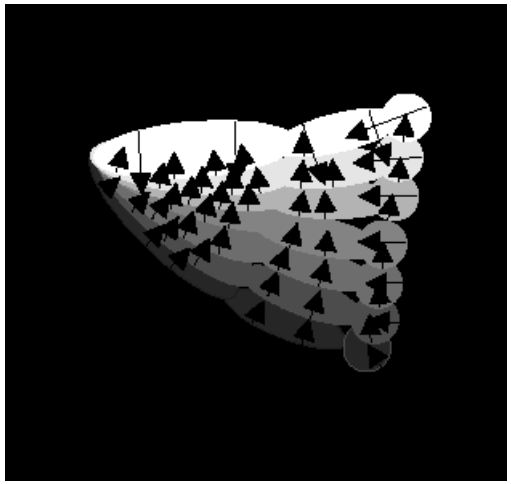
Motion Templates (My work with James Davies)

- Object silhouette
- Motion history images
- Motion history gradients
- Motion segmentation algorithm

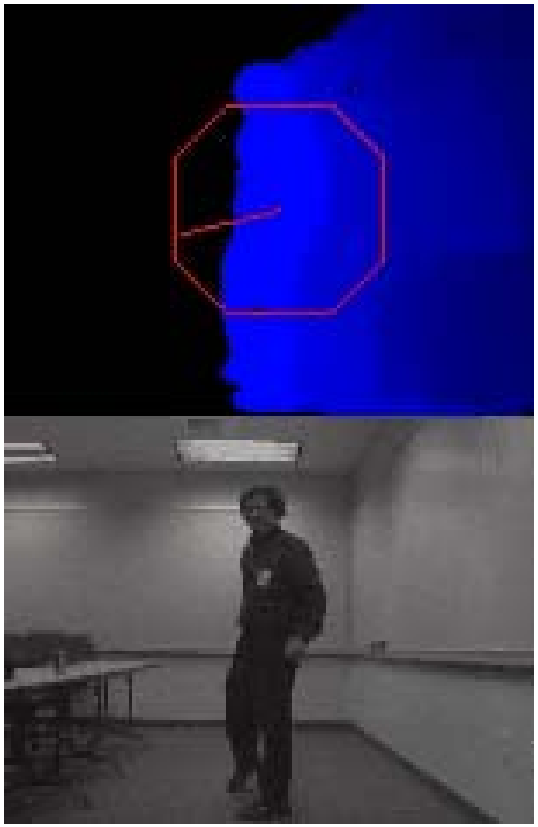
silhouette MHI



MHG



Segmentation, Motion Tracking and Gesture Recognition

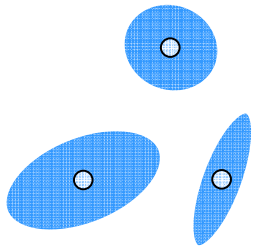


OpenCV Tour

- Installing OpenCV
- Useful structures and conventions
- I/O
- Function Overview
- Notable functions
  - ❖ Machine Learning

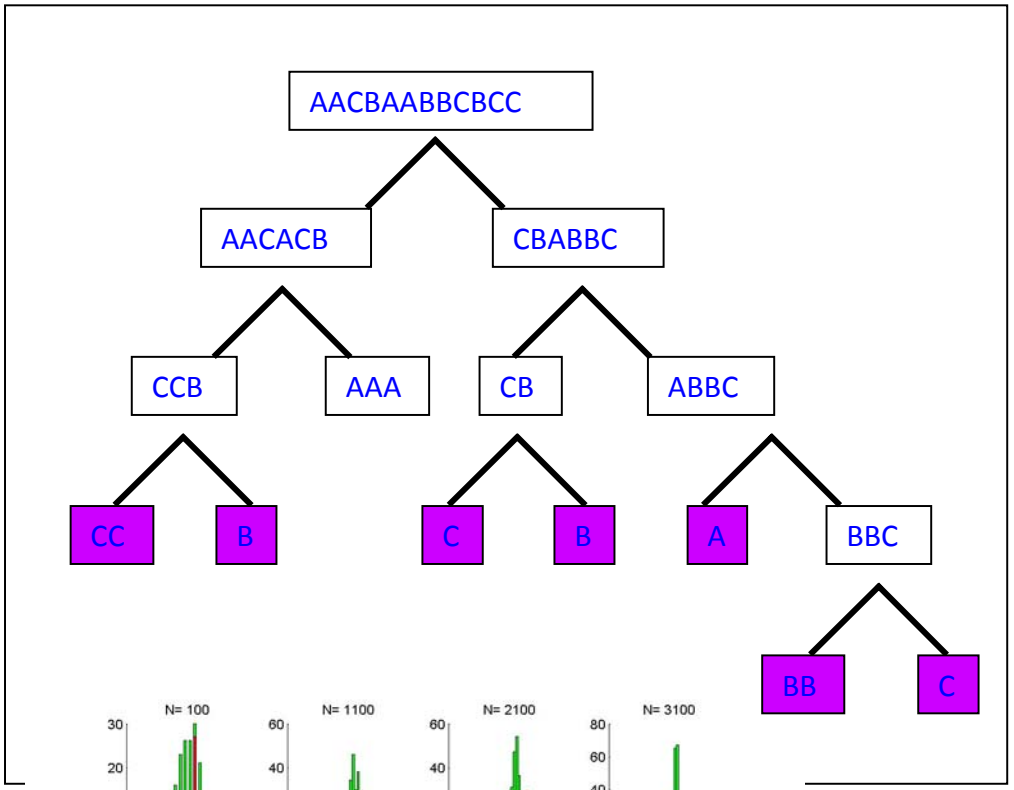
Machine Learning Library (MLL)

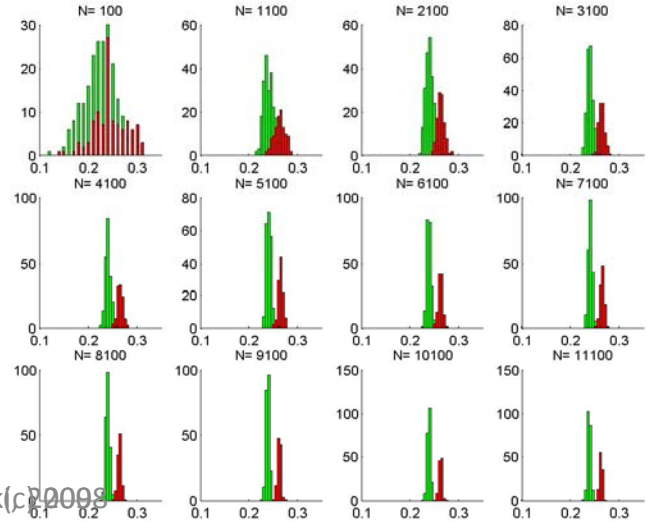
CLASSIFICATION / REGRESSION
CART
Naïve Bayes
MLP (Back propagation)
Statistical Boosting
Random Forests
SVM
K-NN
Face Detector
(Histogram matching)
(Correlation)
Stochastic Discrimination
Logistic



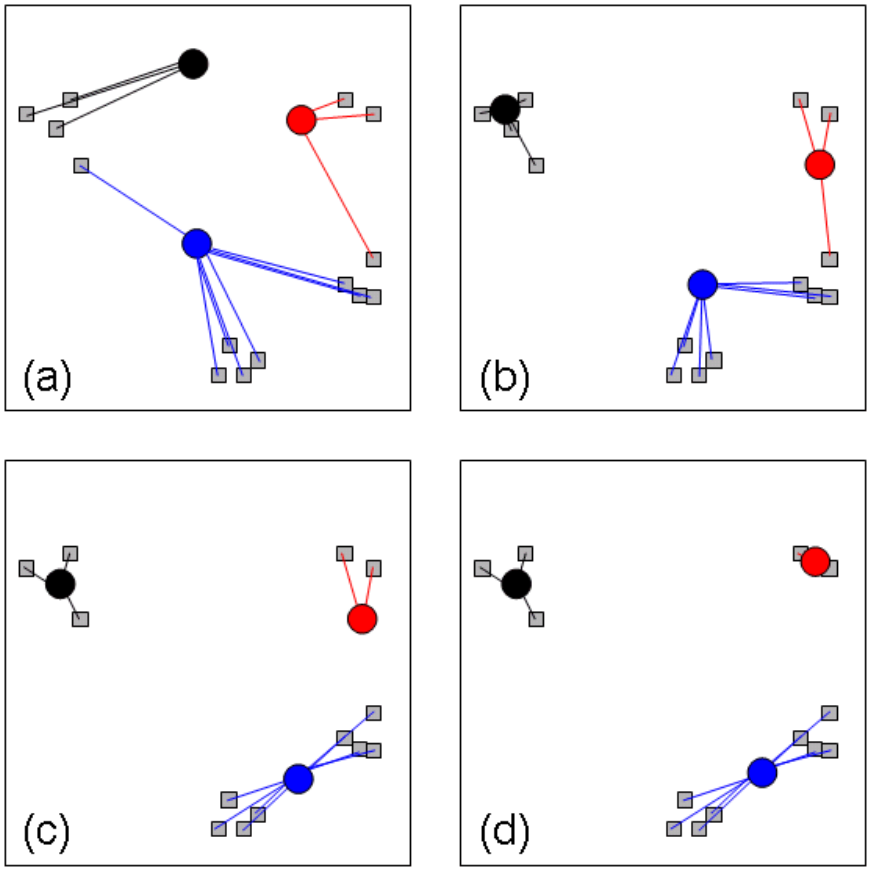
CLUSTERING
K-Means
EM
(Mahalanobis distance)
Agglomerative

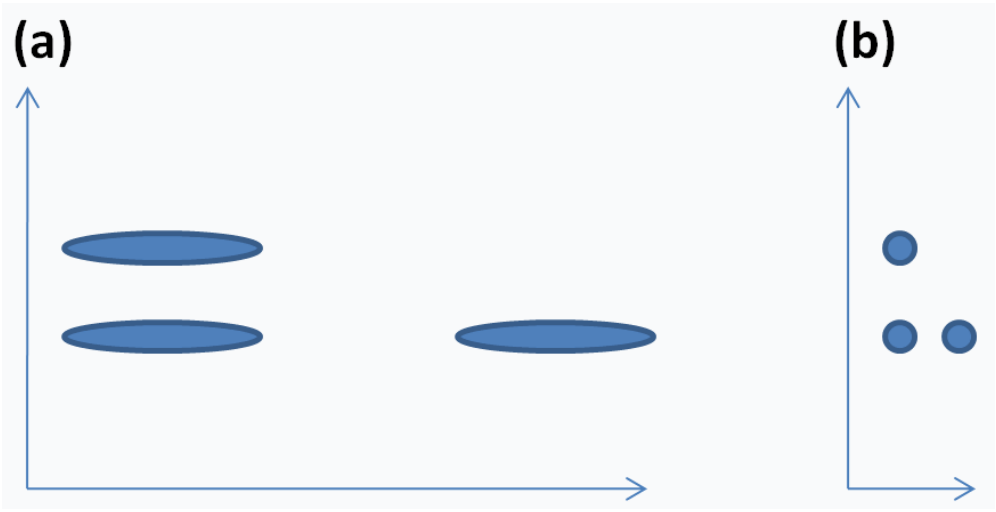
TUNING/VALIDATION
Cross validation
Bootstrapping
Variable importance
Sampling methods





K-Means, Mahalanobis





Patch Matching

Gesture Recognition

(Earlier work in gesture recognition)

Gestures: Up R L Stop OK

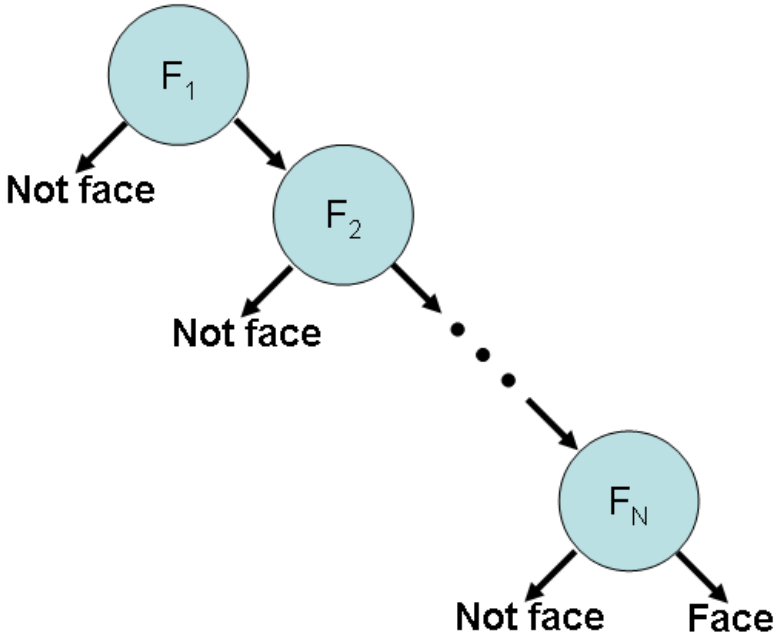
Gesture via:

Gradient histogram* based gesture recognition with Tracking.

Meanshift Algorithm used to track, histogram intersection with gradient used to recognize.

* Bill Freeman

Boosting: Face Detection with Viola-Jones Rejection Cascade



Outline

- Why OpenCV?
- What is OpenCV
- Use in Robotics at Willow Garage
- OpenCV Overview/Tour
- OpenCV Plans – Summer Release
- Questions?

Summer Release

- See:
  - http://opencv.willowgarage.com/wiki/OpenCV200906

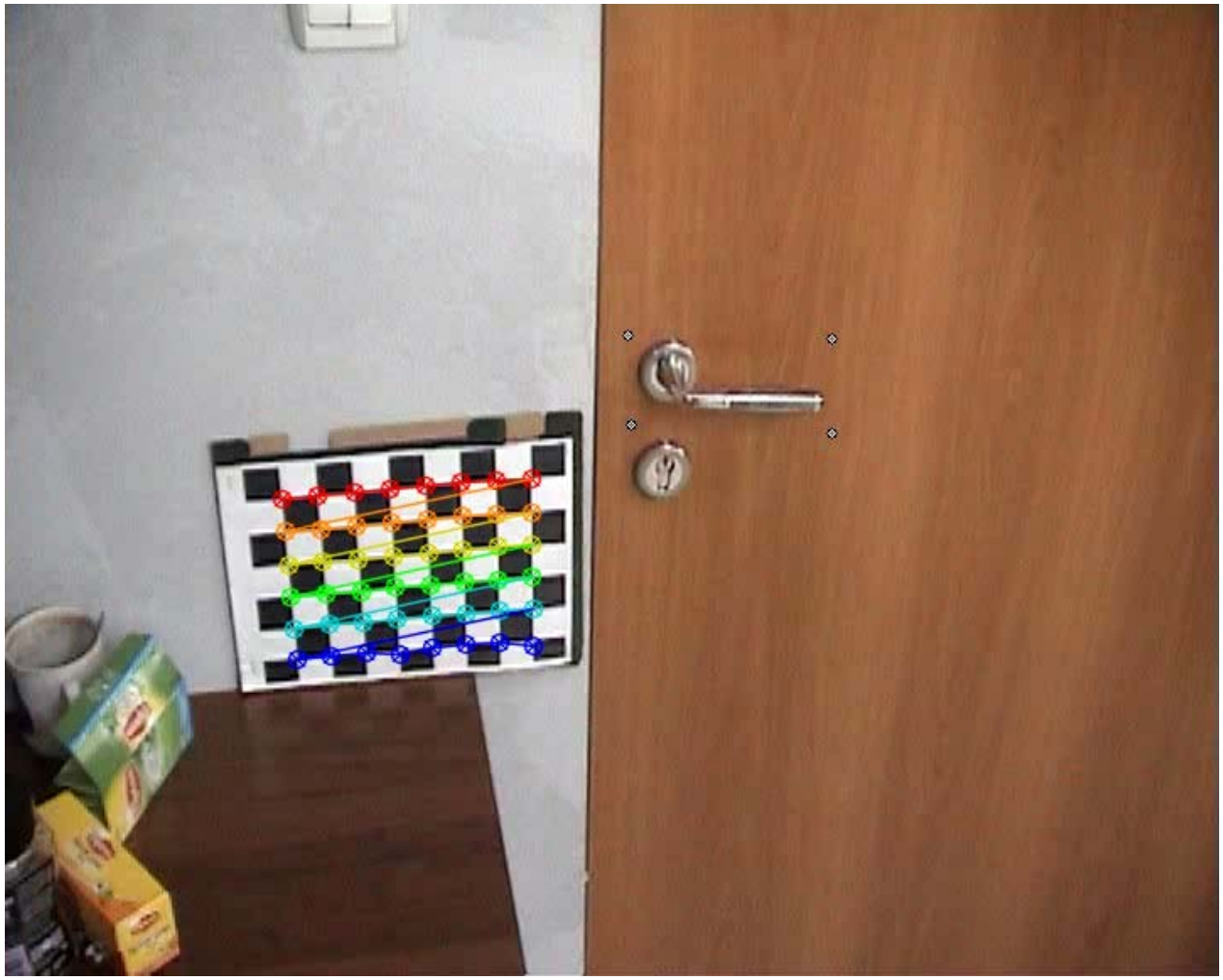
1. **New C++ interface (with backward compatibility),**
2. **(MUCH) Better Python interface with low-level capabilities.**
3. **More 2D feature detectors and descriptors;**
4. **Add 3D vision support:**

   Better stereo support. 3D reconstruction, (If we can ...) image stitching.
   - **3D feature detectors and descriptors.**

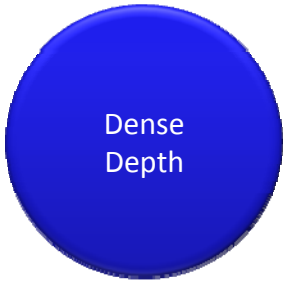
5. **Application section:**
   - **Human detector**
   - **Camera location from chessboard/3D bounding box**

6. **Better optical flow (dense, sparse), particle filtering, deformable templates.**
7. **New functionality and optimizations in MLL,**
   - **Algorithmic and regression tests, general cascade detector.**
8. **Optimization and threading.**
9. **Revised documentation.**
10. **Highgui enhancements.**

Example (New) Application section



Example 3D Support: Dense Depth



Useful OpenCV Links

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

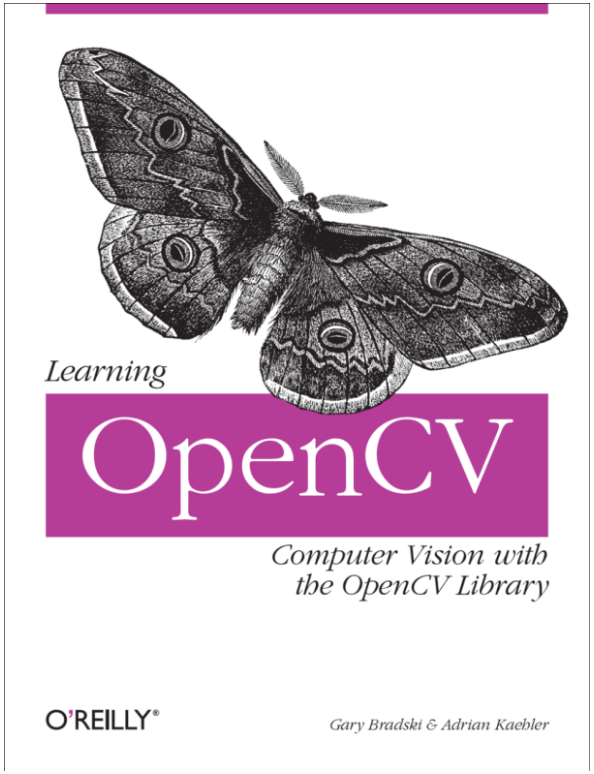
User Group:
http://tech.groups.yahoo.com/group/OpenCV/join

OpenCV Code Repository:
http://sourceforge.net/projects/opencvlibrary/

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

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

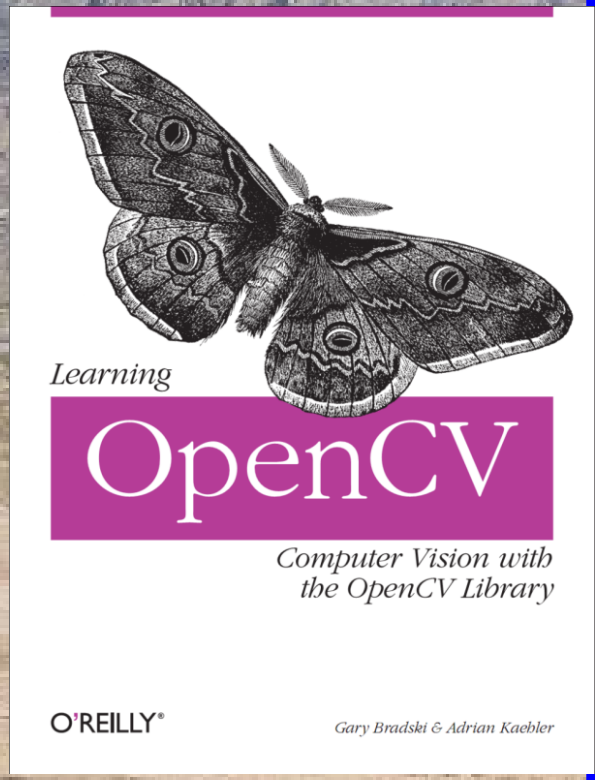
Code examples from the book:
http://examples.oreilly.com/9780596516130/



Outline

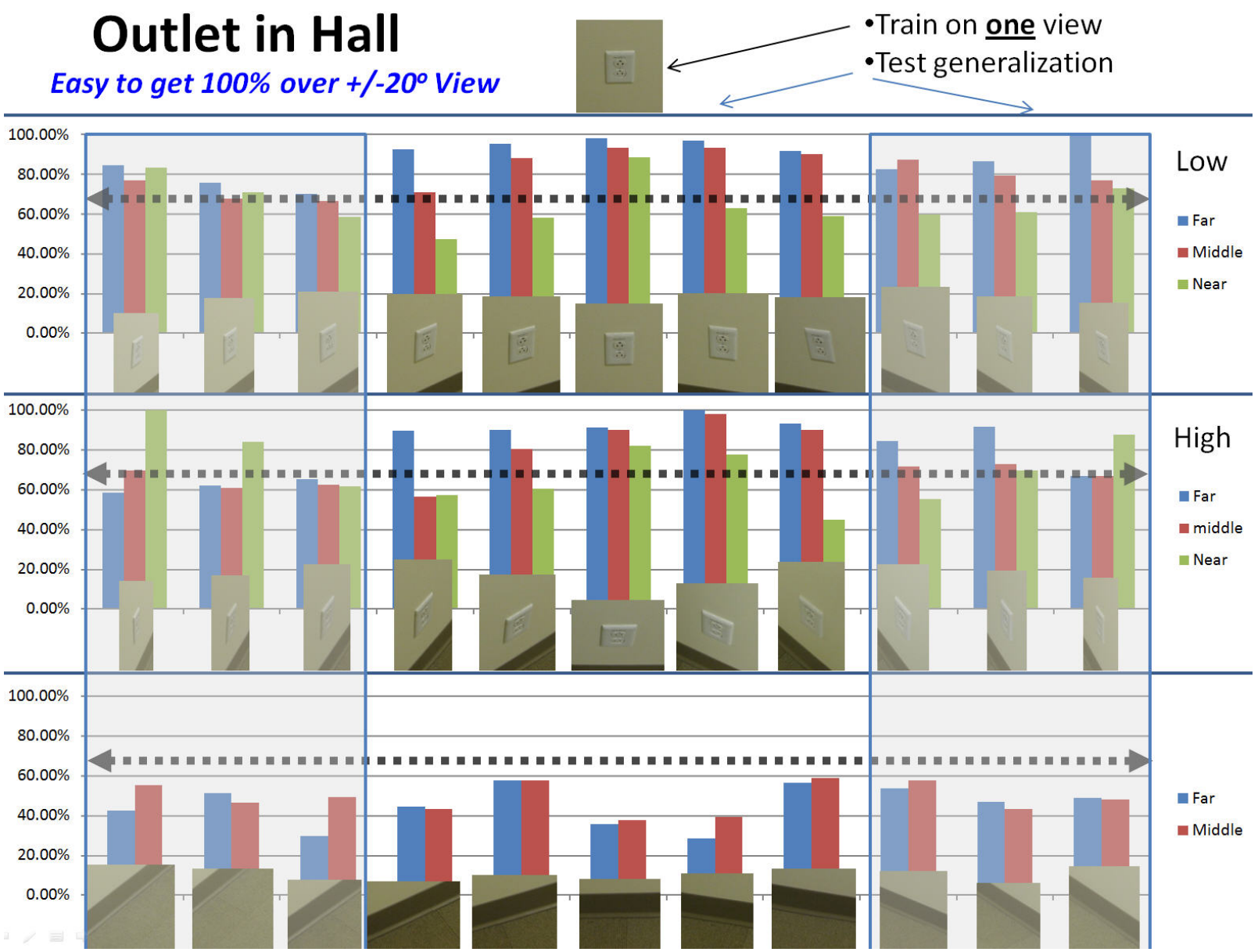
- Why OpenCV?
- What is OpenCV
- Use in Robotics at Willow Garage
- OpenCV Overview/Tour
- OpenCV Plans – Summer Release
- Questions?

Questions?


Learning
OpenCV
Computer Vision with the OpenCV Library
O'REILLY
Gary Bradski & Adrian Kaebler

Stop

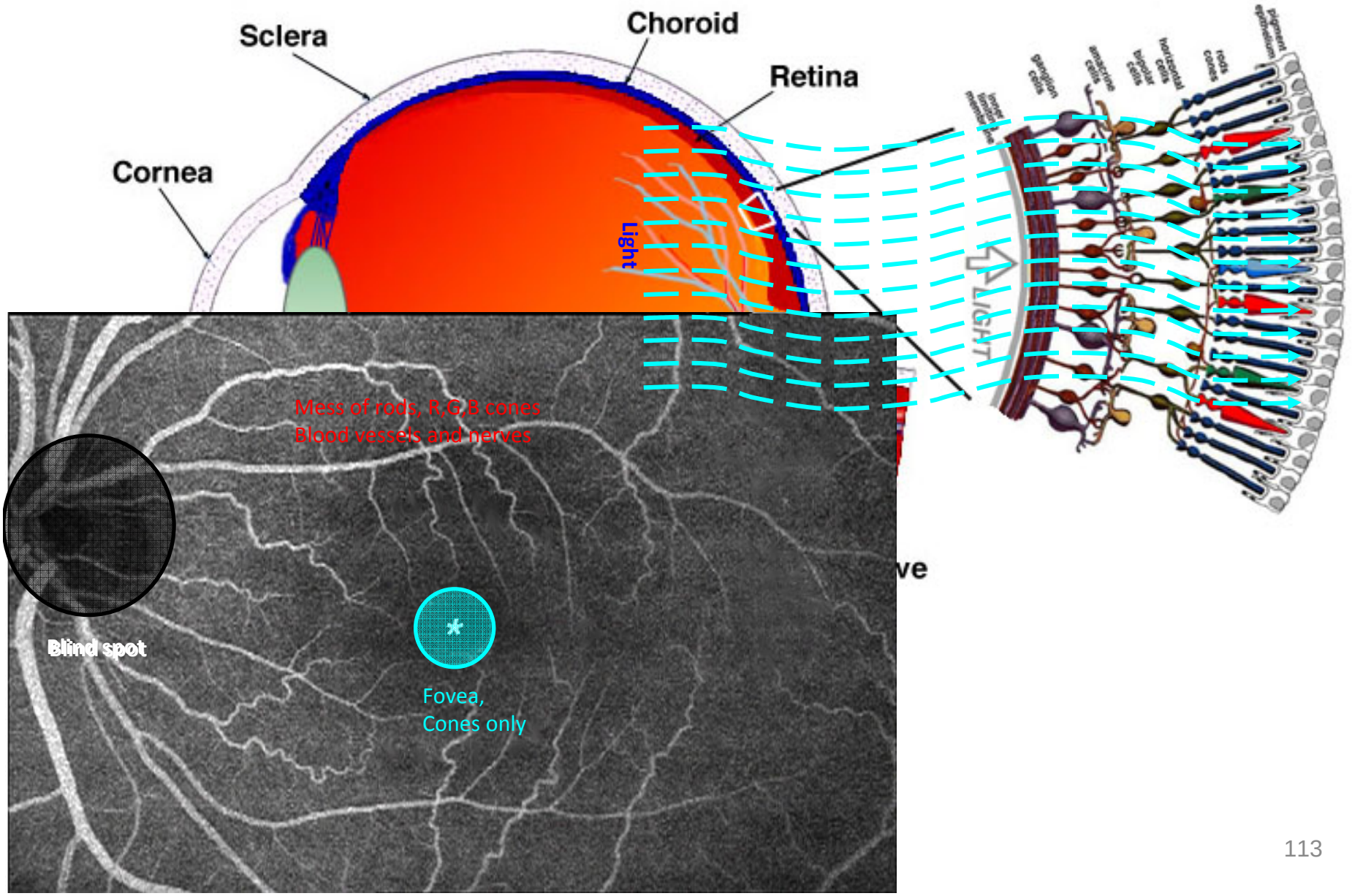
Recent Work

- 
Outlet in Hall
Easy to get 100% over +/-20º View
•Train on one view
•Test generalization
100.00%
80.00%
60.00%
40.00%
20.00%
0.00%
Low
Far
Middle
Near
High
middle
Near
Far
Middle

OpenCV Status

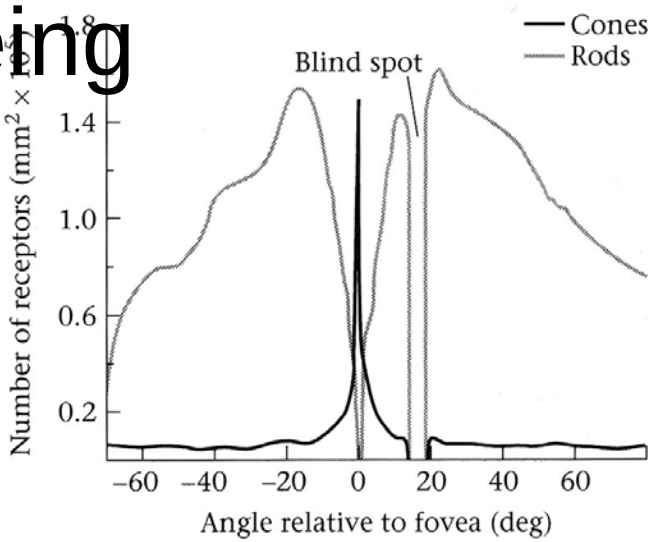
- Willow Garage now supporting
  - Russian team by December 15: 5 full time
- New template API being worked on
  - First with Matrix Class, better support for alpha channel
- Areas of Expansion:
  - Features:
    - SURF, STAR detector, C-Descriptor, Spin, 3D Shape
  - Odometry
  - Object Recognition (Extreme Random Forests)
  - Calibration, Laser-Camera
  - Hausdorf distance and alignment
- Cmake, now on SVN
- Amazon Mechanical Turk, human labeling interface

Seeing


Sclera
Choroid
Retina
Cornea
Light
Mess of rods, R,G,B cones
Blood vessels and nerves
Blind spot
Fovea,
Cones only

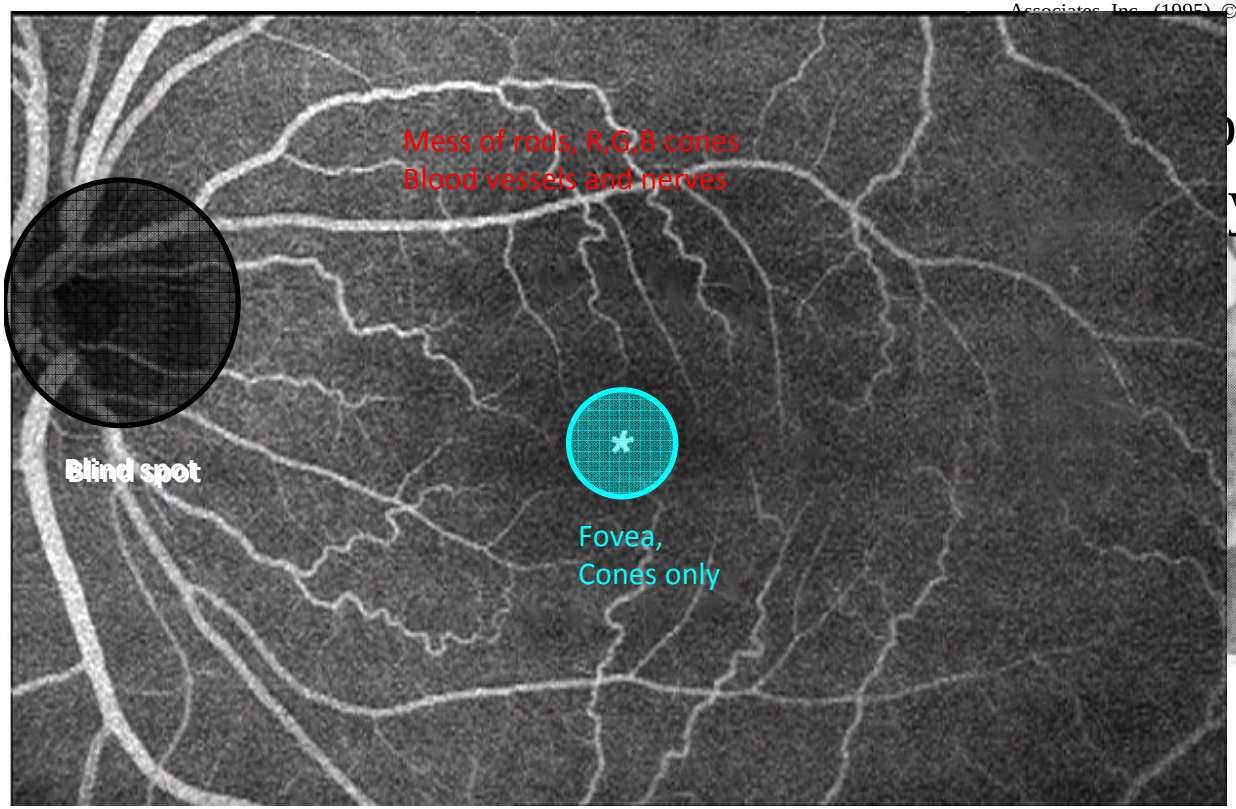
Seeing

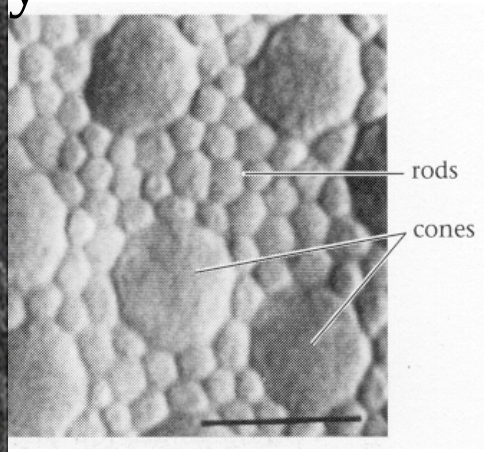
The distribution of rods and cones across the retina



Reprinted from Foundations of Vision, by B. Wandell, Sinauer Associates, Inc. (1995) © 1995 Sinauer Associates, Inc.

ones in
y





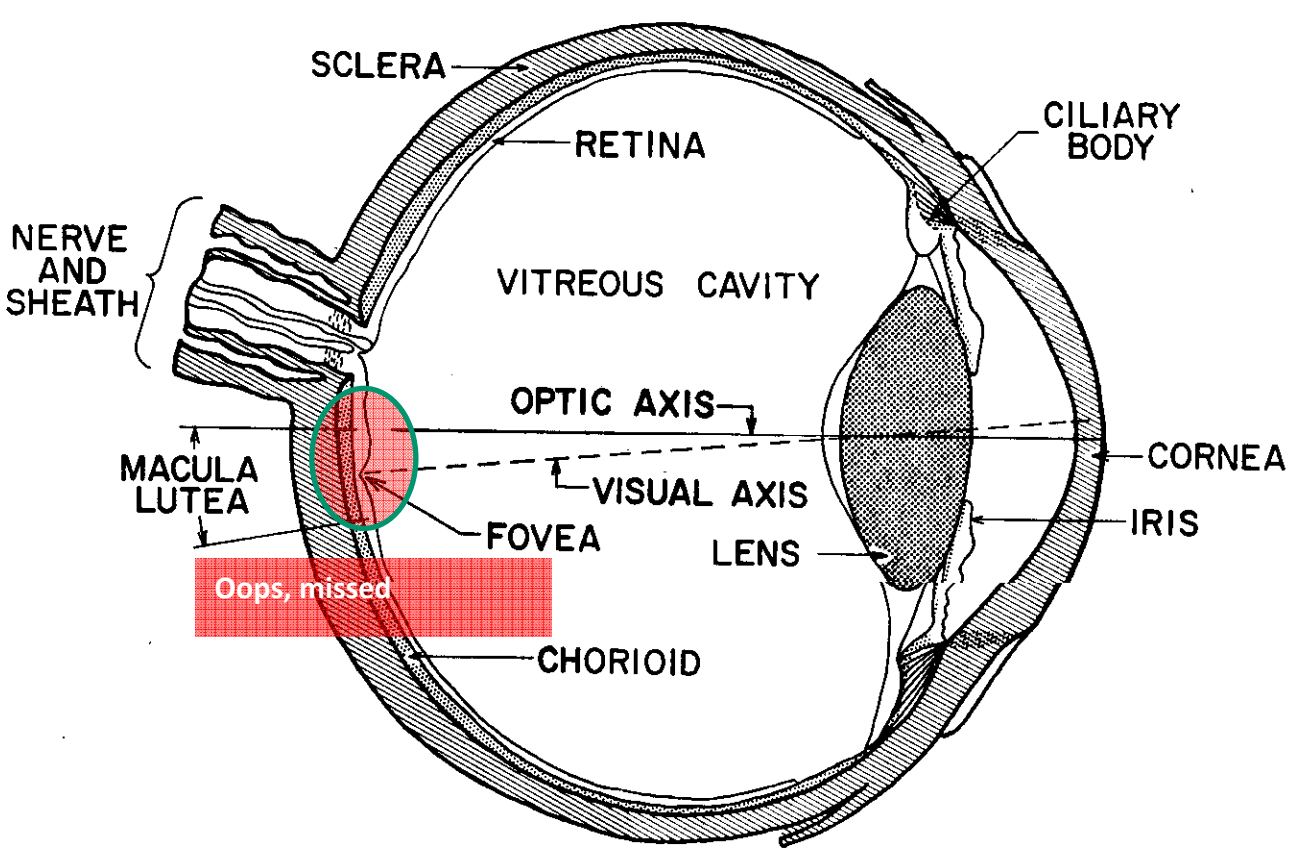
(gray)

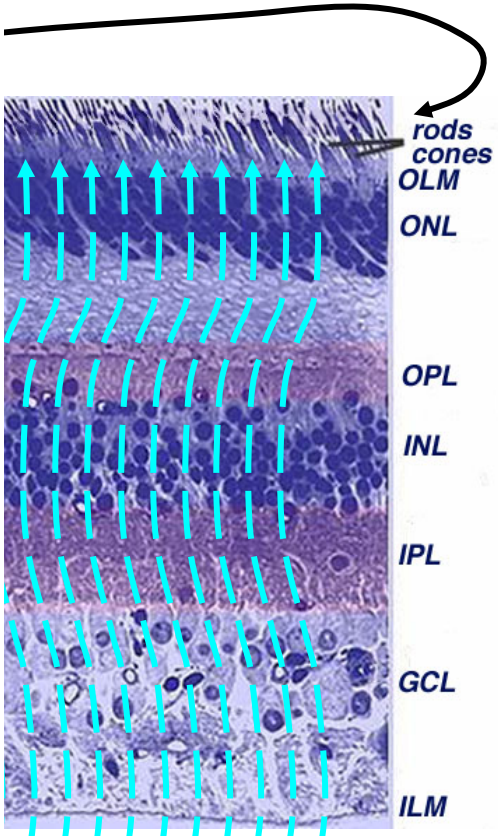
(color)

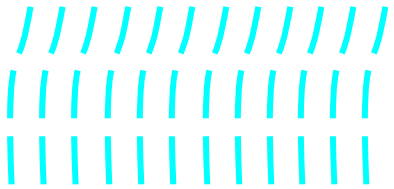
Seeing

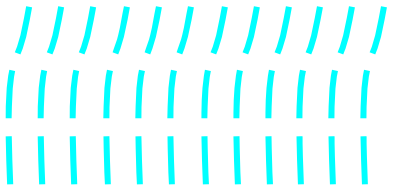
Sloppy Engineering: Human Eye

RGB **B/W happens here**



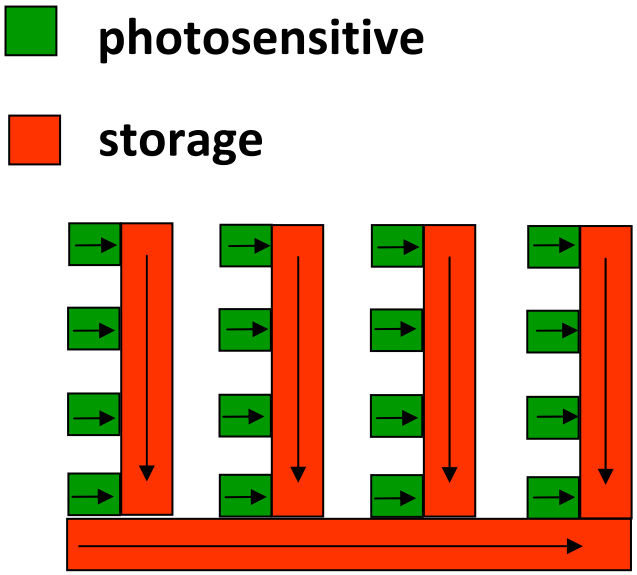




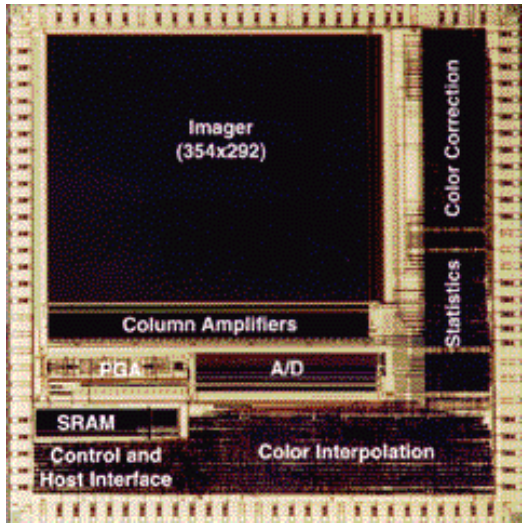


Seeing by silicon

- **CCD**
  - Cell are chained
  - Collect and dump

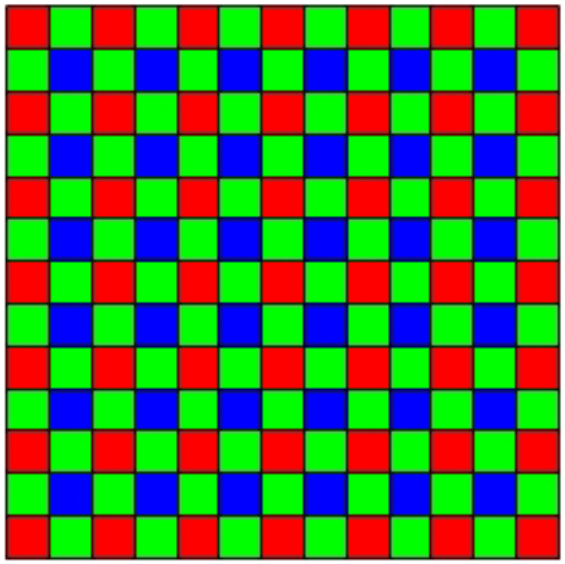


- **CMOS**
  - Same collectors, but
  - Each site independent
  - More like looking up the memory of what light fell where



Seeing Color is done by filters

- Interpolate Red Green Blue



Bayer filter

Outline

- Insights into vision
- About OpenCV
- Installing OpenCV
- Useful structures and conventions
- I/O
- Function Overview
- Notable functions
- Wither computer vision and OpenCV