OpenCV 2.1 Cheat Sheet (C++)

The OpenCV C++ reference manual is here:
http://opencv.willowgarage.com/documentation/cpp/

Use Quick Search to find descriptions of the particular functions and classes.

Key OpenCV Classes

- Point3: Template 2D point class
- Point3f: Template 3D point class
- Size: Template size (width, height) class
- Vec: 4-element vector
- Scalar: Integer value range
- Rect: Rectangle
- Range: Image rows/cols range
- Mat: 2D dense array (used as both a matrix or an image)
- Mat32F: Multi-dimensional sparse array
- Mat32FC3: Multi-dimensional dense array
- SparseMat: Template smart pointer class
- Vec: Template short vector class
- Point: 2D dense array (used as both a matrix or an image)
- Point3: 3D dense array (used as both a matrix or an image)

Matrix Basics

Create a matrix
- Mat image(240, 320, CV_8UC3); // Creates a matrix image of size 240x320 with 8-bit color channels.
- [Re]allocate a pre-declared matrix
- image.create(480, 640, CV_8UC3); // Creates a matrix image of size 480x640 with 8-bit color channels.
- Create a matrix initialized with a constant
- Mat B33(3, 3, CV_32SF); // Creates a matrix B33 of size 3x3 with float elements.
- Create a matrix initialized with specified values
- double a = CV_PI/3; // Sets a to 1.0472.
- Mat A33 = (Mat::eye(3, 3, CV_32F) + 5.); // Creates a matrix A33 of size 3x3 with float elements, the 5th element is 5.

Initialize a random matrix
- randn(image, Scalar(0), 10); // uniform dist
- randn(image, Scalar(CV_8UC3)); // Gaussian dist

Convert matrix to/from other structures

- Convert matrix to/from other structures without copying the data
- cvtToMat: Mat image = cvtToMat(image, CV_32F); // Converts a CV_8UC3 matrix to a CV_32F matrix.
- Convert matrix to/from other structures with copying the data
- cvtToMat: Mat oldC0 = cvtToMat(oldC0, CV_32F); // Copies a CV_32F matrix to a new CV_32F matrix.

Access matrix elements

- A33.at<float>(i,j) = A33.at<float>(j,i)*1; // Accesses the element at row i, column j of matrix A33.
- Mat dyImage(image.size(), image.type()); // Creates a matrix dyImage of the same size and type as image.

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 dimensions matrices and/or number of channels without copying data
m.row(i).clone() // Take a row row/matrix
m.rowRange(Range(i1, i2)) // Take a matrix row/column span
m.col(i) // Take a column
m.colRange(Range(j1, j2)) // Take a submatrix
m.diag(i) // Take a submatrix diagonal
m.col2row() // Take a column to row vector
m.transpose() // Transpose the matrix
m.trace() // Trace of the matrix
m.sum() // Sum of elements of the matrix
m.determinant() // Determinant of the matrix
m.inv() // Inverse of the matrix
m.log() // Natural logarithm of each element
m.sqrt() // Square root of each element
m.exp() // Exponential of each element
m.pow() // Power of each element
m.mean() // Mean of elements of the matrix
m.meanStdDev() // Standard deviation of each column of the matrix
m.minMaxLoc() // Minimum and maximum elements of the matrix
m.meanStdDev() // Mean and standard deviation of each column of the matrix
m.epsilon() // Point-wise comparison of two matrices or a matrix and a scalar.

Example. Alpha compositing function:
void alphaCompose(const Mat& rgba1,
const Mat& rgba2, Mat& rgba_dest)
{
Mat oldC1 = newC; CvMat oldC2 = newC;
for(int y = 0; 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<unsigned char>(
  nextRow[x][c] - prevRow[x][c]);
}
Mat newC = cvarrToMat(oldC0).clone(); // Create a new matrix from the old one.
}

Simple Matrix Operations

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

- add(), subtract(), multiply(), divide(), absdiff(), bitwise_and(), bitwise_or(), bitwise_xor(), max(), min(), compare()
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), 0, -1, -1, -1, 1, 1, 1, -1, -1, 1, 1, Point(1, 1), 128);

Geometrical Transformations

resize() 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(), grabCut() Marker-based image segmentation algorithms. 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; MatND temp;
cvtColor(image, hsv, CV_BGR2HSV);
int planes[] = {0, 1, 2}, hsvSize = {32, 32},
calcHist(&hsv, 1, planes, Mat(), tempH, 2, hsize, 0);
H = tempH;

Contours
See contours.cpp and squares.c samples on how the contours and how to use them.

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

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

FileStorage fs("test.yml", FileStorage::WRITE);
fs << "i" << 5 << "w" << 3.1 << "stx" << "ABCDEFGH";
fs << "mtx" << Mat::eye(3, 3, CV_32F);
fs << "mylist" << "[" << "CV_PI" << "%i" << "[";
fs << "% month" << 12 << "% day" << 31 << "% year" << 1969 << "]" << "]";
fs << "mystruct" << "[" << "x" << 1 << "y" << 2 << "w" << 100 << "height" << 200 << "lbp" << "[:;
const uchar arr[] = {0, 1, 0, 1, 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

Reading the data back
FileStorage fs("test.yml", FileStorage::READ);
int i = (int)fs["i"];
double r = (double)fs["r"];
Mat M; fs["mtx"] >> M;

Camera Calibration, Pose Estimation and Depth Estimation
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.
Calibrate stereo camera.
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-fist 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
matchTemplate Compute proximity map for given template.
CascadeClassifier Viola’s Cascade of Boosted classifiers using Haar or LBP features. Suits for detecting faces, facial features and some other objects without diverse textures.
HOGDescriptor N. Dalal’s 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
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.