OpenCV Tutorial C++

Read & Display Image

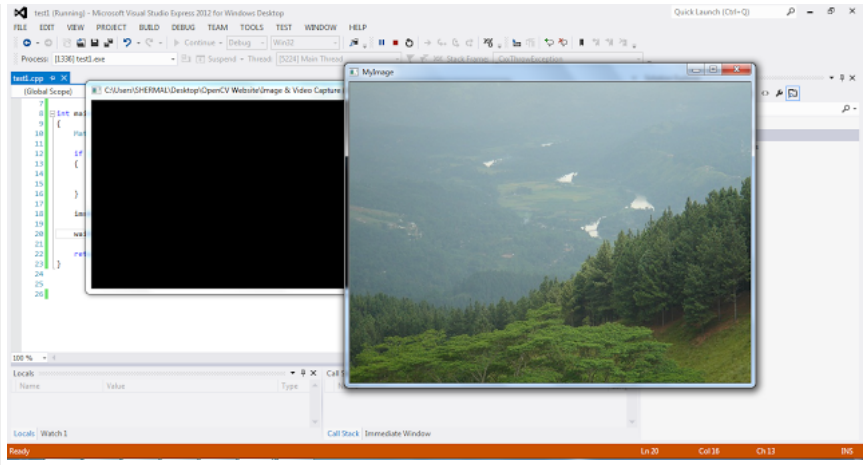
Read Image from File and Display

Here I am going to explain how to read an image from a file and display the content using OpenCV library functions. First of all, open your C++ IDE and create a new project. You have to configure your new project in order to use OpenCV library functions. If you have not configured the project for OpenCV yet, please refer to Installing & Configuring with Visual Studio.

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, const char** argv )
{
    Mat img = imread("MyPic.JPG", CV_LOAD_IMAGE_UNCHANGED); //read the image data in the file "MyPic.JPG" and store it in 'img'

    if (img.empty()) //check whether the image is loaded or not
    {
        cout << "Error : Image cannot be loaded..!!" << endl;
        //system("pause"); //wait for a key press
        return -1;
    }

    namedWindow("MyWindow", CV_WINDOW_AUTOSIZE); //create a window with the name "MyWindow"
    imshow("MyWindow", img); //display the image which is stored in the 'img' in the "MyWindow" window

    waitKey(0); //wait infinite time for a keypress

    destroyWindow("MyWindow"); //destroy the window with the name, "MyWindow"

    return 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Before you run this program, put any image file (MyPic.JPG) into the folder where your c++ file is.Otherwise you have to change the first argument of imread() function and give the absolute path to your image file.

You can download this OpenCV visual c++ project from here. (The downloaded file is a compressed .rar folder. So, you have to extract it using Winrar or other suitable software)



Load & Diplay Image from File

Explanation

Let's review the above OpenCV code line by line.

- **#include "stdafx.h"**

If you are developing your application in Visual Studio, don't forget to put this line above the code.

- **#include "opencv2/highgui/highgui.hpp"**

imread(), **namedWindow()**, **imshow()** and **waitKey()** functions are declared in the above header file. So you must include it.

We are using **Mat** data structure in the above program. It is declared in **"opencv2/core/core.hpp"** header file. Then why don't we include this? It's because **"opencv2/highgui/highgui.hpp"** header file include that header file inside it. So, we don't need to include it again in our program.

- **using namespace cv;**

All the data structures and functions in **"opencv2/core/core.hpp"** and **"opencv2/highgui/highgui.hpp"** are declared inside **cv** namespace. So, we have to add the above line in the top of our program. Otherwise we have to append '**cv::**' specifier before each OpenCV functions and data structures. (e.g - cv::Mat, cv::imread() , etc). If you are not familiar with namespaces, please refer this article.

'**#include <iostream>**' and '**using namespace std**' are added because we are using '**cout**' to display some strings in the console. This is basic C++ and you should be familiar with this.

- **Mat img = imread(**const string& **filename**, int **flags**=CV_LOAD_IMAGE_COLOR**)**

Mat is a data structure to store images in a matrix. It is declared in **"opencv2/core/core.hpp"** header file.

imread() is a function declared in **"opencv2/highgui/highgui.hpp"** header file. It loads an image from a file and stores it in Mat data structure.

Arguments of imread() function

- **filename** - location of the file. If you just give the filename only, that image should be in the same folder as your C++ file. Otherwise you have to give the full path to your image.
- **flags** - There are four possible inputs
  - CV_LOAD_IMAGE_UNCHANGED - image-depth=8 bits per pixel in each channel, no. of channels=unchanged
  - CV_LOAD_IMAGE_GRAYSCALE - image depth=8 bits, no. of channels=1
  - CV_LOAD_IMAGE_COLOR - image-depth=?, no. of channels=3
  - CV_LOAD_IMAGE_ANYDEPTH - image-depth=unchanged , no. of channels=?
  - CV_LOAD_IMAGE_ANYCOLOR - image-depth=?, no. of channels=unchanged

You can combine these above parameters to get desired image output.
e.g -
CV_LOAD_IMAGE_ANYDEPTH | CV_LOAD_IMAGE_ANYCOLOR - image-depth=unchanged, no. of channels=unchanged
CV_LOAD_IMAGE_COLOR | CV_LOAD_IMAGE_ANYDEPTH - image-depth=unchanged, no. of channels=3

If you are not sure what to do, use CV_LOAD_IMAGE_COLOR as the 2nd parameter of imread() function.

To understand image-depth and concept of channels, you should be familiar with theory of image processing. So, let's discuss little bit of theory of image processing.

Any digital image consists of pixels. Any pixel should have some value. The minimum value for a pixel is 0 and it represents black.When the value of the pixel is increased, the intensity of that pixel is also increased. In a computer memory,). In decimal, it is 255. fixed number of bits are allocated for every pixel. Say the number of allocated bits per pixel is 8. Then the maximum number that a pixel can have is 255 (11111111 in binary)

Now what is image-depth? The **image-depth** means the number of bits allocated for each pixel. If it is 8, each pixel can have a value between 0 and 255. If it is 4, each pixel can have a value between 0 to 15 (1111 in binary).

Here is a simple model of a image with image-depth of 8 bits. Each small box represents a pixel. So, each box may contain a value between 0 to 255.

Here is some properties of the following image.

- Image-depth 8 bit
- 1 channel (So, this is a grayscale image)
- The height is 4 pixel
- The width is 5 pixels
- The resolution of this image is 4x5.

This is a grayscale image (black and white image) because this image has no color content. If the value of this pixel is higher, it will be shown more brighter. If the value is low, it will be shown more darker.

23	23	34	255	0
78	245	129	25	251
23	12	89	90	37
84	26	47	127	199

Grayscale Image with image depth of 8

Following image is a simple model of a color image. Color image should consist of at least 3 planes; Red, Green and Blue. Any color can be created using a particular combination of these 3 colors. Any pixel is a combination of three 3 values. (255, 0, 0) represent pure red. (0, 255, 0) represent pure green. (255, 0, 255) represents pure violate. In the same way, you can create many color. Image-depth is 24 because each pixel is represented with 8 x 3 bits (8 bits from each channel).

Here is some properties of the following image.

- Image-depth 24 bit
- 3 channels (So, this is a color image)
- The height is 4 pixel
- The width is 5 pixels
- The resolution of this image is 4x5.

Red Plane					Green Plane					Blue Plane				
23	23	34	255	0	231	0	35	45	45	46	45	3	78	13
78	245	129	25	251	77	21	79	1	74	75	50	70	71	42
23	12	89	90	37	145	154	47	10	34	14	214	111	74	88
84	26	47	127	199	71	255	74	27	19	123	72	90	13	67

R, G, B planes of a color image

In the above model, top left pixel is (23, 231, 46). It will be shown as a greenish color because the green value(231) of that pixel is larger than the red(23) and blue(46) value.

- **if (img.empty())**

If **imread**() function fails to load the image, **'img'** will not be loaded any data. Therefore **'img.empty()'** should return true. It's a good practice to check whether the image is loaded successfully and if not exit the program. Otherwise your program will crash when executing **imshow()** function.

- **bool Mat::empty()**

This function returns true, if Mat::data==NULL or Mat::total() == 0

- **//system("pause");**

If you are using Visual Studio, it's better to uncomment this line because it will pause the program until user press any key. If we don't uncomment it, the program will exit immediately so that user will not see the error message.

- **void namedWindow(const string& winname, int flags = WINDOW_AUTOSIZE);**

This function creates a window.

Parameters -

- **winname** - Title of the window. That name will display in the title bar of the newly created window
- **flags** - determine the size of the window. There are two options

- **WINDOW_AUTOSIZE -** User cannot resize the image. Image will be displayed in its original size
- **CV_WINDOW_NORMAL -** Image will resized if you resize the the window

- **void imshow(const string& winname, InputArray mat);**

This function shows the image which is stored in the '**mat**' in a window specified by **winname**. If the window is created with **WINDOW_AUTOSIZE** flag, image will be displayed in its original size. Otherwise image may be scaled to the size of the window.

Parameters -

- **winname** - Title of the window. This name is used to identify the window created by namedWindow() function.
- **mat** - hold the image data

- **int waitKey(int delay = 0)**

waitKey() function wait for keypress for certain time, specified by **delay** (in milliseconds). If **delay** is zero or negative, it will wait for infinite time. If any key is pressed, this function returns the ASCII value of the key and your program will continue. If there is no key press for the specified time, it will return -1 and program will continue.

- **void destroyWindow(const string& winname)**

This function closes the opened window, with the title of **winname** and deallocate any associated memory usage. This function is not essential for this application because when the program exits, operating system usually close all the opened windows and deallocate any associated memory usage.

Summary

When running this program, the image of '**MyPic.JPG**' is loaded into the variable, '**img**' of type Mat. Then a window named '**MyWindow**' is opened. After that '**img**' is loaded to that window. The window with the image will be displayed until any key is pressed.

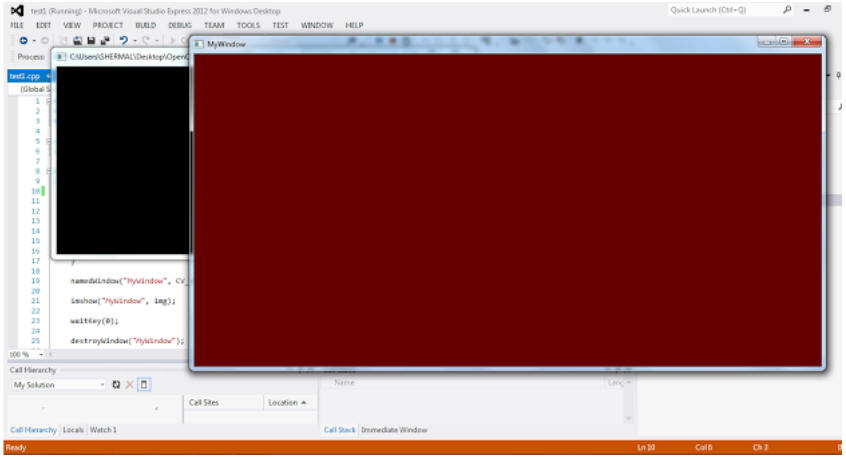
Create a Blank Image & Display

This program is also very much similar to the previous application. The only difference is that this program creates a blank image instead of loading an existing image from a file.

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, const char** argv )
{
    Mat img(500, 1000, CV_8UC3, Scalar(0,0, 100)); //create an image ( 3 channels, 8 bit image depth, 500 high, 1000 wide, (0, 0, 100)
assigned for Blue, Green and Red plane respectively. )

    if (img.empty()) //check whether the image is loaded or not
    {
        cout << "Error : Image cannot be loaded..!!" << endl;
        //system("pause"); //wait for a key press
        return -1;
    }

    namedWindow("MyWindow", CV_WINDOW_AUTOSIZE); //create a window with the name "MyWindow"
    imshow("MyWindow", img); //display the image which is stored in the 'img' in the "MyWindow" window

    waitKey(0);  //wait infinite time for a keypress

    destroyWindow("MyWindow"); //destroy the window with the name, "MyWindow"

    return 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Before you run this program, put any image file (MyPic.JPG) into the folder where your c++ file is.Otherwise you have to change the first argument of imread() function and give the absolute path to your image file.

You can download this OpenCV visual c++ project from here. (The downloaded file is a compressed .rar folder. So, you have to extract it using Winrar or other suitable software)



Create & Display Image

New OpenCV functions

- **Mat::Mat(int rows, int cols, int type, const Scalar& s);**

This is the one of the many constructor available in **Mat** class. It initialize the Mat object with the value given by the Scalar object

Parameters :

- **rows** - Number of rows in the 2D array (height of the image in pixels)
- **cols** - Number of columns in the 2D array (width of the image in pixels)
- **type** - specify the bit depth, data type and number of channels of the image. I gave **CV_8UC3** and it specify 8 bit unsigned integers with 3 channels. Here are some of possible inputs for this parameter
  - CV_8UC1 - 8 bit unsigned integers with single channel
  - CV_8UC3 - 8 bit unsigned integers with 3 channels
  - CV_64FC1 - 64 bit floating point value with 1 channels

If you want more details about this, please refer to **Data Types for Arrays** in the Basics of OpenCV API

- **s -** Initialize each array element with the value given by s. In the above application, I gave Scalar(0,0,100). So, it initialize my first channel (Blue plane) with 0, 2nd channel (Green plane) with 0 and 3rd channel (Red Plane) with 100. So, my final image is red. You can try different combinations of these three and see the output image.

Summary

In this program, I created a 3 channel image with 500 height and 1000 width. 8 bit unsigned integer is allocated for each pixel in each channel. (8x3 = 24 bits per each pixel) And each pixel is assigned with (0,0,100) scalar value. That means 1st channel is all zero, 2nd channel is also all zero and the 3rd channel is all 100. Therefore we can see a red image as the output of the program.

Next Tutorial : Video from File or Webcam

Previous Tutorial : Basics of OpenCV API

Posted by Shermal Fernando

94 comments:

Anonymous July 3, 2013 at 12:47 PM

great tutorial! thanks

Reply

Anonymous July 14, 2013 at 5:35 AM

really thank you a lot , great work :)

Reply

Anonymous July 17, 2013 at 6:11 PM

this is the best tutorial i have ever seen!Thank you so much!!!!FANTASY!

Reply