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>Matx</td>
<td>Template small matrix 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 or 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>

Matrix Basics

Create a matrix

Mat image(240, 320, CV_8UC3);

Mat reshape(nc,nrows) Change matrix dimensions and/or number of channels without copying data

Mat m(roi) Make a bigger matrix from a smaller one

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 datatpe

Matrix Operations

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

Example 1. Smooth image ROI in-place

Mat imROI = image(Rect(10, 20, 100, 200));
GaussianBlur(imROI, imROI, 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 r1(1, 1, 10, 20);
Mat dstROI = dst(Rect(0, 10, r1.width, r1.height));
src(r1).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(), 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:

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

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

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

Image Processing

Filtering

filter2D() Non-separable linear filter
sepFilter2D() Separable linear filter
boxFilter() Smooth the image with one of the linear or non-linear filters
GaussianBlur() Compute the spatial image derivatives
medianBlur() compute Laplacian: $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
Sobel(), Scharr() Morphological operations
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, -1, -1, -1, -1, -1), Point(1,1), 128);
```

### Geometrical Transformations
- `resize()`: Resize image
- `getRectSubPix()`: Extract an image patch
- `warpAffine()`: Warp image affine
- `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 using a fixed or a variable threshold
- `adaptiveThreshold()`: 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 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:
```
planes[] = {0, 1}, hsize[] = {32, 32};
cvtColor(image, hsv, CV_BGR2HSV);
Mat hsv, H;
Example. Compute proximity map for given template:
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 (.ar), JPEG 2000 (.jp2).

### Simple GUI (highgui module)
```cpp
namedWindow(winname, flag);
createTrackbar(\ldots) Add trackbar (slider) to the specified window
setMouseCallback(\ldots) Set the callback on mouse clicks and movements in the specified window
```

### 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.
- `calibrateCamera()`: 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.

### 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.
- `faceDetect.cpp`: Dalal’s object detector using Histogram-of-Oriented-Gradients (HOG) features. Suits for detecting people, cars and other objects with well-defined silhouettes.
- `peopledetect.cpp`