OpenCV 2.4 Cheat Sheet (C++)

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

Key OpenCV Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point_</td>
<td>2D point class</td>
</tr>
<tr>
<td>Point3_</td>
<td>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>Mat</td>
<td>Template small matrix class</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>

SparseMat

Multi-dimensional sparse array

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix Basics</td>
<td></td>
</tr>
</tbody>
</table>

Create a matrix

- Mat image(240, 320, CV_8UC3);
- [R]allocate a pre-declared matrix
- image.create(480, 640, CV_8UC3);

Create a matrix initialized with a constant

- Mat A33(3, 3, CV_32F, Scalar(S));
- Mat B33(3, 3, CV_32F); B33 = Scalar(S);
- 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 I1(ipitvec); // I – 10x1 CV_32SC2 matrix
- IplImage* oldDC = cvCreateImage(cvSize(320, 240),16,1);
- Mat neuC = cvMatToMat(oldDC);
- IplImage oldDC = newC; CvMat oldDC = newC;
- (without copying the data)
- Mat neuC = cvMatToMat(oldDC).clone();
- vector<Point2f> ptvec = Mat_<Point2f>(I1.1);

Access matrix elements

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A33.at&lt;float&gt;(i,j) = A33.at&lt;float&gt;(j,i)+1;</td>
<td></td>
</tr>
</tbody>
</table>

Matrix Manipulations: Copying, Shuffling, Part Access

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

Example 1. Smooth image ROI in-place

- Mat imgray = image(Rect(10, 20, 100, 100));
- GaussianBlur(imgray, imgray, 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 ROI image into another image with conversion

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

Simple Matrix Operations

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

- add(), subtract(), multiply(), divide(), addSdiff(), 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.

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

Mat delta = (J.t() * lambdam * Mat::eye(J.cols, J.cols).inv(CV_SVD)).t();

implements the core of Levenberg-Marquardt optimization algorithm.

Image Processing

Filtering

- filter2D() Non-separable linear filter
- sepFilter2D() Separable linear filter
- Smooth the image with one of the linear or non-linear filters
- boxFilter(), GaussianBlur(), medianBlur(), bilateralFilter()
- Sobel(), Scharr()
- Compute the spatial image derivatives
- Laplacian() and
- erode(), dilate() Morphological operations
Example. Filter image in-place with a 3x3 high-pass kernel (preserve negative responses by shifting the result by 128):
filter2D(image, image, image.depth(), (Mat_<float>(3,3)« -1, -1, -1, -1, -1, -1, -1, -1, -1), Point(1,1,128);

Geometrical Transformations

resiz() // Resize image
getRectSubPix() // Extract an image patch
warpAffine() // Warp image affinely
warpPerspective() // Warp image perspective
remap() // Generic image warping
convertMaps() // Optimize maps for a faster remap() execution

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

Various Image Transformations
cvtColor() // Convert image from one color space to another
threshold() // Convert grayscale image to binary image
adaptiveThreshold() // using a fixed or a variable threshold
floodFill() // Find a connected component using region growing algorithm
integral() // Compute integral image
distanceTransform() // build distance map or discrete Voronoi diagram for a binary image.
watershed() // marker-based image segmentation algorithms.
grabCut() // See the samples watershed.cpp and grabcut.cpp.

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;
cvtColor(image, hsv, CV_BGR2HSV);
Mat dst; resize(src, dst, Size(), 1./sqrt(2), 1./sqrt(2));

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

Reading the data back // Type of the file is determined from the content
FileStorage fs("test.yml", FileStorage::WRITE);
fs << "i" << 5 << "r" << 3.1 << "str" << "ABCDEFGH";
fs << "str" << Mat::eye(3,3,CV_32F);
fs << "mylist" << "[" << CV_PI << "=1"] << ";";
fs << "mystruct" << "[" << "x" << 1 << "y" << 2 << "width" << 100 << "height" << 200 << "lbp" << ";:
const uchar arr[] = {0, 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 op operator.

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", frame); if(waitKey(30) >= 0) break;

Writing data to YAML (or XML)
// Type of the file is determined from the extension

Example. Read/write images in the following formats:
- BMP (.bmp), JPEG (.jpg), PNG (.png), TIFF (.tif .tiff), PNM (.png), PBM/PGM/PPM (.p?.m), Sun Raster (.ar), JPEG 2000 (.jp2). Every format supports 8-bit, 1- or 3-channel images. Some formats (PNG, JPEG 2000) support 16 bits per channel.

The functions can read/write images:
Mat image_grayscale_copy = imread("myimage.jpg", 0);
Mat image_color_copy = imread("myimage.jpg", 1);
imwrite("myimage.jpg", image);

Camera Calibration, Pose Estimation and Depth Estimation
calibrateCamera() // Calibrate camera from several views of a pattern calibration.
findChessboardCorners() // Find feature points on the checker-board calibration pattern.
solvePnP() // Find the object pose from the known projections of its feature points.
StereoBM() // Calibrate stereo camera.
StereoRectify() // Compute the rectification transforms for a calibrated stereo camera.
initUndistortRectifyMap() // Compute rectification transforms for a stereo camera.
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.

Object Detection
cascadeClassifier // Compute proximity map for given template.

Violator'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

HOGDescriptor // 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

Simple GUI (highgui module)
namedWindow("winname_flag", Create named highgui window
destroyWindow("winname") // Destroy the specified window
showWindow("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.cpp and other OpenCV samples on how to use the GUI functions.