Image Processing
Robot Programming
Elective in Artificial Intelligence
Artificial Intelligence and Robotics

Domenico Daniele Bloisi
Contact

Domenico Daniele Bloisi, PhD
Assistant Professor

Location: DIAG A209

bloisi@dis.uniroma1.it
http://www.dis.uniroma1.it/~bloisi
Introduction

OpenCV has a modular structure

- core
- imgproc
- video
- calib3d
- features2d
- objdetect
- highgui
- gpu

Image Processing

Robot Programming

http://opencv.org/documentation.html
http://docs.opencv.org/2.4.7/modules/core/doc/intro.html
Image Processing

core - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.

imgproc - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
**Image Processing**

**features2d** - salient feature detectors, descriptors, and descriptor matchers.

**highgui** - an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities.
Image Processing

#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/features2d/features2d.hpp>
Data types

Set of primitive data types the library can operate on

- 8-bit unsigned integer (uchar)
- 8-bit signed integer (schar)
- 16-bit unsigned integer (ushort)
- 16-bit signed integer (short)
- 32-bit signed integer (int)
- 32-bit floating-point number (float)
- 64-bit floating-point number (double)
Mat A, C;  // creates just the header parts
A = imread(argv[1], CV_LOAD_IMAGE_COLOR);  // here we'll know the method used (allocate matrix)
Mat B(A);  // Use the copy constructor
C = A;  // Assignment operator

Mat D = A.clone();
Mat E;
A.copyTo(E);
Creating a Mat object

Mat M(2,2, CV_8UC3, Scalar(0,0,255));
cout << "M = " << endl << " " << M << endl << endl;

M.create(4,4, CV_8UC(2));
cout << "M = " << endl << " " << M << endl << endl;
MATLAB style initializer

Mat E = Mat::eye(4, 4, CV_64F);
cout << "E = " << endl << " " << E << endl << endl;

Mat O = Mat::ones(2, 2, CV_32F);
cout << "O = " << endl << " " << O << endl << endl;

Mat Z = Mat::zeros(3,3, CV_8UC1);
cout << "Z = " << endl << " " << Z << endl << endl;
How the image matrix is stored in the memory?

<table>
<thead>
<tr>
<th>Row</th>
<th>Column 0</th>
<th>Column 1</th>
<th>Column ...</th>
<th>Column m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 0</td>
<td>0,0</td>
<td>0,1</td>
<td>...</td>
<td>0, m</td>
</tr>
<tr>
<td>Row 1</td>
<td>1,0</td>
<td>1,1</td>
<td>...</td>
<td>1, m</td>
</tr>
<tr>
<td>Row ...</td>
<td>... ,0</td>
<td>... ,1</td>
<td>...</td>
<td>... , m</td>
</tr>
<tr>
<td>Row n</td>
<td>n,0</td>
<td>n,1</td>
<td>n,...</td>
<td>n, m</td>
</tr>
</tbody>
</table>

Note that the order of the channels is inverse: BGR instead of RGB.
How to scan Gray scale images

cv::Mat I = ...

...

for( int i = 0; i < I.rows; ++i) {
    for( int j = 0; j < I.cols; ++j) {
        uchar g = I.at<uchar>(i,j);
        ...
    }
}


How to scan RGB images

cv::Mat I = ... 

... 

for( int i = 0; i < I.rows; ++i) {
    for( int j = 0; j < I.cols; ++j) {
        uchar blue = I.at<cv::Vec3b>(i,j)[0];
        uchar green = I.at<cv::Vec3b>(i,j)[1];
        uchar red = I.at<cv::Vec3b>(i,j)[2];
        ...
    } 
}

Background Subtraction

How to Use Background Subtraction Methods

- Background subtraction (BS) is a common and widely used technique for generating a foreground mask (namely, a binary image containing the pixels belonging to moving objects in the scene) by using static cameras.

- As the name suggests, BS calculates the foreground mask performing a subtraction between the current frame and a background model, containing the static part of the scene or, more in general, everything that can be considered as background given the characteristics of the observed scene.

- Background modeling consists of two main steps:
  1. Background Initialization;
  2. Background Update.

In the first step, an initial model of the background is computed, while in the second step that model is updated in order to adapt to possible changes in the scene.

In this tutorial we will learn how to perform BS by using OpenCV. As input, we will use data coming from the publicly available data set Background Models Challenge (BMC).

Goals

In this tutorial you will learn how to:

1. Read data from videos by using VideoCapture or image sequences by using imread;
2. Create and update the background model by using BackgroundSubtractor class;
3. Get and show the foreground mask by using inRange.

http://docs.opencv.org/trunk/doc/tutorials/video/background_subtraction/background_subtraction.html