Introduction to OpenCv.

Introduction

This practical course is a short introduction to OpenCV Library. It is mainly based on the online tutorial `http://opencv-srf.blogspot.fr/` . If you are new to computer vision, you may be wondering where to start. First you have to understand the basic principles of image processing and computer vision. Then you have to choose a suitable language to develop your computer vision application. Some of the most popular methods are using OpenCV with $C/C++$ and MATLAB. If you don't really know why you would choose one over the other, here is an explanation.

MATLAB is the most easiest and the inefficient way to process images and OpenCV is the most efficient and hardest way to process images.

OpenCV has become hardest only because there is no proper documentation and error handling codes. But OpenCV has lots of basic inbuilt image processing functions so that those who want to learn computer vision can develop their applications through proper understanding about what they do.

So, It is worthy to learn computer vision with OpenCV. Therefore in this practical course, it is presented basic image processing functions and computer vision applications with line by line explanations.

a skeleton of a C++ Visual 2010 project is provided : you don't have to create a empty project from scratch.

1 What is OpenCv ?

OpenCV is an open source $C++$ library for image processing and computer vision, originally developed by Intel and now supported by Willow Garage. It is free for both commercial and non-commercial use. Therefore it is not mandatory for your OpenCV applications to be open or free. It is a library of many inbuilt functions mainly aimed at real time image processing. Now it has several hundreds of image processing and computer vision algorithms which make developing advanced computer vision applications easy and efficient.

Key Features are :

- Optimized for real time image processing and computer vision applications
- Primary interface of OpenCV is in $C++$
- There are also C, Python and JAVA full interfaces
- OpenCV applications run on Windows, Android, Linux, Mac and iOS
- Optimized for Intel processors

OpenCv is divided into main modules :

Core : This is the basic module of OpenCV. It includes basic data structures (e.g.- Mat data structure) and basic image processing functions. This module is also extensively used by other modules like highgui, etc.

Highgui : This module provides simple user interface capabilities, several image and video codecs, image and video capturing capabilities, manipulating image windows, handling track bars and mouse events and etc. If you want more advanced UI capabilities, you have to use UI frameworks like Qt, WinForms, etc. (e.g. - Load and Display Image, Capture Video from File or Camera, Write Image and Video to File)

imgproc : This module includes basic image processing algorithms including image filtering, image transformations, color space conversions and etc.

video : This is a video analysis module which includes object tracking algorithms, background subtraction algorithms and etc.

objdetect : This includes object detection and recognition algorithms for standard objects.

In the following, we consider that the library is correctly installed on you computer (PC-WIN, PC-LINUX or MAC) and that you correct compiling and building options have been set in the makefile (project parameters).

2 Read and display an image

the first program consists in reading and displaying an image.

```
////////////////////////////////////////////////////
#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.

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 :

- `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.

`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.

2.1 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.

2.2 Questions

1. test the above program by reading an JPG image.
2. modify this program to display an image uniform image with 3 channels, 8 bit image depth, 480 high, 640 wide, (0, 0, 50) assigned for Blue, Green and Red plane respectively. This can be done using a constructor of `Mat`.
3. modify this program to display an jpg image and a the same image but with half size. You may use the `resize` function

3 Read and Display a video

You just need to initialize a VideoCapture object which will open a video file and read frame by frame from that opened video. Here is the sample OpenCV code. If you are using Visual Studio, you will need to include "stdafx.h" header file.

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

using namespace cv;
using namespace std;

int main(int argc, char* argv[])
{
    VideoCapture cap("C:/Users/SHERMAL/Desktop/SampleVideo.avi");
    // open the video file for reading

    if ( !cap.isOpened() )  // if not success, exit program
    {
         cout << "Cannot open the video file" << endl;
         return -1;
    }

    //cap.set(CV_CAP_PROP_POS_MSEC, 300);
    //start the video at 300ms

    double fps = cap.get(CV_CAP_PROP_FPS);
    //get the frames per seconds of the video

     cout << "Frame per seconds : " << fps << endl;

    namedWindow("MyVideo",CV_WINDOW_AUTOSIZE);
    //create a window called "MyVideo"

    while(1)
    {
        Mat frame;

        bool bSuccess = cap.read(frame);
        // read a new frame from video

         if (!bSuccess) //if not success, break loop
        {
                        cout << "Cannot read the frame from video file" << endl;
                       break;
        }

        imshow("MyVideo", frame); //show the frame in "MyVideo" window

        if(waitKey(30) == 27)
        //wait for 'esc' key press for 30 ms. If 'esc' key is pressed, break loop
       {
                cout << "esc key is pressed by user" << endl;
                break;
            }
        }

        return 0;

}
////////////////////////////////
```

3.1 Relevant functions

`VideoCapture::VideoCapture(const string& filename)` This is one of few constructors available in VideoCapture class. This constructor open the video file and initializes the VideoCapture object for reading the video stream from the specified file. The destructor of this class will deallocated any associated memory with this object. Therefore you don't need to deallocate memory explicitly in your program.

`bool VideoCapture::IsOpened()` If the previous call to VideoCapture constructor is successful, this method will return true. Otherwise it will return false. It is essential to check that whether the VideoCapture initialize successfully. If it is unsuccessful, program should be exited. Otherwise when you try to read a frame from the VideoCapture object, your program will crash.

`bool VideoCapture::set(int propId, double value)` You can change some properties of VideoCapture object. If it is successful, this method will return true. Otherwise it will return false. You should try to change some properties of the video stream in your code. In my code, I have shown you how to change the position of the video, by changing the `CV_CAP_PROP_POS_MSEC` property.

Parameters are :

- `int propID` - This argument specify the property you are going to change. There are many options for this argument. Some of them are listed here.
- `CV_CAP_PROP_POS_MSEC` - current position of the video in milliseconds
- `CV_CAP_PROP_POS_FRAMES` - current position of the video in frames
- `CV_CAP_PROP_FRAME_WIDTH` - width of the frame of the video stream
- `CV_CAP_PROP_FRAME_HEIGHT` - height of the frame of the video stream
- `CV_CAP_PROP_FPS` - frame rate (frames per second)
- `CV_CAP_PROP_FOURCC` - four character code of codec

`double value` - This is the new value you are going to assign to the property, specified by the propID

`doubleVideoCapture::get(int propId)`. This function returns the value of the property which is specified by propId. You should try to obtain some properties of the video stream in your code. In my code, I have shown you how to obtain the frame rate (frames per second) of the video, by using the `CV_CAP_PROP_FPS` argument.

Parameters are :

- `int propID` - This argument specify the property you are going to obtain. There are many options for this argument. Some of them are listed here.
- `CV_CAP_PROP_POS_MSEC` - current position of the video in milliseconds
- `CV_CAP_PROP_POS_FRAMES` - current position of the video in frames
- `CV_CAP_PROP_FRAME_WIDTH` - width of the frame of the video stream
- `CV_CAP_PROP_FRAME_HEIGHT` - height of the frame of the video stream

- `CV_CAP_PROP_FPS` - frame rate (frames per second)
- `CV_CAP_PROP_FOURCC` - four character code of codec

`bool VideoCapture::read(Mat& image);` : The function grabs the next frame from the video, decodes it and stores it in the 'image' variable. Inside this function, VideoCapture : :grap() and VideoCapture : :retrieve() will be called. If you want, you can use these two functions instead of VideoCapture : :read() function. If the operation successful, it will return true. Otherwise it will return false.

`waitKey(30)` : The function waits for 30 milliseconds. If a key was pressed before the specified time, it returns the ASCII value of the pressed key. If that value is 27 (ASCII value of `'esc'` key is 27), the program will execute inside the if block. If no key is pressed during that 30ms, the function returns -1 program will continue the while loop.

`VideoCapture::~VideoCapture()` Destructor of VideoCapture object will destroy any associated memory of that particular object. This destructor will be called implicitly on exit of the main method of the above program.

3.2 Summary

At first, this program captures a video from a file. Then the program enters into a infinite loop. In that loop, it grabs frames from the captured video sequentially, decodes it, shows it in a window and waits for 30 milliseconds. If the video file has no more frames or if the user presses the `'esc'` key, the program will break the infinite loop.

Note : Using `waitKey(int)` function is very important because `imshow(string&, MAT)` function need time to paint the image in the window and `waitKey(int)` will give that necessary time.

3.3 Questions

1. test the above program by reading an video sequence.
2. modify this program to display your PC WebCam. The main difference of the following program from the above program is the argument to the VideoCapture constructor. Here you just need to give the index of your camera to the constructor of the VideoCapture object instead of the filename in the above program :

   ```
   VideoCapture cap(0); // open the video camera no. 0
   ```

4 Conclusion

This tutorial was a short introduction to both capture and display images and videos from a file or a webcam. Image and video processing is now the next step.