OpenCV User Guide

v2.2

December, 2010

Contents

Part I

C++ API Reference

Chapter 1

cv::Mat. Basic operations with images.

1.1 Basic operations with images

Input/Output

Load an image from a file:

```
Mat img = imread(filename);
```

If you read a jpg file, a 3 channel image is created by default. If you need a grayscale image, use:

```
Mat img = imread(filename, 0);
```

Save an image to a file:

```
Mat img = imwrite(filename);
```

Accessing pixel intensity values

In order to get pixel intensity value, you have to know the type of an image and the number of channels. Here is an example for a single channel grey scale image (type 8UC1) and pixel coordinates x and y:

```
Scalar intensity = img.at<uchar>(x, y);
```

`intensity.val[0]` contains a value from 0 to 255.

Now let us consider a 3 channel image with `bgr` color ordering (the default format returned by imread):

```
Vec3b intensity = img.at<Vec3b>(x, y);
uchar blue = intensity.val[0];
uchar green = intensity.val[1];
uchar red = intensity.val[2];
```

You can use the same method for floating-point images (for example, you can get such an image by running Sobel on a 3 channel image):

```
Vec3f intensity = img.at<Vec3f>(x, y);
float blue = intensity.val[0];
float green = intensity.val[1];
float red = intensity.val[2];
```

The same method can be used to change pixel intensities:

```
img.at<uchar>(x, y) = 128;
```

There are functions in OpenCV, especially from calib3d module, such as `projectPoints`, that take an array of 2D or 3D points in the form of `Mat`. Matrix should contain exactly one column, each row corresponds to a point, matrix type should be 32FC2 or 32FC3 correspondingly. Such a matrix can be easily constructed from std::vector:

```
vector<Point2f> points;
//... fill the array
Mat _points = Mat(points);
```

One can access a point in this matrix using the same method `Mat::at`:

```
Point2f point = _points.at<Point2f>(i, 0);
```