Author NTN (/profile/4973) ○ 2020-07-28

19658 11

Xử Lý Ảnh Với OpenCV Trong C++ Cho Người Mới Bắt Đầu

OpenCV (Open Source Computer Vision) là một thư viện mã nguồn mở về thị giác máy với hơn 500 hàm và hơn 2500 các thuật toán đã được tối ưu về xử lý ảnh, và các vấn đề liên quan tới thị giác máy. Bài viết này sẽ giúp các bạn hiểu openCV là gì và các thao tác xử lý ảnh cơ bản với nó

OpenCV là gì?

OpenCV được thiết kế một cách tối ưu, sử dụng tối đa sức mạnh của các dòng chip đa lõi... để thực hiện các phép tính toán trong thời gian thực, nghĩa là tốc độ đáp ứng của nó có thể đủ nhanh cho các ứng dụng thông thường. OpenCV là thư viện được thiết kế để chạy trên nhiều nền tảng khác nhau (cross-patform), nghĩa là nó có thể chạy trên hệ điều hành Window, Linux, Mac, iOS ... Việc sử dụng thư viện OpenCV tuân theo các quy định về sử dụng phần mềm mã nguồn mở BSD do đó bạn có thể sử dụng thư viện này một cách miễn phí cho cả mục đích phi thương mại lẫn thương mại.

Dự án về OpenCV được khởi động từ những năm 1999, đến trong một hội nghị của IEEE về các vấn đề trong thị giác máy và nhận dạng, tuy nhiên bản OpenCV 1.0 mãi tới tận năm 2006 mới chính thức được công bố và năm 200

Hi! How can we help you?

CODELEARN (/) Login

1.1 (pre-release) mới được ra đời. Tháng 10 năm 2009, bản OpenCV thế hệ thứ hai ra đời (thường gọi là phiên bản 2.x), phiên bản này có giao diện của C++ (khác với phiên bản trước có giao diện của C) và có khá nhiều điểm khác biệt so với phiện bản thứ nhất.

Thư viện OpenCV ban đầu được sự hỗ trợ từ Intel, sau đó được hỗ trợ bởi Willow Garage, một phòng thí nghiệm chuyên nghiên cứu về công nghệ robot. Cho đến nay, OpenCV vẫn là thư viện mở, được phát triển bởi nguồn quỹ không lợi nhuận (none -profit foundation) và được sự hưởng ứng rất lớn của cộng đồng.

Cài đặt OpenCV với C++

- Tải và cài đặt CMake tại địa chỉ: https://cmake.org/download/ (https://cmake.org/download/)
- Tải và cài đặt MinGW tại địa chỉ: https://sourceforge.net/projects/mingw-w64/ (https://sourceforge.net/projects/mingw-w64/)

Build thư viện OpenCV từ Source Code bằng CMake

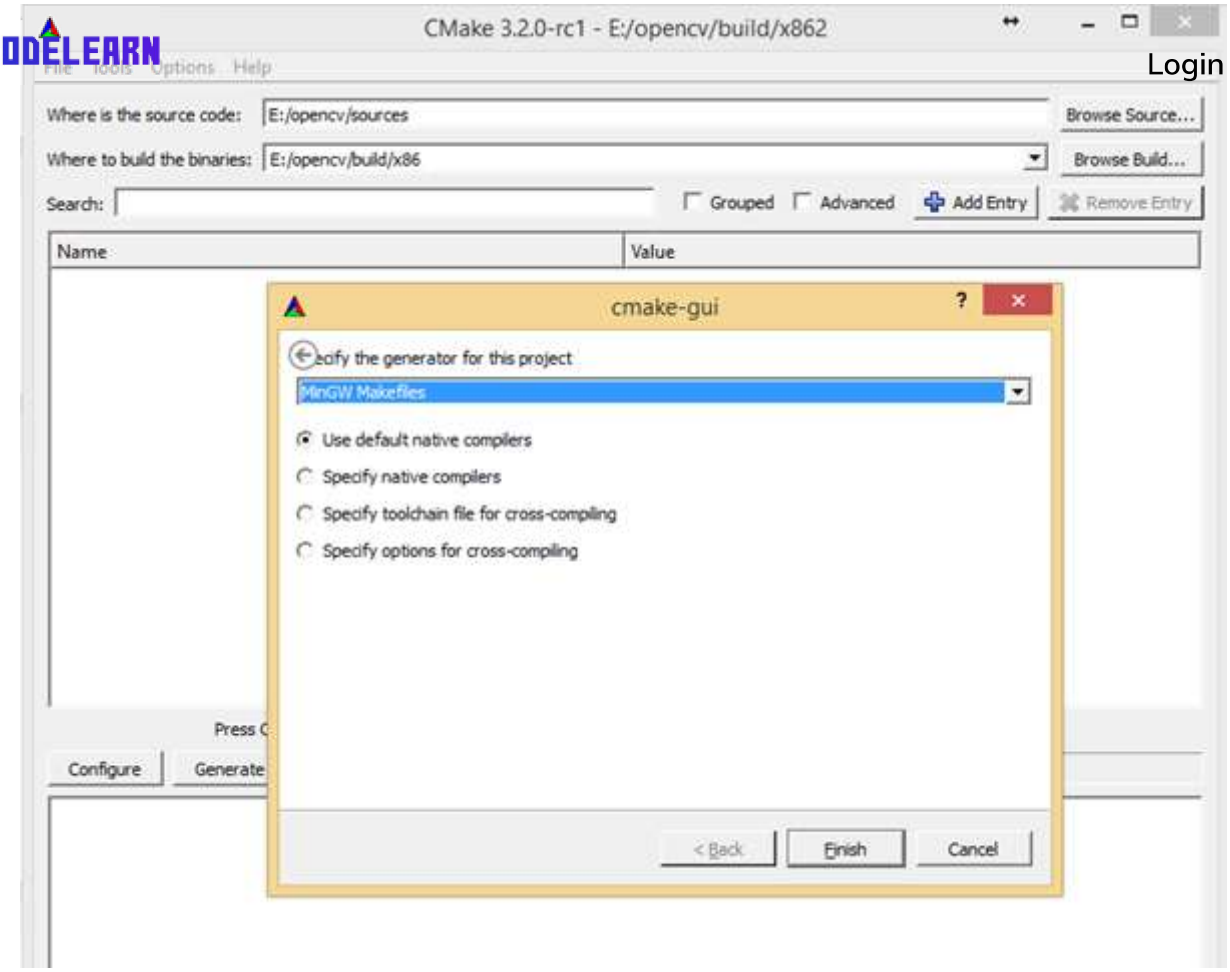
Tại trường "Where is the code", chọn địa chỉ source code trong thư mục OpenCV vừa cài đặt là **E:/opencv/sources**, và trường "Where to build the binaries" tại một thư mục sẽ sử dụng để build. Ở đây mình chọn là **E:/opencv/build/x86**. Sau khi chọn xong ấn vào nút **Configure**.

Các bạn chọn genertor bằng **MinGW Makefiles**

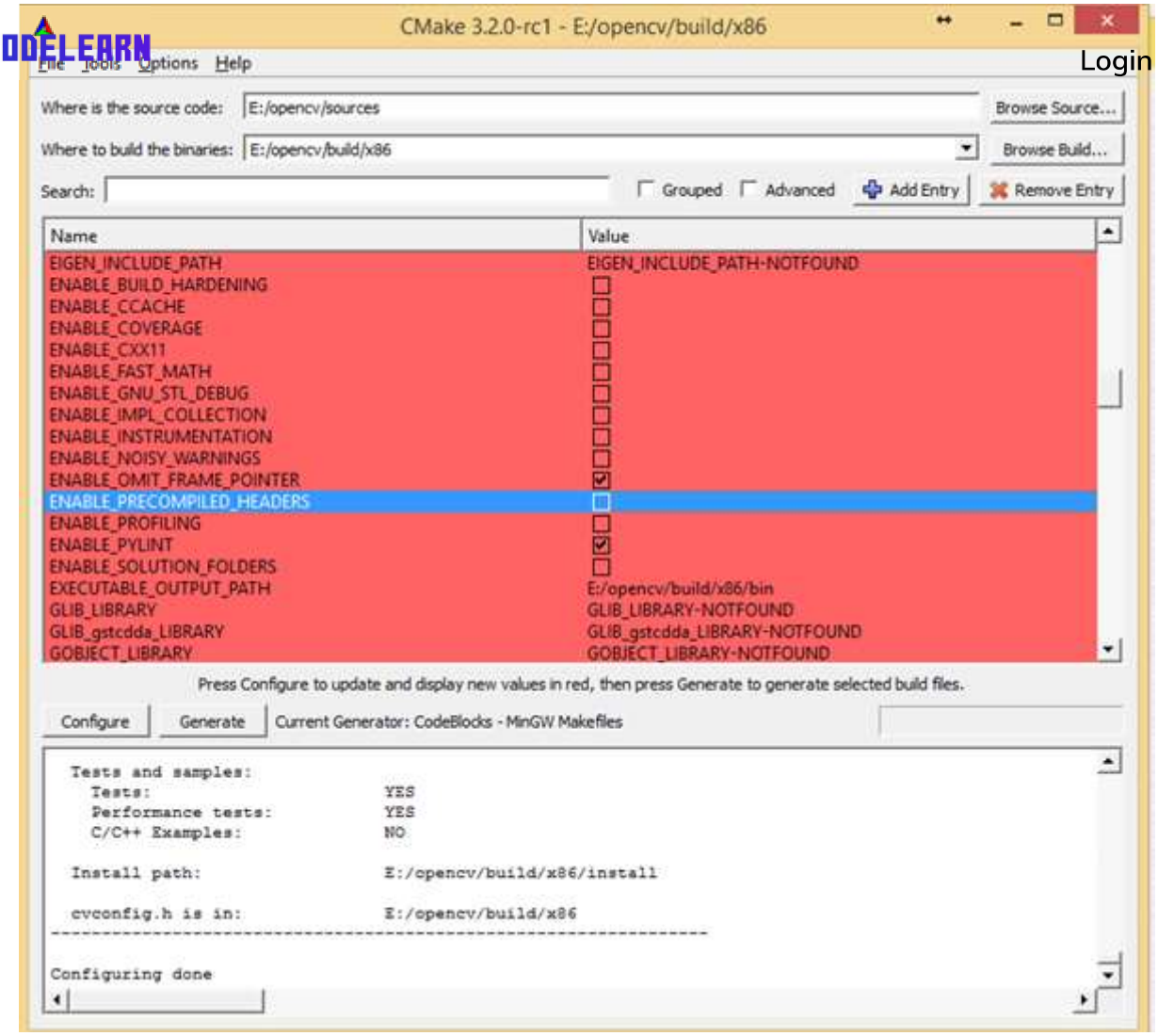
Chú ý: Khi hiển thị config lên, các bạn nhớ bỏ chọn dòng **ENABLE_PRECOMPILED_HEADERS**

Ấn nút **Generate**

Hi! How can we help you?



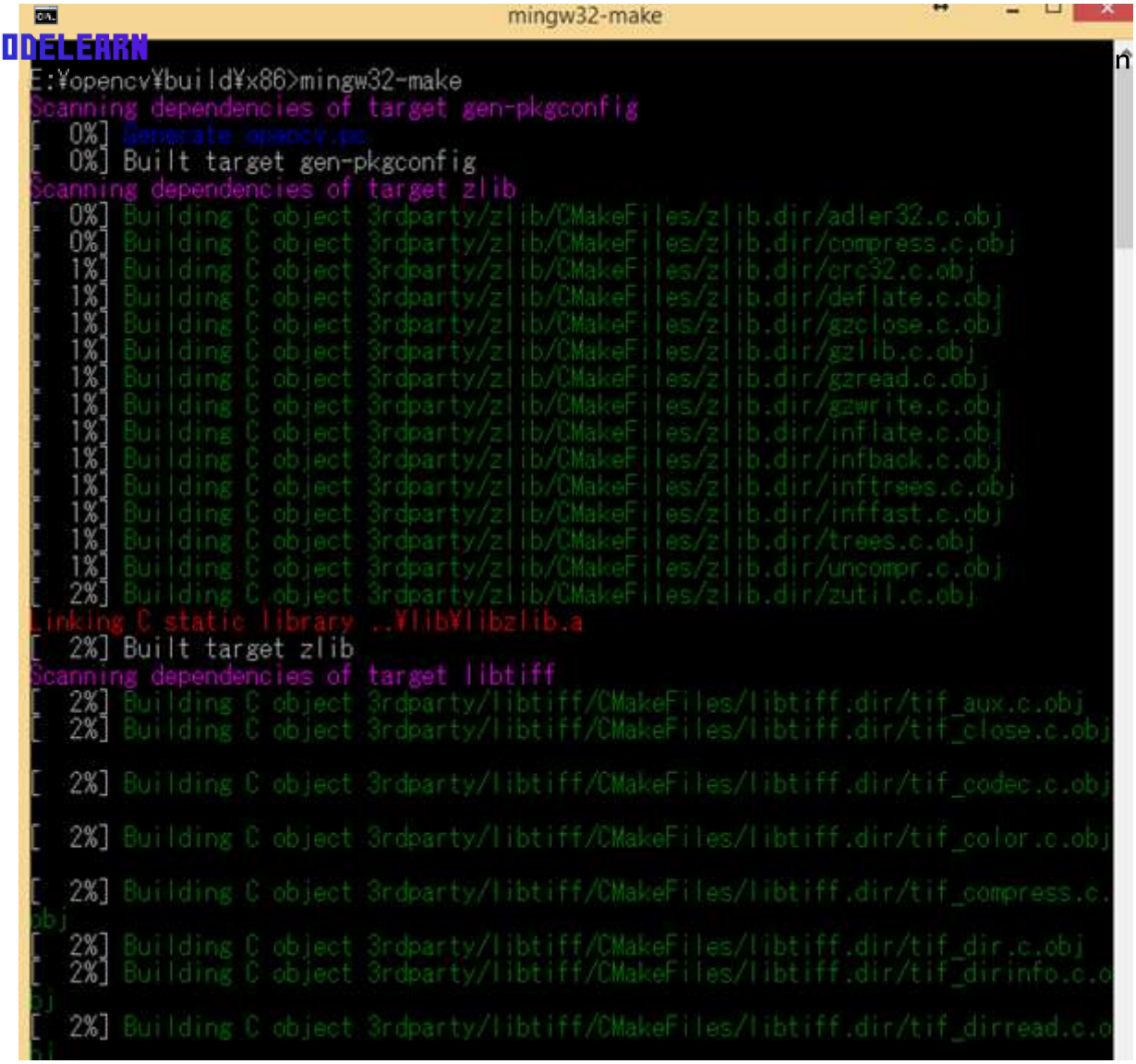
Hi! How can we help you?



Sau khi CMake tạo xong, các bạn hãy chạy lệnh mingw32-make từ thư viện MinGW vừa cài. Nếu các bạn muốn chạy nhiều core (tăng tốc độ thực hiện), có thể thực hiện lệnh **mingw32-make-j4** (-j4 ở đây mang ý nghĩa builf trên 4 core CPU)

Hi! How can we help you?

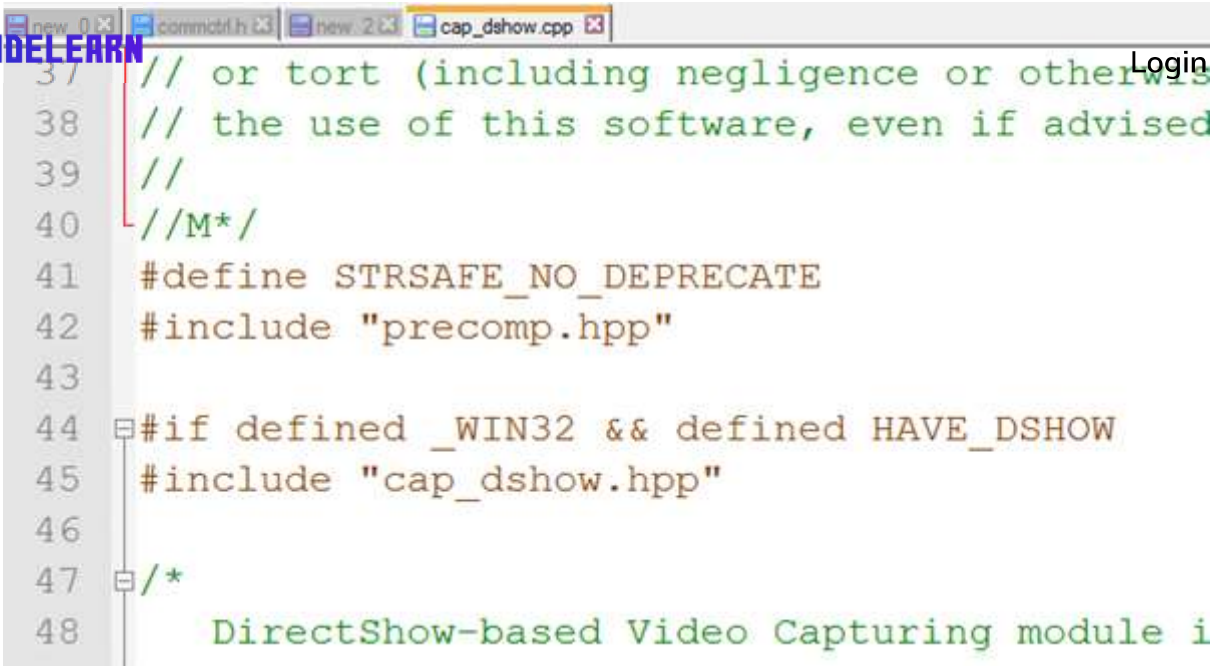
(/)



Chú ý: Khi gặp phải lỗi khi build module **videoio**, hãy mở đến file cap_dshow.cpp và thêm dòng code sau trên đầu file

```
#define STRSAFE_NO_DEPRECATE
```

Hi! How can we help you?



Các phép xử lý ảnh đơn giản

- Để xử lý một ảnh, có rất nhiều vấn đề mà chúng ta quan tâm đến. Tuy nhiên để có thể thao tác xử lý ảnh một cách chuyên nghiệp với OpenCV, bạn cần phải nắm rõ các cách xử lý cơ bản với OpenCV trong C++.
- Trong bài viết lần này, mình sẽ hướng dẫn các bạn một số phép xử lý ảnh đơn giản với OpenCV trong C++ như: điều chỉnh độ sáng, độ tương phản, phóng to, thu nhỏ, xoay ảnh,...

1. Cách load ảnh và hiển thị một ảnh với OpenCV trong C++

Chương trình minh họa:

Hi! How can we help you?

CODELEARN
(/)
Login

```
#include "stdfx.h"
#include <iostream>
#include <opencv2\highgui\highgui.hpp>
#include <opencv2\core\core.hpp>

using namespace std;
using namespace cv;

int main(){
    cout << "chuong trinh dau tien" << endl;
    Mat img = imread("vietnam.jpg", CV_LOAD_IMAGE_COLOR);
    namedWindow("Viet Nam", CV_WINDOW_AUTOSIZE);
    imshow("Viet Nam", img);
    waitKey(0);
    return 0;
}
```

Trong OpenCV với giao diện C++, tất cả các kiểu dữ liệu ảnh, ma trận đều được lưu dưới dạng cv::Mat. Hàm **imread** sẽ đọc ảnh đầu vào và lưu vào biến *img*. Nguyễn mẫu của hàm này như sau: **cv::Mat imread(const std::string &filename, int flags)** trong đó, filename là đường dẫn tới file ảnh, nếu file ảnh không nằm trong thư mục làm việc hiện hành thì ta phải chỉ ra đường dẫn tương đối có dạng như *D:\Anh\abc.jpg*. *Flags* là tham số loại ảnh mà ta muốn load vào, cụ thể nếu muốn load ảnh màu thì ta để *CV_LOAD_IMAGE_COLOR*, nếu là ảnh xám thì ta để *CV_LOAD_IMAGE_GRAYSCALE*....

Để hiển thị ảnh lên màn hình ta phải tạo ra một cửa sổ, hàm **namedWindow(const std::string &winname, int flags)** sẽ tạo ra cửa sổ với tiêu đề cửa sổ là một chuỗi string *winname*. Tham số *flags* sẽ chỉ ra kiểu cửa sổ muốn tạo: nếu tham số *CV_WINDOW_AUTOSIZE* được sử dụng thì kích cỡ cửa sổ tạo ra sẽ được hiển thị một cách tự động tùy thuộc vào kích thước của ảnh, nếu là tham số *CV_WINDOW_AUTOSIZE_FULLSCREEN* kích thước cửa sổ sẽ khít với màn hình máy tính...

Hàm **imshow(const std::string winname, cv::InputArray Mat)** sẽ hiển thị ảnh ra cửa sổ đã được tạo trước đó.

Hàm **waitKey(int delay)** sẽ đợi cho đến khi có một phím [illegible] thời gian delay. Ta dùng hàm này mục đích là để dừng m[illegible] thời gian bằng tham số delay (tính theo đơn vị ms). Nếu muốn dừng lại màn hình mã[illegible] đặt tham số delay bằng 0.

Hi! How can we help you?

2. Điều chỉnh độ sáng và độ tương phản trong ảnh

Một điểm ảnh được lưu trữ trên máy tính là một ma trận các điểm ảnh (hay pixel). Trong OpenCV nó được biểu diễn dưới dạng *cv::Mat*. Ta xét một kiểu ảnh thông thường nhất, đó là ảnh RGB. Với ảnh này, mỗi pixel ảnh quan sát được là sự kết hợp của các thành phần màu R (Red), G (Green), B (Blue). Sự kết hợp này theo tỉ lệ R, G, B khác nhau sẽ tạo ra vô số các màu sắc khác nhau. Giả sử ảnh được mã hóa bằng 8 bit với từng kênh màu, khi đó mỗi giá trị của R, G, B sẽ nằm trong khoảng [0, 255]. Như vậy, ta có thể biểu diễn tới 255*255*255 ~ 1,6 triệu màu từ ba màu cơ bản trên. Ta có thể xem cách biểu diễn ảnh trong OpenCV ở định dạng *cv::Mat* qua hình ảnh sau:

	Cột 0			Cột 1			Cột m	
Hàng 0	0, 0	0, 0	0, 0	0, 1	0, 1	0, 1	0, m	0, m	0, m
Hàng 1	1, 0	1, 0	1, 0	1, 1	1, 1	1, 1	1, m	1, m	1, m
Hàng 2	2, 0	2, 0	2, 0	2, 1	2, 1	2, 1	2, m	2, m	2, m
Hàng n	n, 0	n, 0	n, 0	N, 1	n, 1	n, 1	n, m	n, m	n, m

Như vậy, mỗi ảnh sẽ có n hàng và m cột, m được gọi là chiều dài của ảnh, n được gọi là chiều cao của ảnh. Mỗi pixel ở vị trí (i, j) trong ảnh sẽ tương ứng với 3 kênh màu kết hợp trong nó. Để truy xuất tới từng pixel ảnh với những kênh màu riêng ta sẽ sử dụng mẫu sau:

img.at<cv::Vec3b>(i, j)[k]

Trong đó, i, j là pixel ở hàng thứ i và cột thứ j, img là ảnh mà ta cần truy xuất tới các pixel của nó. *Cv::Vec3b* là kiểu vector *uchar* 3 thành phần, dùng để biểu thị 3 kênh màu tương ứng. k là kênh màu thứ k, k= 0,1,2,... tương ứng với kênh màu B, G, R. Chú ý là trong OpenCV, hệ màu RGB được biểu diễn theo thứ tự chữ cái là BGR.

Sau đây ta sẽ áp dụng kiến thức trên để làm tăng, giảm độ sáng và tương phản của một ảnh màu, việc làm này cũng hoàn toàn tương tự đối với ảnh xám, chỉ khác biệt là ảnh ta dùng một kênh duy nhất để biểu diễn ảnh xám.

Giả sử f là một hàm biểu diễn cho một ảnh nào đó, f(x,y) là giá trị của pixel trong ảnh vị trí (x,y). Đặt $g(x,y) = \alpha f(x,y) + \beta$. Khi đó, nếu , thì ta nói ảnh g(x,y) có độ tương phản gấp lần so với ảnh f(x,y. Nếu ta nói độ sáng của ảnh g(x,y) đã thay đổi một lượng là . Dựa vào công thức trên ta có chương trình thay đổi độ sáng và tương phản của ảnh như sau:

Hi! How can we help you?

CODELEARN
(/)
Login

```
#include "stdfx.h"
#include <iostream>
#include <opencv2\highgui\highgui.hpp>
#include <opencv2\core\core.hpp>

using namespace std;
using namespace cv;

int main(){
    cout << "chuong trinh dieu chinh do sang va tuong phan" << endl;
    Mat src = imread("hoa_huong_duong.jpg", 1);
    Mat dst = src.clone();
    double alpha = 2.0;
    int beta = 30;
    for(int i = 0; i < src.rows; i++)
        for(int j = 0; j < src.cols; j++)
            for(int k = 0; k < 3; k++)
                dst.at<Vec3b>(i,j)[k]  = saturate_cast<uchar>(alpha*(src
    imshow("anh goc", src);
    imshow("anh co sau khi chinh do tuong phan va do sang", dst);
    waitKey(0);
    return 0;
}
```

Trong chương trình trên, hàm **clone()** sẽ sao chép một ảnh giống hệt như ảnh gốc cho vào ảnh đích (**drt = src.clone()**). Giá trị của các pixel ảnh *f(x,y)* và *g(x,y)* ở đây phải nằm trong khoảng [0,255], trong khi phép biến đổi $g(x,y) = \alpha f(x,y) + \beta$ có thể khiến cho giá trị *g(x,y)* vượt qua ngưỡng đó. Để tránh tình trạng tràn số hoặc kiểu dữ liệu không tương thích, ta dùng thêm hàm **saturate_cast<uchar>(type)**. Hàm này sẽ biến kiểu dữ liệu *type* nếu không phải là *uchar* thành kiểu dữ liệu *uchar*.

Sau đây là kết quả với α = 2.0 và β = 30.

Hi! How can we help you?

3. Phóng to, thu nhỏ và xoay ảnh

Ảnh số thực chất là một ma trận các điểm ảnh, do đó để có thể phóng to, thu nhỏ hay xoay một tấm ảnh ta có thể sử dụng các thuật toán tương ứng trên ma trận

Ta sẽ sử dụng biến đổi *affine* để quay và thay đổi tỉ lệ to, nhỏ của một ma trận.

Biến đổi affine

Giả sử ta có vector

$$p = [x, y]^T$$

và ma trận M có kích thước 2x2. Phép biến đổi *affine* trong không gian hai chiều được định nghĩa p' = Mp, trong đó:

$$p' = [x', y']^T$$

Viết một cách tường minh ta có:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \alpha & \delta \\ \gamma & \beta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

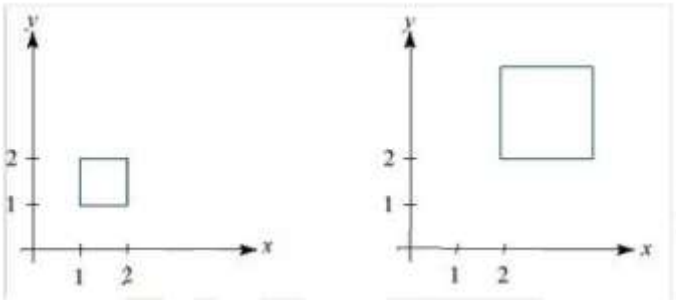
Hay x' = αx + δy, y' = γx + βy.

Xét ma trận

$$\begin{bmatrix} \alpha & \delta \\ \gamma & \beta \end{bmatrix}$$

Nếu δ = γ, khi đó x' = αx và y' = βy, phép biến đổi này làm thay đổi tỉ lệ của ma trận. Nếu là trong ảnh nó sẽ phóng to hoặc thu nhỏ ảnh. Hình sau mô tả phép biến đổi với tỉ lệ α = β = 2.

Hi! How can we help you?



Nếu ta định nghĩa ma trận

$$M = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

thì phép biến đổi sẽ vừa là phép biến đổi theo tỉ lệ và quay.

Bây giờ ta sẽ xét chương trình phóng to, thu nhỏ và quay ảnh.

```
#include "stdfx.h"
#include <iostream>
#include <opencv2\highgui\highgui.hpp>
#include <opencv2\core\core.hpp>
#include <opencv2\imgproc\imgproc.hpp>

using namespace std;
using namespace cv;

int main(){
    Mat src = imread("HoaSen.jpg");
    Mat dst = src.clone();

    double angle = 45.0;
    double scale = 1.5;
    Point2f center(src.cols / 2, src.rows / 2);
    Mat mat_rot = getRotationMatrix2(center, angle, scale);
    warpAffine(src, dst, mat_rot, src.size());
    imshow("Anh goc", src);
    imshow("Anh sau phep bien doi", dst);
    waitKey(0);
    return 0;
}
```

Hi! How can we help you?

Trong chương trình trên, hàm *cv::getRotationMatrix2D(cv::Point center, double ar double scale)* sẽ tạo ra ma trận với tâm quay *center*, góc quay *angle* và tỉ lệ *scale* trận này được tính toán trong OpenCV là ma trận như sau:

$$M = \begin{bmatrix} \alpha & \beta & (1-\alpha).center.x - \beta.center.y \\ -\beta & \alpha & \beta.center.x - (1-\alpha).center.y \end{bmatrix}$$

với α = scale.cos(angle) và β = scale.sin(angle).

Ta thấy rằng ma trận này là hoàn toàn tương đương với ma trận của phép biến đổi affine đã nói ở trên, ngoại trừ thành phần thứ 3 là thành phần giúp dịch chuyển tâm quay vào chính giữa của bức ảnh. Chú ý là có sự khác biệt một chút về chiều của hệ tọa độ trong ảnh, hệ tọa độ trong ảnh lấy góc trên bên trái làm gốc tọa độ (0,0) còn hệ tọa độ thông thường ta hay lấy điểm dưới bên trái làm gốc, do đó có sự ngược chiều.

Kết quả của chương trình trên với tỉ lệ scale = 1.5 và góc quay = 45 độ:

Ngoài hai phép biến đổi là tỉ lệ và quay như trên, ta có thể thực hiện các biến đổi khác của phép biến đổi *affine* như phép trượt (shearing), hoặc phép phản chiếu (reflection) bằng việc định nghĩa lại ma trận M. Ta thử định nghĩa lại ma trận M để được một ảnh trượt của ảnh gốc. Quay lại ma trận M như trên, nếu ta định nghĩa α = β = 1 còn δ nhận một giá trị bất kì, khi đó ta sẽ có:

$$\begin{cases} x' = x + \delta y \\ y' = y \end{cases}$$

ta sẽ định nghĩa ma trận

$$M = \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

để trượt ảnh ban đầu thành ảnh mới với hệ số mượt δ = 0.5, chú ý là thành phần thứ ba

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

định nghĩa ma trận trong OpenCV sẽ thể hiện độ dịch chuyển, giống như trong ví dụ trên ta chuyển tâm quay về tâm của bức ảnh chẳng hạn. Ta sẽ làm giống hệt ví dụ trên, chỉ thay ma trận M là ma trận ta tự định nghĩa

```
double I[2][3] = {1, 0.5, 0, 0, 1, 0};
Mat mat_rot(2, 3, CV_64F, I);
warpAffine(src, dst, mat_rot, src.size());
```

Hi! How can we help you?

Và ta có kết quả:

Tạm kết

Trên đây mình đã giới thiệu đến các bạn một vài kỹ thuật xử lý ảnh cơ bản sử dụng OpenCV trong C++. Trong các bài viết tiếp theo, mình sẽ giới thiệu đến các bạn các kỹ thuật xử lý ảnh ở level cao hơn. Qua đó các bạn có thể dùng OpenCV để nhận diện khuôn mặt hay nhận diện chữ viết.
Cảm ơn các bạn đã đọc bài viết của mình. Các bạn cùng đón chờ bài viết tiếp theo trong series **xử lý ảnh với OpenCV trong C++** nhé!!!

<Tham khảo Internet>

opencv (/sharing/tags/opencv) c++ (/sharing/tags/c%2b%2b)

hoc-lap-trinh (/sharing/tags/hoc-lap-trinh)

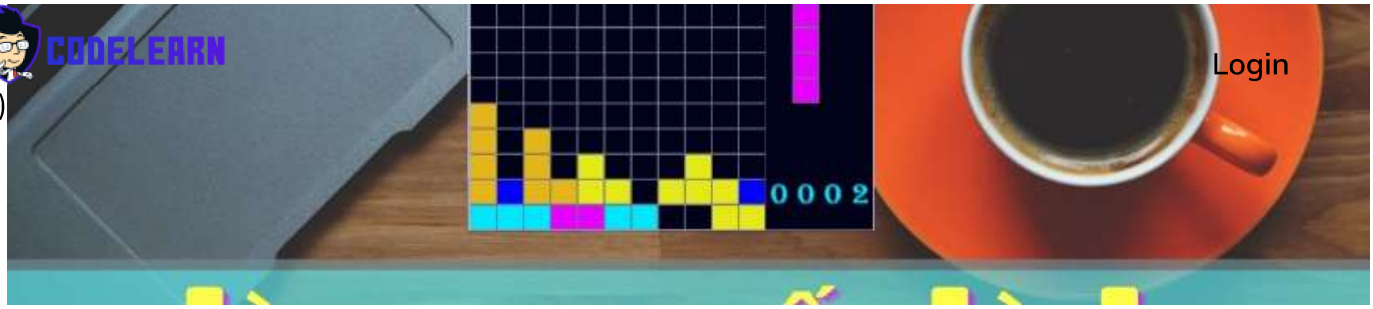
Share

★★★★★ 5.0 (3 votes)

Please login to comment

Related posts

Hi! How can we help you?



(/sharing/lam-game-xep-hinh-bang-java)

Làm Game Xếp Hình Bằng Java (/sharing/lam-game-xep-hinh-bang-java)

★★★★★

Xin chào các bạn chắc hẳn các bạn đã quen thuộc với trò chơi xếp hình cổ điển, trong bài viết này mình sẽ hướng dẫn các bạn làm trò chơi này bằng ngôn ngữ Java nhé.

game (/sharing/tags/game) java (/sharing/tags/java) hoc-lap-trinh (/sharing/tags/hoc-lap-trinh) tutorial (/sharing/tags/tutorial)

Author: HaiZuka

○ 2021-06-14 ○ 12501 0

(/sharing/lap-trinh-game-sudoku-bang-java)

Lập Trình Game Sudoku Bằng Java (/sharing/lap-trinh-game-sudoku-bang-java)

★★★★★

Trò chơi sudoku đòi hỏi ta sự nhanh nhạy nhưng vẫn phải đảm bảo tính chính xác và cẩn thận. Hôm nay minh sẽ hướng dẫn các bạn lập trình game sudoku nhé.

hoc-lap-trinh (/sharing/tags/hoc-lap-trinh) sudoku (/sharing/tags/sudoku) java (/sharing/tags/java)

Author: HaiZuka

○ 2021-04-29 ○ 9521 1

Hi! How can we help you?

(/)

(/sharing/fron-end-hay-back-end-chon-cai-nao)

Front-end Hay Back-end, Lựa Chọn Nào Ổn Hơn? (/sharing/fron-end-hay-back-end-chon-cai-nao)

★★★★★

Có rất nhiều vai trò khác tham gia vào phát triển web. Nhưng hầu hết, thường sẽ có hai loại lập trình viên: một cho front end và một cho backend.

back-end (/sharing/tags/back-end) front-end (/sharing/tags/front-end)
programming (/sharing/tags/programming) hoc-lap-trinh (/sharing/tags/hoc-lap-trinh)

Author: Phoebe

○ 2021-03-22 ○ 11779 108

(/sharing/can-bang-phuong-trinh-hoa-hoc-bang-cpp)

Cân Bằng Phương Trình Hóa Học Bằng C++ (/sharing/can-bang-phuong-trinh-hoa-hoc-bang-cpp)

★★★★★

Hóa học là một nỗi sợ và có thể nói là nỗi ám ảnh của đa số học sinh, vậy nếu ta tạo ra một chương trình cho phép cân bằng PTHH một cách chớp nhoáng thì sao nhỉ?

c++ (/sharing/tags/c%2b%2b) pthh (/sharing/tags/pthh) tutorial (/sharing/tags/tutorial)
hoc-lap-trinh (/sharing/tags/hoc-lap-trinh)

Hi! How can we help you?

Author: HaiZuka

○ 2021-03-28 ○ 14886 441

(/sharing/nguon-hoc-cau-truc-du-lieu-va-giai-thuat)

1001 Nguồn Học Cấu Trúc Dữ Liệu Và Giải Thuật Cực Hiệu Quả (/sharing/nguon-hoc-cau-truc-du-lieu-va-giai-thuat)

★★★★★

Mình bắt đầu gia nhập Amazon dưới vai trò Thực tập sinh Kỹ thuật Phát triển Phần mềm trong 6 tháng kể từ tháng 2 năm 2021. Hôm nay, mình sẽ chia sẻ tất cả các tài nguyên quan trọng mà mình đã theo h...

algorithm (/sharing/tags/algorithm) data-structure (/sharing/tags/data-structure)
hoc-lap-trinh (/sharing/tags/hoc-lap-trinh)

Author: Phanhtrinh

○ 2021-03-15 ○ 26940 422

AUTHOR

NTN (/profile/4973)

-

3 posts (/sharing/post/NTN) | 4 followers

Follow

(https://www.facebook.com/hiep.phamminhhiep.9/) (javaScript

Hi! How can we help you?

WRITTEN BY SAME AUTHOR

(/)

(/sharing/tim-bien-anh-de-dang-voi-bo-loc-canny) Tìm Biên Ảnh Dễ Dàng Với Bộ Lọc Canny (/sharing/tim-bien-...

(/sharing/ky-thuat-loc-anh-voi-opencv-trong-cpp) Kỹ Thuật Lọc Ảnh Với OpenCV Trong C++ Cho Beginner...

TOP READS

Kiến Thức Cơ Bản Cần Nắm Khi Bắt Đầu Học C++ (/sharing/kien-thuc-co-ban-khi-bat-dau-hoc-cpp)

Hướng Dẫn Cài Đặt Visual Studio Code Lập Trình C++ (/sharing/huong-dan-cai-dat-visual-studio-code-lap-trinh-cpp)

Lộ Trình Học Cấu Trúc Dữ Liệu Và Giải Thuật (Phần 1) (/sharing/lo-trinh-hoc-cau-truc-du-lieu-va-giai-thuat-phan-1)

Độ Phức Tạp Của Thuật Toán Và Lựa Chọn Cách Giải Thuật (/sharing/do-phuc-tap-cua-thuat-toan-va-lua-chon-cach-giai-thuat)

Thuật toán là gì? Học thuật toán làm quái gì? (/sharing/thuat-toan-la-gi-hoc-thuat-toan-lam-quai-gi)

TOP AUTHORS

(/profile/415342)

T_Flower (/profile/415342)

(/profile/3305)

3ron (/profile/3305)

Hi! How can we help you?

HASHTAG

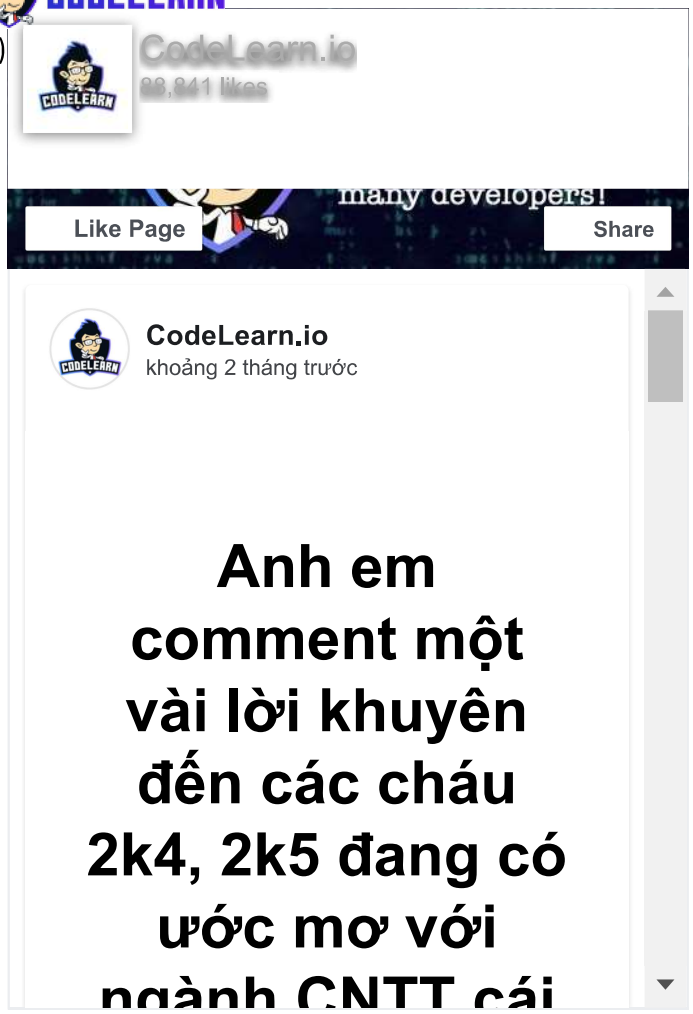
hoc-lap-trinh (/sharing/tags/hoc-lap-trinh) c++ (/sharing/tags/c%2b%2b) opencv (/sharing/tags/opencv)

Hi! How can we help you?

CODELEARN

(/)

Login



CODELEARN

CodeLearn is an online platform that helps users to learn, practice coding skills and join the online coding contests.

Hi! How can we help you?

LINKS

Learning (/learning)

Training (/training)

INFORMATION

About Us (/aboutus)

Terms of Use (/terms)

HELP

Help (/help)

Discussion (/discussion)

Contact us (mailto:support@codelearn.io)

Powered by CodeLearn (/) © 2022. All Rights Reserved. rev 7/23/2022 11:45:23 AM

Hi! How can we help you?