OpenCV Tutorial

Victor Eruhimov
Itseez, CTO

http://opencv.willowgarage.com
www.willowgarage.com
www.itseez.com
Outline

- OpenCV Overview
  - Cheatsheet
  - Simple Programs
  - Tour
  - Features2D
  - Applications
OpenCV Overview:

> 500 algorithms

General Image Processing Functions

Image Pyramids

Segmentation

Geometric descriptors

Transforms

Features

Machine Learning:
- Detection,
- Recognition

Tracking

Utilities and Data Structures

Camera calibration, Stereo, 3D

Fitting

Matrix Math

opencv.willowgarage.com

> 500 algorithms

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

http://opencv.willowgarage.com
OpenCV History

• **Original goal:**
  - Accelerate the field by lowering the bar to computer vision
  - Find compelling uses for the increasing MIPS out in the market

• **Timeline:**

• **Staffing:**
  - Climbed in 1999 to average 7 first couple of years
  - Starting 2003 support declined between zero and one with exception of transferring the machine learning from manufacturing work I led (equivalent of 3 people).
  - Support to zero the couple of years before Willow.
  - 5 people over the last year
New Directory Structure

• Re-Organized in terms of processing pipelines
• Code site:
  https://code.ros.org/gf/project/opencv/
  – Core
  – Calibration, features, I/O, img processing
  – Machine Learning, Obj. Rec
  – Python
• ∼2.5M downloads
OpenCV Tends Towards Real Time

Comparison with other libs: Performance

Test station: Pentium M, 1.7GHz
Libraries: OpenCV 1.0pre, IPP 5.0, LTI 1.9.14, VXL 1.4.0
2D DFT: Forward Fourier Transform of 512x512 image
Resize: 512x512→384x384 bilinear interpolation, 8-bit 3-channel image
Optical flow: 520 points tracked with 41x41 window, 4 pyramid levels.
Neural Net: mushroom benchmark from FANN

http://opencv.willowgarage.com
Software Engineering

• Works on:
  – Linux, Windows, Mac OS

• Languages:
  – C++, Python, C

• Online documentation:
  – Online reference manuals: C++, C and Python.

• We’ve been expanding Unit test code

• Will soon standardize on cxx or Google’s test system.

• TEST COVERAGE:
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
What’s New in OpenCV?

- Complete C++ interface
- Organization into functional code stacks
- Good Python interface
- More optimizations and test code
- Revamped online docs
- Inpainting image restoral
- Grabcut segmentation
What’s Coming in November 2010
OpenCV 2.2?

• Detector/Descriptor pipeline (Features2D)
  – Many supporting detectors and descriptor features
• Easy interface to Pascal VOC
• BOW and Latent SVM classification engines
• Experimental User Contrib
• Focus detector?
• Visualization (“HighGUI”) will be based on Qt
• Official support of Android OS
• Updated FLANN library
• Limited Cuda support (stereo)
What’s in Progress?

• Image stitching, spherical and cylindrical panoramas
• Object Recognition Infrastructure (REIN)
  – Includes Felzenschwalb’s algorithm as an example
  – Many more feature types such as LARK, PAS, Color SIFT, …
  – Obj. rec and 6DOF pose using conjunctions of interest point
• Spherical calibration
• Generalized calibration patterns
• Generalized 2D Barcodes
• 3-camera stereo calibration
• Converters and Serializers that work with ROS
• 3D Model capture from SFM and other
• Visual Odometry
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

Screen shots by Gary Bradski, 2005
Useful OpenCV Links

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

**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:

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

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

**User Group (39717 members):**
http://tech.groups.yahoo.com/group/OpenCV/join
Outline

• OpenCV Overview
• Cheatsheet
• Simple Programs
• Tour
• Features2D
• Applications
## Key OpenCV Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>Template 2D point class</td>
</tr>
<tr>
<td>Point3</td>
<td>Template 3D point class</td>
</tr>
<tr>
<td>Size</td>
<td>Template size (width, height) class</td>
</tr>
<tr>
<td>Vec</td>
<td>Template short vector class</td>
</tr>
<tr>
<td>Scalar</td>
<td>4-element vector</td>
</tr>
<tr>
<td>Rect</td>
<td>Rectangle</td>
</tr>
<tr>
<td>Range</td>
<td>Integer value range</td>
</tr>
<tr>
<td>Mat</td>
<td>2D dense array (used as both a matrix or an image)</td>
</tr>
<tr>
<td>MatND</td>
<td>Multi-dimensional dense array</td>
</tr>
<tr>
<td>SparseMat</td>
<td>Multi-dimensional sparse array</td>
</tr>
<tr>
<td>Ptr</td>
<td>Template smart pointer class</td>
</tr>
</tbody>
</table>
New “Image”: cv::Mat

Matrix Basics
Create a matrix
Mat image(240, 320, CV_8UC3);

Reallocate a pre-declared matrix
image.create(480, 640, CV_8UC3);

Create a matrix initialized with a constant
Mat A33(3, 3, CV_32F, Scalar(5));
Mat B33(3, 3, CV_32F); B33 = Scalar(5);
Mat C33 = Mat::ones(3, 3, CV_32F)*5.;
Mat D33 = Mat::zeros(3, 3, CV_32F) + 5.;

Create a matrix initialized with specified values
double a = CV_PI/3;
Mat A22 = (Mat_<float>(2, 2) <<
    cos(a), -sin(a),
    sin(a), cos(a));
float B22data[] = {cos(a), -sin(a), sin(a), cos(a)};
Mat B22 = Mat(2, 2, CV_32F, B22data).clone();

Initialize a random matrix
randu(image, Scalar(0), Scalar(256)); // uniform dist
randn(image, Scalar(128), Scalar(10)); // Gaussian dist

Convert matrix to/from other structures
(without copying the data)
Mat image.alias = image;
float* Idata=new float[480*640*3];
Mat I(480, 640, CV_32FC3, Idata);
vector<Point> iptvec(10);
Mat iP(iptvec); // iP - 10x1 CV_32SC2 matrix
IplImage* oldC0 = cvCreateImage(cvSize(320, 240),16,1);
Mat newC0 = cvarrToMat(oldC0);
IplImage oldC1 = newC;
CvMat oldC2 = newC;

... (with copying the data)
Mat newC2 = cvarrToMat(oldC0).clone();
vector<Point2f> ptvec = Mat.<Point2f>(iP);

Access matrix elements
A33.at<float>(i,j) = A33.at<float>(j,i)+1;
Mat dyImage(image.size(), image.type());
for(int y = 1; y < image.rows-1; y++) {
    Vec3b* prevRow = image.ptr<Vec3b>(y-1);
    Vec3b* nextRow = image.ptr<Vec3b>(y+1);
    for(int x = 0; y < image.cols; x++) {
        for(int c = 0; c < 3; c++) {
            dyImage.at<Vec3b>(y,x)[c] =
                saturate_cast<char>(
                    nextRow[x][c] - prevRow[x][c]);
        }
    }
}
A33.row(1) = A33.col(1);
image.copyTo(dyImage);
It should do the right thing when it goes out of scope
you can also easily make stl vectorts or maps
out of Mat.
Mat are Simple

Mat M(480,640,CV_8UC3); // Make a 640x480 img
Rect roi(100,200, 20,40); // Make a region of int
Mat subM = M(roi); // Take a sub region, // no copy is done

Mat_<Vec3b>::iterator it= subM.begin<Vec3b>(), itEnd = subM.end<Vec3b>(); //0 out places in subM where blue > red
for(; it != itEnd; ++it)
    if( (*it)[0] > (*it)[2]) (*it)[0] = 0;
Matrix Manipulation

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

Example 1. Smooth image ROI in-place
Mat imgroi = image(Rect(10, 20, 100, 100));
GaussianBlur(imgroi, imgroi, Size(5, 5), 1.2, 1.2);

Example 2. Somewhere in a linear algebra algorithm
m.row(i) += m.row(j)*alpha;

Example 3. Copy image ROI to another image with conversion
Rect r(1, 1, 10, 20);
Mat dstroi = dst(Rect(0,10,r.width,r.height));
src(r).convertTo(dstroi, dstroi.type(), 1, 0);
Simple Matrix Operations

- `add()`, `subtract()`, `multiply()`, `divide()`, `absdiff()`, `bitwise_and()`, `bitwise_or()`, `bitwise_xor()`, `max()`, `min()`, `compare()`

- Correspondingly, addition, subtraction, element-wise multiplication ... comparison of two matrices or a matrix and a scalar.

Example. Alpha compositing function:

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

- `sum()`, `mean()`, `meanStdDev()`, `norm()`, `countNonZero()`, `minMaxLoc()`

- Various statistics of matrix elements.

- `exp()`, `log()`, `pow()`, `sqrt()`, `cartToPolar()`, `polarToCart()`

- The classical math functions.

- `scaleAdd()`, `transpose()`, `gemm()`, `invert()`, `solve()`, `determinant()`, `trace()`, `eigen()`, `SVD`.

- The algebraic functions + SVD class.

- `dft()`, `idft()`, `dct()`, `idct()`

- Discrete Fourier and cosine transformations.

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

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

Implements the core of Levenberg-Marquardt optimization algorithm.
Simple Image Processing

- `filter2D()`  
- `sepFilter2D()`  
- `boxFilter()`  
- `GaussianBlur()`  
- `medianBlur()`  
- `bilateralFilter()`  
- `Sobel()`, `Scharr()`  
- `Laplacian()`  
- `erode()`, `dilate()`  

- Non-separable linear filter  
- Separable linear filter  
- Smooth the image with one of the linear or non-linear filters  
- Compute the spatial image derivatives  
- compute Laplacian: $\Delta I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$  
- Erode or dilate the image

Example. Filter image in-place with a 3x3 high-pass kernel (preserve negative responses by shifting the result by 128):

```cpp
filter2D(image, image, image.depth(), (Mat_<float>(3,3)<< -1, -1, -1, -1, 9, -1, -1, -1, -1), Point(1,1), 128);
```
## Image Conversions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>resize()</code></td>
<td>Resize image</td>
</tr>
<tr>
<td><code>getRectSubPix()</code></td>
<td>Extract an image patch</td>
</tr>
<tr>
<td><code>warpAffine()</code></td>
<td>Warp image affinely</td>
</tr>
<tr>
<td><code>warpPerspective()</code></td>
<td>Warp image perspective</td>
</tr>
<tr>
<td><code>remap()</code></td>
<td>Generic image warping</td>
</tr>
<tr>
<td><code>convertMaps()</code></td>
<td>Optimize maps for a faster remap() execution</td>
</tr>
</tbody>
</table>

Example. Decimate image by factor of $\sqrt{2}$:
Mat dst; resize(src, dst, Size(), 1./sqrt(2), 1./sqrt(2))

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cvtColor()</code></td>
<td>Convert image from one color space to another</td>
</tr>
<tr>
<td><code>threshold()</code></td>
<td>Convert grayscale image to binary image using a fixed or a variable threshold</td>
</tr>
<tr>
<td><code>adaptiveThreshold()</code></td>
<td></td>
</tr>
<tr>
<td><code>floodFill()</code></td>
<td>Find a connected component using region growing algorithm</td>
</tr>
<tr>
<td><code>integral()</code></td>
<td>Compute integral image</td>
</tr>
<tr>
<td><code>distanceTransform()</code></td>
<td>build distance map or discrete Voronoi diagram for a binary image.</td>
</tr>
<tr>
<td><code>watershed()</code></td>
<td>marker-based image segmentation algorithms. See the samples watershed.cpp and grabcut.cpp.</td>
</tr>
<tr>
<td><code>grabCut()</code></td>
<td></td>
</tr>
</tbody>
</table>
Histograms

calcHist() Compute image(s) histogram
calcBackProject() Back-project the histogram
equalizeHist() Normalize image brightness and contrast
compareHist() Compare two histograms

Example. Compute Hue-Saturation histogram of an image:
Mat hsv, H; MatND tempH;
cvtColor(image, hsv, CV_BGR2HSV);
int planes[]={0, 1}, hsize[] = {32, 32};
calcHist(&hsv, 1, planes, Mat(), tempH, 2, hsize, 0);
H = tempH;
I/O

Writing and reading raster images
```
imwrite("myimage.jpg", image);
Mat image_color_copy = imread("myimage.jpg", 1);
Mat image_grayscale_copy = imread("myimage.jpg", 0);
```

The functions can read/write images in the following formats:
- BMP (.bmp)
- JPEG (.jpg, .jpeg)
- TIFF (.tif, .tiff)
- PNG (.png)
- PBM/PGM/PPM (.p?m)
- Sun Raster (.sr)
- JPEG 2000 (.jp2)

Every format supports 8-bit, 1- or 3-channel images. Some formats (PNG, JPEG 2000) support 16 bits per channel.

Reading video from a file or from a camera
```
VideoCapture cap;
if(argc > 1) cap.open(string(argv[1])); else cap.open(0);
Mat frame; namedWindow("video", 1);
for(;;) {
    cap >> frame; if(!frame.data) break;
    imshow("video", frame); if(waitKey(30) >= 0) break;
}
Serialization I/O

Data I/O
XML/YAML storage are collections (possibly nested) of scalar values, structures and heterogeneous lists.

Writing data to YAML (or XML)

```cpp
// Type of the file is determined from the extension
FileStorage fs("test.yml", FileStorage::WRITE);
fs << "i" << 5 << "r" << 3.1 << "str" << "ABCDEFGH";
fs << "mtx" << Mat::eye(3,3,CV_32F);
fs << "mylist" << "[" << CV_PI << "1+1" <<
   "{" << "month" << 12 << "day" << 31 << "year"
   << 1969 << "}" << "]";
fs << "mystruct" << "{" << "x" << 1 << "y" << 2 <<
   "width" << 100 << "height" << 200 << "lbp" << "[:";
const uchar arr[] = {0, 1, 1, 0, 1, 1, 0, 1};
fs.writeRaw("u", arr, (int)(sizeof(arr)/sizeof(arr[0])));
fs << "]" << "]";
```

Scalars (integers, floating-point numbers, text strings), matrices, STL vectors of scalars and some other types can be written to the file storages using `<<` operator.
Serialization I/O

Reading the data back

// Type of the file is determined from the content
FileStorage fs("test.yml", FileStorage::READ);
int i1 = (int)fs["i"]; double r1 = (double)fs["r"];
string str1 = (string)fs["str"];
Mat M; fs["mtx"] >> M;
FileNode tl = fs["mylist"];
CV_Assert(tl.type() == FileNode::SEQ && tl.size() == 3);
double t10 = (double)tl[0]; string t11 = (string)tl[1];
int m = (int)tl[2]["month"], d = (int)tl[2]["day"];
int year = (int)tl[2]["year"];
FileNode tm = fs["mystruct"];
Rect r; r.x = (int)tm["x"], r.y = (int)tm["y"],
r.width = (int)tm["width"], r.height = (int)tm["height"];
int lbp_val = 0;
FileNodeIterator it = tm["lbp"].begin();
for(int k = 0; k < 8; k++, ++it)
    lbp_val |= ((int)*it) << k;

Scalars are read using the corresponding FileNode’s cast
operators. Matrices and some other types are read using >>
operator. Lists can be read using FileNodeIterator’s.
GUI ("HighGUI")

namedWindow(winname, flags) Create named highgui window
destroyWindow(winname) Destroy the specified window
imshow(winname, mtx) Show image in the window
waitKey(delay) Wait for a key press during the specified time interval (or forever). Process events while waiting. *Do not forget to call this function several times a second in your code.*
createTrackbar(...) Add trackbar (slider) to the specified window
setMouseCallback(...) Set the callback on mouse clicks and movements in the specified window

See camshiftdemo.c and other OpenCV samples on how to use the GUI functions.
Camera Calibration, Pose, Stereo

calibrateCamera() Calibrate camera from several views of a calibration pattern.

findChessboardCorners() Find feature points on the checkerboard calibration pattern.

solvePnP() Find the object pose from the known projections of its feature points.

stereoCalibrate() Calibrate stereo camera.

stereoRectify() Compute the rectification transforms for a calibrated stereo camera.

initUndistortRectifyMap() Compute rectification map (for remap()) for each stereo camera head.

StereoBM, StereoSGBM The stereo correspondence engines to be run on rectified stereo pairs.

reprojectImageTo3D() Convert disparity map to 3D point cloud.

findHomography() Find best-fit perspective transformation between two 2D point sets.

To calibrate a camera, you can use calibration.cpp or stereo_calib.cpp samples. To get the disparity maps and the point clouds, use stereo_match.cpp sample.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>matchTemplate</td>
<td>Compute proximity map for given template.</td>
</tr>
<tr>
<td>CascadeClassifier</td>
<td>Viola’s Cascade of Boosted classifiers using Haar or LBP features. Suits for detecting faces, facial features and some other objects without diverse textures. See facedetect.cpp</td>
</tr>
<tr>
<td>HOGDescriptor</td>
<td>N. Dalal’s object detector using Histogram-of-Oriented-Gradients (HOG) features. Suits for detecting people, cars and other objects with well-defined silhouettes. See peopledetect.cpp</td>
</tr>
</tbody>
</table>
samples/c

bgfg_codebook.cpp - Use of a image value codebook for background detection for collecting 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
detection.c
edge.c - Discrete fourier transform
disttrans.c - distance map from edges in an image
drawing.c - Various drawing functions
delaunay.c - Edge detection
detect.c - Face detection by classifier cascade
detector.c - Flood filling demo
dfind_obj.cpp - Demo use of SURF features
dfitellipse.c - Robust ellipse fitting
dhoughlines.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.
morphology.c - Shows use of Cv_SEQ
motempl.c - Demonstrates Erode, Dilate, Open, Close
mushroom.cpp - Demonstrates motion templates (orthogonal optical flow given silhouettes)
pyramid_segmentation.c - Demonstrates use of decision trees (CART) for recognition
squares.c - Color segmentation in pyramid
stereo_calib.cpp - Uses contour processing to find squares in an image
watershed.cpp - Stereo calibration, recognition and disparity map computation

- Watershed transform demo.
samples/C++

- build3dmodel.cpp
- calibration.cpp
- connected_components.cpp
- contours2.cpp
- descriptor_extractor_matcher.cpp
- fern_params.xml
- generic_descriptor_match.cpp
- matcher_simple.cpp
- morphology2.cpp
- segment_objects.cpp
- select3dobj.cpp
Samples/python

- camera.py
- camshift.py
- chessboard.py
- contours.py
- convexhull.py
- cv20squares.py
- cvutils.py
- delaunay.py
- demhist.py
- dft.py
- distrans.py
- dmtx.py
- drawing.py
- edge.py
- facedetect.py
- fback.py
- ffldemo.py
- fitellipse.py
- houghlines.py
- inpaint.py
- kalman.py
- kmeans.py
- laplace.py
- lkdemo.py
- logpolar.py
- minarea.py
- minidemo.py
- morphology.py
- motempl.py
- numpy_array.py
- numpy_warhol.py
- peopledetect.py
- pyramid_segmentation.py
- squares.py
- watershed.py
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 display 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_18.cpp  Reading an XML file
ch3_ex3_20.cpp  How to check if IPP acceleration is on
Book Examples

ch4_ex4_1.cpp
ch4_ex4_2.cpp
ch4_ex4_3.cpp

Use a mouse to draw boxes
Use a trackbar as a button
Finding the video codec

ch5_ex5_1.cpp
ch5_ex5_2.cpp
ch5_ex5_3.cpp
ch5_ex5_4.cpp

Using CvSeq
cvThreshold example
Combining image planes
Adaptive thresholding

ch6_ex6_1.cpp
ch6_ex6_2.cpp
ch6_ex6_3.cpp
ch6_ex6_4.cpp
ch6_ex6_5.cpp

cvHoughCircles example
Affine transform
Perspective transform
Log-Polar conversion
2D Fourier Transform

ch7_ex7_1.cpp
ch7_ex7_2.txt
ch7_ex7_3_expanded.cpp
ch7_ex7_4.txt
ch7_ex7_5.cpp
ch7_ex7_5_HistBackProj.cpp

Using histograms
Earth Mover’s Distance interface
Earth Mover’s Distance set up
Using Earth Mover’s Distance
Template matching/Cross Corr.
Back projection of histograms

ch8_ex8_1.txt
ch8_ex2.cpp
ch8_ex8_2.cpp
ch8_ex8_3.cpp

CvSeq structure
Contour structure
Finding contours
Drawing contours
Book Examples

ch9_ex9_1.cpp
ch9_watershed.cpp
ch9_AvgBackground.cpp
ch9_backgroundAVG.cpp
    average
ch9_backgroundDiff.cpp
ch9_ClearStaleCB_Entries.cpp
cv_yuv_codebook.cpp

ch10_ex10_1.cpp
ch10_ex10_1b_Horn_Schunck.cpp
ch10_ex10_2.cpp
ch10_motempl.cpp

ch11_ex11_1.cpp
camera
ch11_ex11_1_fromdisk.cpp
ch11_chessboards.txt

ch12_ex12_1.cpp
ch12_ex12_2.cpp
ch12_ex12_3.cpp
ch12_ex12_4.cpp
ch12_list.txt

ch13_dtree.cpp
ch13_ex13_1.cpp
ch13_ex13_2.cpp
ch13_ex13_3.cpp
ch13_ex13_4.cpp
cvx_defs.cpp

Sampling from a line in an image
Image segmentation using Watershed transform
Background model using an average image
Background averaging using a codebook compared to just an average
Use the codebook method for doing background differencing
Refine codebook to eliminate stale entries
Core code used to design OpenCV codebook

Optical flow using Lucas-Kanade in an image pyramid
Optical flow based on Horn-Schunck block matching
Kalman filter example code
Using motion templates for segmenting motion.

Camera calibration using automatic chessboard finding using a camera
Doing the same, but read from disk
List of included chessboards for calibration from disk example

Creating a bird’s eye view of a scene using homography
Computing the Fundamental matrix using RANSAC
Stereo calibration, rectification and correspondence
2D robust line fitting
List of included stereo L+R image pair data

Example of using a decision tree
Using k-means
Creating and training a decision tree
Training using statistical boosting
Face detection using Viola-Jones
Some defines for use with codebook segmentatio
#!/usr/bin/python

""
This program is demonstration python ROS Node for face and object detection using haar-like features.
The program finds faces in a camera image or video stream and displays a red box around them. Python implementation by: Roman Stanchak, James Bowman
""

import roslib
roslib.load_manifest('opencv_tests')
import sys
import os
from optparse import OptionParser
import rospy
import sensor_msgs.msg
from cv_bridge import CvBridge
import cv

# Parameters for haar detection
# From the API:
# The default parameters (scale_factor=2, min_neighbors=3, flags=0) are tuned
# for accurate yet slow object detection. For a faster operation on real video
# images the settings are:
# scale_factor=1.2, min_neighbors=2, flags=CV_HAAR_DO_CANNY_PRUNING,
# min_size=<minimum possible face size

min_size = (20, 20)
image_scale = 2
haar_scale = 1.2
min_neighbors = 2
haar_flags = 0
Python Face Detector Node: 2

The Core

```python
if __name__ == '__main__':
    pkgdir = roslib.packages.get_pkg_dir('opencv2')
    haarfile = os.path.join(pkgdir, "opencv/share/opencv/haarcascades/haarcascade_frontalface_alt.xml")

    parser = OptionParser(usage = "usage: %prog [options] [filename|camera_index]")
    parser.add_option("-c", "--cascade", action="store", dest="cascade", type="str", help="Haar cascade file, default %default", default = haarfile)
    (options, args) = parser.parse_args()

    cascade = cv.Load(options.cascade)
    br = CvBridge()

    def detect_and_draw(imgmsg):
        img = br.imgmsg_to_cv(imgmsg, "bgr8")
        # allocate temporary images
        gray = cv.CreateImage((img.width,img.height), 8, 1)
        small_img = cv.CreateImage((cv.Round(img.width / image_scale),
                                   cv.Round(img.height / image_scale)), 8, 1)

        # convert color input image to grayscale
        cv.CvtColor(img, gray, cv.CV_BGR2GRAY)

        # scale input image for faster processing
        cv.Resize(gray, small_img, cv.CV.INTER_LINEAR)

        cv.EqualizeHist(small_img, small_img)
        if(cascade):
            faces = cv.HaarDetectObjects(small_img, cascade, cv.CreateMemStorage(0),
                                          haar_scale, min_neighbors, haar_flags, min_size)
            if faces:
                for ((x, y, w, h), n) in faces:
                    # the input to cv.HaarDetectObjects was resized, so scale the
                    # bounding box of each face and convert it to two CvPoints
                    pt1 = (int(x * image_scale), int(y * image_scale))
                    pt2 = (int((x + w) * image_scale), int((y + h) * image_scale))
                    cv.Rectangle(img, pt1, pt2, cv.RGB(255, 0, 0), 3, 8, 0)
        cv.ShowImage("result", img)
        cv.WaitKey(6)

    rospy.init_node('rosfacedetect')
    image_topic = rospy.resolve_name("image")
    rospy.Subscriber(image_topic, sensor_msgs.msg.Image, detect_and_draw)
    rospy.spin()
```

Outline

• OpenCV Overview
• Cheatsheet
• Simple Programs
• Tour
• Features2D
• Applications

Gary Bradski, 2009
New C++ API: Usage Example

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];
}
/*
 * Make an image pyramid with levels of arbitrary scale reduction (0,1)
 * M     Input image
 * reduction Scaling factor 1>reduction>0
 * levels How many levels of pyramid
 * pyr   std vector containing the pyramid
 * sz    The width and height of blurring kernel, DEFAULT 3
 * sigma The standard deviation of the blurring Gaussian DEFAULT 0.5
 * RETURNS Number of levels achieved
 */

int buildGaussianPyramid(const Mat &M, double reduction, int levels, vector<Mat> &pyr, int sz = 3, float sigma = 0.5) {
    if(M.empty()) return 0;
    pyr.clear(); //Clear it up
    if((reduction <= 0.0) || (reduction >= 1.0)) return 0;
    Mat Mblur, Mdown = M;
    pyr.push_back(Mdown);
    Size ksize = Size(sz, sz);
    int L = 1;
    for(; L <= levels; ++L) {
        if((reduction * Mdown.rows) <= 1.0 || (reduction * Mdown.cols) <= 1.0) break;
        GaussianBlur(Mdown, Mblur, ksize, sigma, sigma);
        resize(Mblur, Mdown, Size(), reduction, reduction);
        pyr.push_back(Mdown);
    }
    return L;
}
Outline

• OpenCV Overview
• Cheatsheet
  • Simple Programs
• Tour
• Features2D
• Applications
Canny Edge Detector
Distance Transform

• Distance field from edges of objects

Flood Filling
Space Variant vision: Log-Polar Transform

Screen shots by Gary Bradski, 2005
void cvPyrDown(
    IplImage* src,
    IplImage* dst,
    IplFilter filter = IPL_GAUSSIAN_5x5);

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

Screen shots by Gary Bradski, 2005
Histogram Equalization

Low Dynamic Range Image and its Histogram

Histogram Equalized Image and its Histogram

Screen shots by Gary Bradski, 2005
Contours
Morphological Operations Examples

- Morphology - applying Min-Max Filters and its combinations

\[
\begin{align*}
\text{Image } I &= \text{Original Image} \\
\text{Erosion } I \Theta B &= \text{Eroded Image} \\
\text{Dilatation } I \oplus B &= \text{Dilated Image} \\
\text{Opening } I \Theta B &= (I \Theta B) \oplus B \\
\text{Closing } I \Theta B &= (I \Theta B) \Theta B \\
\text{Grad}(I) &= (I \oplus B) - (I \Theta B) \\
\text{TopHat}(I) &= I - (I \Theta B) \\
\text{BlackHat}(I) &= (I \oplus B) - I
\end{align*}
\]
Image textures

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

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

Screen shots by Gary Bradski, 2005
Recent Algorithms: GrabCut

- Graph Cut based segmentation

Images by Gary Bradski, © 2010
Motion Templates (work with James Davies)

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

Charts by Gary Bradski, 2005
Segmentation, Motion Tracking and Gesture Recognition
New Optical Flow Algorithms

```
// 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)

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

\[
G \cdot \partial X = b,
\]

\[
\partial X = (\partial x, \partial y), G = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, b = \sum I_t \begin{bmatrix} I_x \\ I_y \end{bmatrix}
\]
Tracking with CAMSHIFT

• Control game with head

*Screen shots by Gary Bradski, 2005*
Projections

**Affine (2x2)**
- Parallelograms

**Perspective (3x3)**
- Trapazoids
  - (Includes all of Affine)

*Screen shots by Gary Bradski, 2005*
Stereo ... Depth from Triangulation

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

\[
d = x' - x^r
\]

Depth “Z” and disparity “d” are inversly related:
Stereo

• In aligned stereo, depth is from similar triangles:

\[
\frac{T - (x' - x'')}{Z - f} = \frac{T}{Z} \Rightarrow Z = \frac{fT}{x' - x''}
\]

• Problem: Cameras are almost impossible to align
• Solution: Mathematically align them:
Stereo Rectification

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

- OpenCV Overview
- Cheatsheet
- Simple Programs
  - Tour
- Features2D
- Applications
Features2d contents

**Detection**
- SIFT
- SURF
- FAST
- STAR
- MSER
- HARRIS
- GFTT (Good Features To Track)

**Description**
- SIFT
- SURF
- Calonder
- Ferns
- One way

**Matching**
- Matchers available
  - BruteForce
  - FlannBased
  - BOW
- Matches filters (under construction)
  - Cross check
  - Ratio check
Detector interfaces

class FeatureDetector
{
public:
    virtual ~FeatureDetector() {}

    // Detect keypoints in an image.
    virtual void detect( const Mat& image, vector<KeyPoint>& keypoints,
                          const Mat& mask=Mat() ) const = 0;

    // Detect keypoints in an image set.
    void detect( const vector<Mat>& imageCollection,
                 vector<vector<KeyPoint>>& pointCollection,
                 const vector<Mat>& masks=vector<Mat>() ) const;

    virtual void read( const FileName& fn ) {}
    virtual void write( FileStorage& fs ) const {}

protected:
    ...
};
Creating a detector

• Statically
  SurfFeatureDetector detector;

• Using class factory
  cv::Ptr<FeatureDetector> detector =
  createFeatureDetector("SURF");
Running detector

Mat img = imread( "test.png" );
vector<KeyPoint> keypoints;

SurfFeatureDetector detector;
detector.detect( img, keypoints );
Descriptor interfaces

• For descriptors that can be represented as vectors in multidimensional space:
  
  DescriptorExtractor and DescriptorMatcher

• More general interface (one way, decision-tree-based descriptors):
  
  GenericDescriptorMatcher
class CV_EXPORTS DescriptorExtractor
{
public:
    virtual ~DescriptorExtractor() {}
    // Compute the descriptors for a set of keypoints in an image.
    virtual void compute( const Mat& image, vector<KeyPoint>& keypoints,
                          Mat& descriptors ) const = 0;
    // Compute the descriptors for a keypoints collection detected in image collection.
    void compute( const vector<Mat>& imageCollection,
                  vector<vector<KeyPoint>>& pointCollection,
                  vector<Mat>& descCollection ) const;

    virtual void read( const FileNode& ) {};
    virtual void write( FileStorage& ) const {};
    virtual int descriptorSize() const = 0;
    virtual int descriptorType() const = 0;

protected:
    ...
};
DescriptorExtractor creating

- Statically
  SurfDescriptorExtractor descriptorExtractor;
- Using class factory
  cv::Ptr<DescriptorExtractor> descriptorExtractor =
createDescriptorExtractor("SURF");
Ptr<FeatureDetector> detector = createFeatureDetector("FAST");
Ptr<DescriptorExtractor> descriptorExtractor = createDescriptorExtractor("SURF");

vector<KeyPoint> keypoints;
detector->detect( img, keypoints );
Mat descriptors;
descriptorExtractor->compute( img, keypoints, descriptors );
DescriptorMatcher interfaces

• Two groups of match methods
  – to match descriptors of image pair
  – to match descriptors of one image to image set

• Each group consists from tree type methods
  – match()
  – knnMatch()
  – radiusMatch()
Matching of image pair

// detecting keypoints
SurfFeatureDetector detector;
vector<KeyPoint> keypoints1, keypoints2;
detector.detect( img1, keypoints1 );
detector.detect( img2, keypoints2 );

// computing descriptors
SurfDescriptorExtractor extractor;
Mat descriptors1, descriptors2;
extractor.compute( img1, keypoints1, descriptors1 );
extractor.compute( img2, keypoints2, descriptors2 );

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

Mat img_points;
drawKeypoints( img, keypoints, img_points );
namedWindow( "keypoints", 1 );
imshow( "keypoints", img_points );
waitKey();

Visualize matches

Mat img_matches;
drawMatches( img1, keypoints1, img2, keypoints2, img_matches);
namedWindow( "matches", 1 );
imshow( "matches", img_matches );
waitKey();
Running the sample

- Download OpenCV from TBD link
- Compile
- Run matcher_simple:
  ```
  bin/matcher_simple ../../opencv/samples/c/box.png ../../opencv/samples/c/box_in_scene.png
  ```
- Select a detector that gives the maximum number of keypoints
- Switch SIFT and SURF descriptors
Cross-check outlier match filtering

```cpp
BruteForceMatcher<L2<float> > descriptorMatcher;
vector<DMatch> filteredMatches12, matches12, matches21;
descriptorMatcher.match( descriptors1, descriptors2, matches12 );
descriptorMatcher.match( descriptors2, descriptors1, matches21 );

for( size_t i = 0; i < matches12.size(); i++ )
{
    DMatch forward = matches12[i];
    DMatch backward = matches21[forward.trainIdx];
    if( backward.trainIdx == forward.queryIdx )
        filteredMatches12.push_back( forward );
}
```
Ratio test to filter matches

\[ \text{Ratio} = \frac{\text{MinDist}_1}{\text{MinDist}_2} \in (0,1] \]  
(less is better)

if \( \text{Ratio < threshold (0.3)} \Rightarrow \text{inlier, else outlier} \)
Calculating inliers (planar objects case)

- Detect keypoints
- Find matches using descriptors
- Filter matches using cross-check
- Calculate best homography
- Filter outliers
- Run
  
  $\text{bin/descriptor_extractor_matcher \ SURF \ SURF \ ..../../
  \opencv/samples/c/box.png \ ../..\opencv/samples/c/
  \ box\(_{-}\)in\(_{-}\)scene.png \ 3$

  The last parameter is the reprojection threshold for RANSAC
• Measures of detector repeatability are taken from

• Test images are taken from
  http://www.robots.ox.ac.uk/~vgg/data/data-aff.html

• Testbench is located in
  opencv_extra/testdata/cv/detectors_descriptors_evaluation/detectors
Descriptor testbench

• Measures of descriptor matching accuracy are taken from http://www.robots.ox.ac.uk/~vgg/research/affine(det_eval_files/mikolajczyk_pami2004.pdf

• Test images are taken from http://www.robots.ox.ac.uk/~vgg/data/data-aff.html

• Testbench is located in opencv_extra/testdata/cv/detectors_descriptors_evaluation/descriptors
OpenCV and ROS

• Opencv2 package to fetch and compile opencv

• Messages:
  – sensor_msgs::Image
  – sensor_msgs::CameraInfo

• cv_bridge to convert between messages and images

• image_geometry::PinholeCameraModel and image_geometry::StereoCameraModel to manage 2d <-> 3d conversions
Q&A

• Foils will be available at http://itseez.com