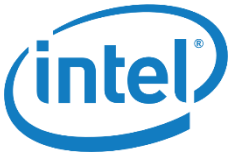
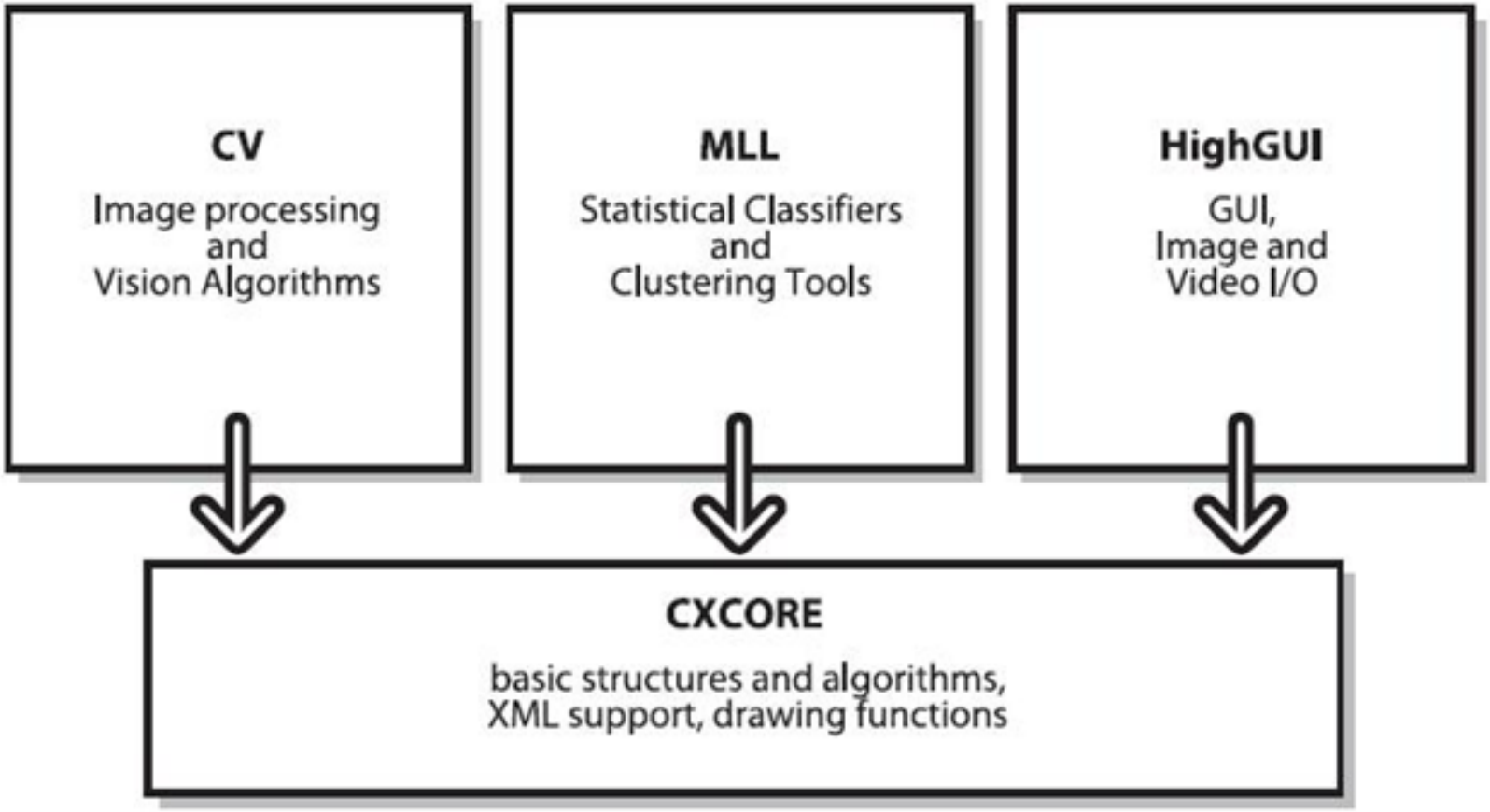
OPENCV

GIỚI THIỆU



- OpenCV (Open Source Computer Vision)
  - Giới thiệu bởi Intel
  - Thư viện mã nguồn mở
  - 500 hàm và 2500 thuật toán tối ưu về xử lý ảnh
  - chạy trên nhiều nền tảng khác nhau: Window, Linux, Mac OS, …
- Download miễn phí tại địa chỉ
  - http://opencv.org/
  - Tích hợp với các IDE phổ biến: Microsoft Visual Studio hoặc Eclipse CDT
    - Tích hợp với Microsoft Visual Studio: https://www.youtube.com/watch?v=v-VgWxkVp2w

CẤU TRÚC OPENCV



CÁC CẤU TRÚC CƠ BẢN

- **Point, Point2f** - 2D Point
- **Size** - 2D size structure
- **Rect** - 2D rectangle object
- **RotatedRect** - Rect object with angle
- **Mat** - image object

CẤU TRÚC POINT

- Một đối tượng mô tả một điểm
  - Có hai thành phần dữ liệu: *int x, y;*
  - Có thành phần hàm
    - Point.dot(<Point>) – tích hai điểm
    - Point.inside(<Rect>) – kiểm tra một điểm có nằm trong hình chữ nhật hay không
  - Có các toán tử:
    - +, -, *, +=, -=, *=, ==, !=

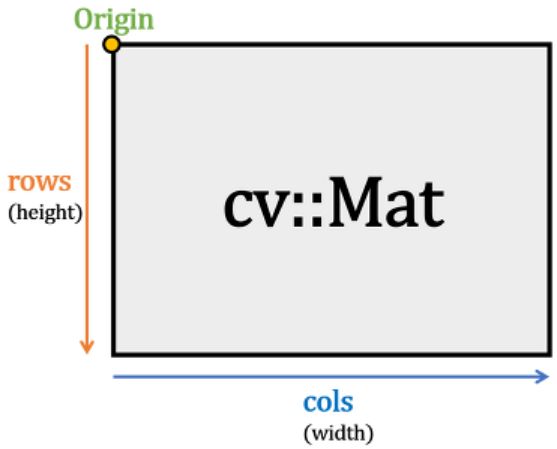
CẤU TRÚC SIZE, RECT

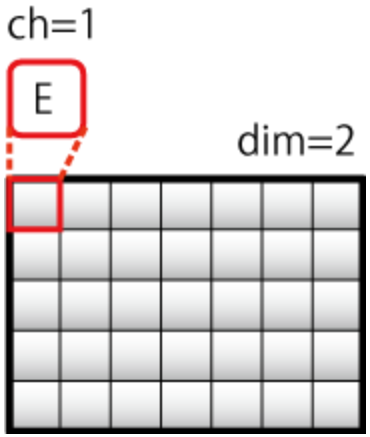
- SIZE: Một đối tượng mô tả kích thước
  - Có thành phần dữ liệu: *int width, height;*
  - Có thành phần hàm
    - Point.area() - returns (width * height)

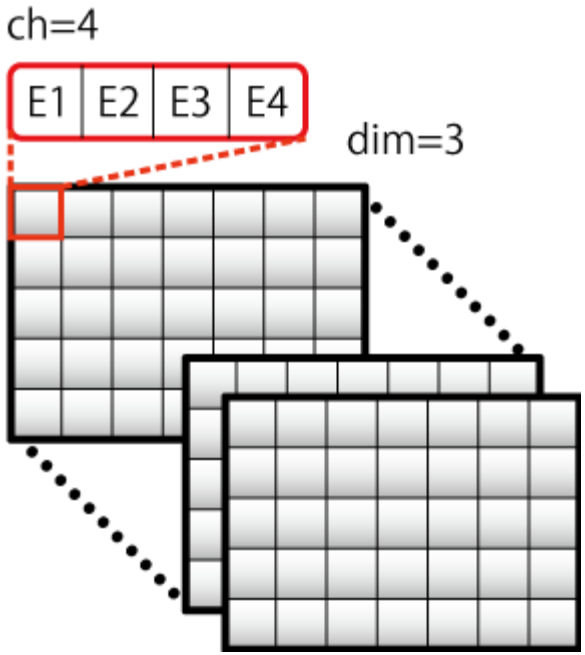
- RECT: Một đối tượng mô tả cấu trúc hình chữ nhật
  - Có thành phần dữ liệu: *int x, y, width, height;*
  - Có thành phần hàm
    - Point.tl() – trả về tọa độ điểm trên bên trái
    - Point.br() - trả về tọa độ điểm dưới bên phải

CẤU TRÚC cv::MAT

- Đây là cấu trúc cơ bản trong OpenCV thường dùng để lưu trữ ảnh
  - Có các thành phần dữ liệu
    - int rows, cols – chiều dài và chiều rộng ảnh
    - int channels – 1: ảnh mức xám, 3 ảnh màu BGR
    - int depth:CV_<depth>C<num chan>







CẤU TRÚC cv::MAT

- Các phương thức
  - Mat.**at**<datatype>(row, col)[channel] – con trỏ trỏ tới vị trí ảnh
  - Mat.**channels**() – số kênh trong ảnh
  - Mat.**clone**() – sao chép ảnh
  - Mat.**create**(rows, cols, TYPE) – cấp phát lại bộ nhớ cho mảng
  - Mat.**cross**(<Mat>) – tính tích chéo hai ma trận
  - Mat.**depth**() – kiểu dữ liệu của ma trận
  - Mat.**dot**(<Mat>) – tích chập hai ma trận
  - Mat(Range(xmin,xmax),Range(ymin,ymax)) – vùng ảnh con của ảnh hiện tại
  - Mat.**type**() – trả về kiểu của ma trận

CÁC KIỂU ẢNH

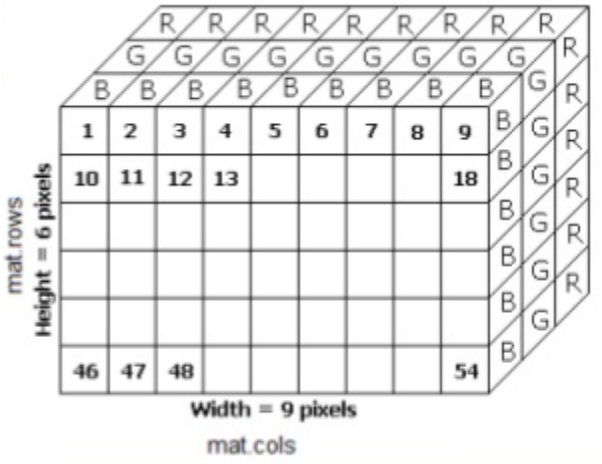
- Biểu diễn các kiểu dữ liệu chứa thông tin ảnh

OpenCV Tag	Representation	OpenCV Value
CV_8U	8 bit unsigned integer	0
CV_8S	8 bit signed integer	1
CV_16U	16 bit unsigned integer	2
CV_16S	16 bit signed integer	3
CV_32S	32 bit signed integer	4
CV_32F	32 bit floating point number	5
CV_64F	64 bit floating point number	6

BIỂU DIỄN ẢNH TRONG OPENCV

- Mỗi ảnh số tương đương với một ma trận các điểm ảnh *cv::Mat*
- Mỗi điểm ảnh bao gồm 3 thành phần RGB
  - Có 255*255*255 ~ 16 triệu màu

	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



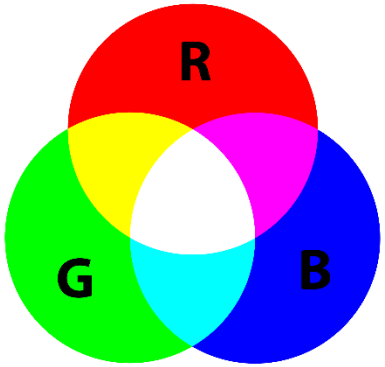
CHƯƠNG TRÌNH ĐẦU TIÊN

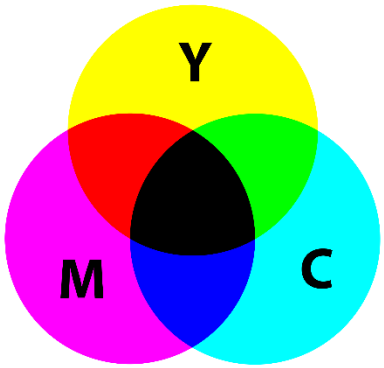
```
#include "stdafx.h"
#include <iostream>
#include <opencv2\highgui\highgui.hpp>
#include <opencv2\core\core.hpp>

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;
}
```

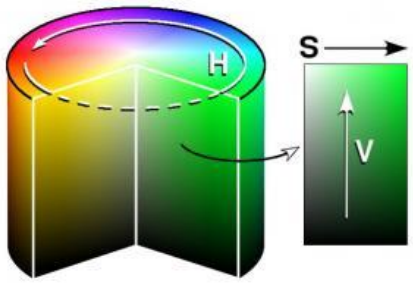
- Ví dụ đầu tiên: Đọc và hiển thị một ảnh
- Cú pháp C++
- Khai báo hai thư viện cơ bản
  - Highgui.hpp
  - Core.hpp
- Namesapce *cv*

CHUYỂN ĐỔI KHÔNG GIAN MÀU

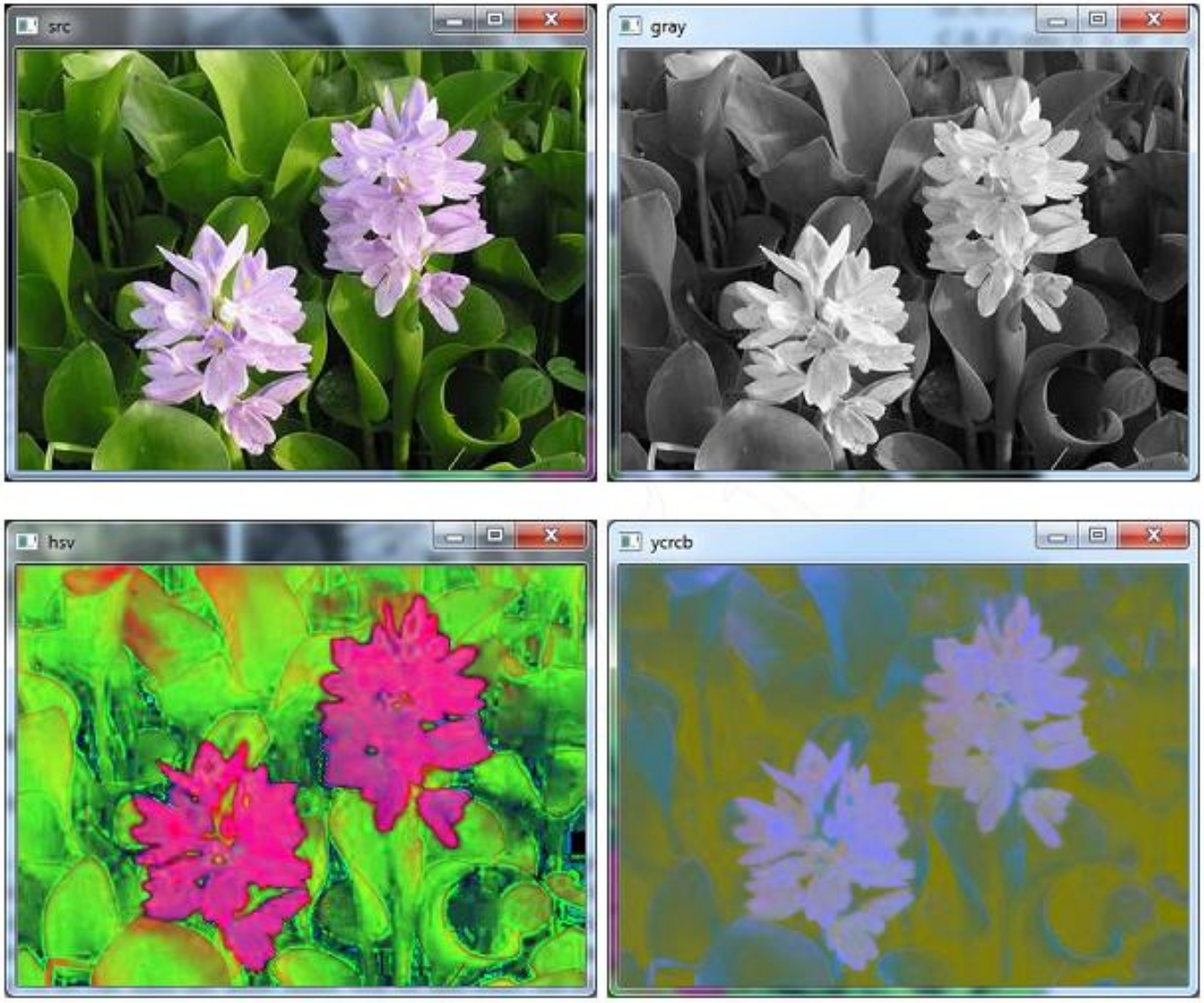
```
#include "stdafx.h"
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>

using namespace cv;

void main()
{
	Mat src = imread("LucBinh.jpg", CV_LOAD_IMAGE_COLOR);
	Mat gray, hsv, ycrcb;
	cvtColor(src, gray, CV_BGR2GRAY);
	cvtColor(src, hsv, CV_BGR2HSV);
	cvtColor(src, ycrcb, CV_BGR2YCrCb);
	imshow("src", src);
	imshow("gray", gray);
	imshow("hsv", hsv);
	imshow("ycrcb", ycrcb);
	waitKey(0);
 }
```







CHUYỂN ĐỔI KHÔNG GIAN MÀU



CHỈNH ĐỘ SÁNG VÀ TƯƠNG PHẢN

```
#include "stdafx.h"
#include <iostream>
#include <opencv2\core\core.hpp>
#include <opencv2\highgui\highgui.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.at<Vec3b>(i,j)[k] ) + beta);
	imshow("anh goc", src);
	imshow("anh co sau khi chinh do tuong phan va do sang", dst);
	waitKey(0);
	return 0;
```

CHỈNH ĐỘ SÁNG VÀ TƯƠNG PHẢN

PHÂN NGƯỠNG ẢNH NHỊ PHÂN

```
// Adaptive Threshold
#include <opencv2\core\core.hpp>
#include <opencv2\highgui\highgui.hpp>
#include <opencv2\imgproc\imgproc.hpp>

#include <iostream>

using namespace std;
using namespace cv;

int main()
{
	cout<<"Nhi phan anh voi nguong dong"<<endl;
	Mat src = imread("Thap_But.jpg", CV_LOAD_IMAGE_GRAYSCALE);
	Mat dst;
	adaptiveThreshold(src, dst, 255, CV_ADAPTIVE_THRESH_MEAN_C,
				CV_THRESH_BINARY, 35, 5);
	imshow("Anh xam goc", src);
	imshow("Anh nhi phan voi nguong dong", dst);
	waitKey(0);

	return 1;
}
```

PHÂN NGƯỠNG ẢNH NHỊ PHÂN

- **THRESH_BINARY**

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxVal} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

- **THRESH_BINARY_INV**

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxVal} & \text{otherwise} \end{cases}$$

- **THRESH_TRUNC**

$$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

- **THRESH_TOZERO**

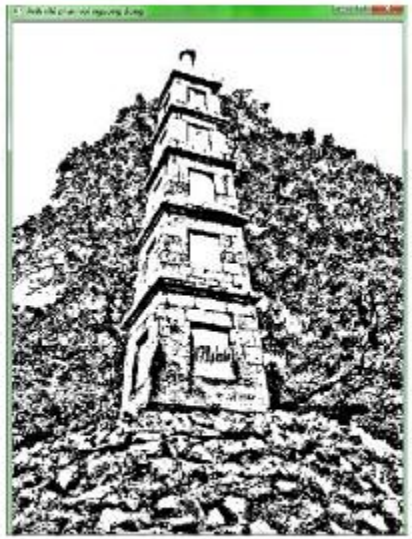
$$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

- **THRESH_TOZERO_INV**

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

PHÂN NGƯỠNG ẢNH NHỊ PHÂN

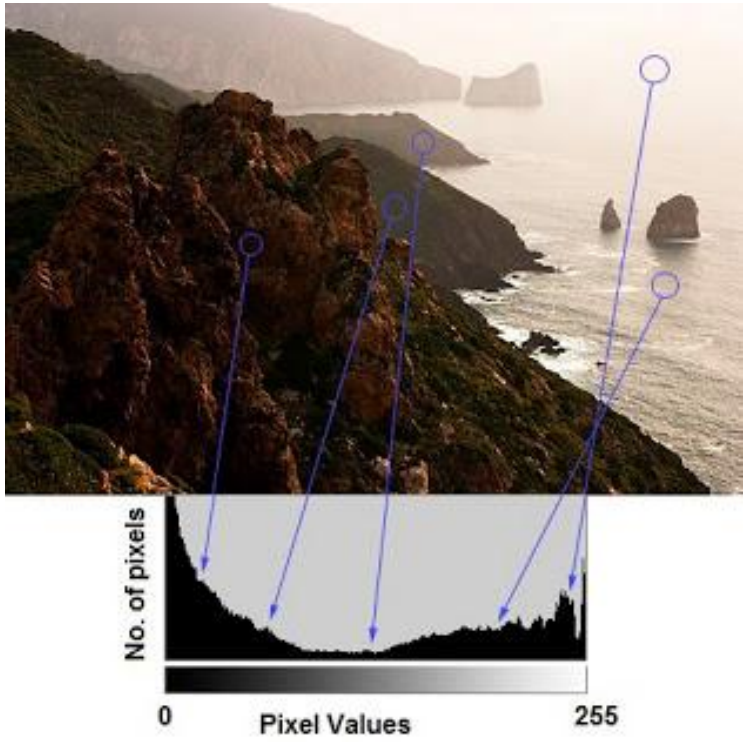
Ảnh xám



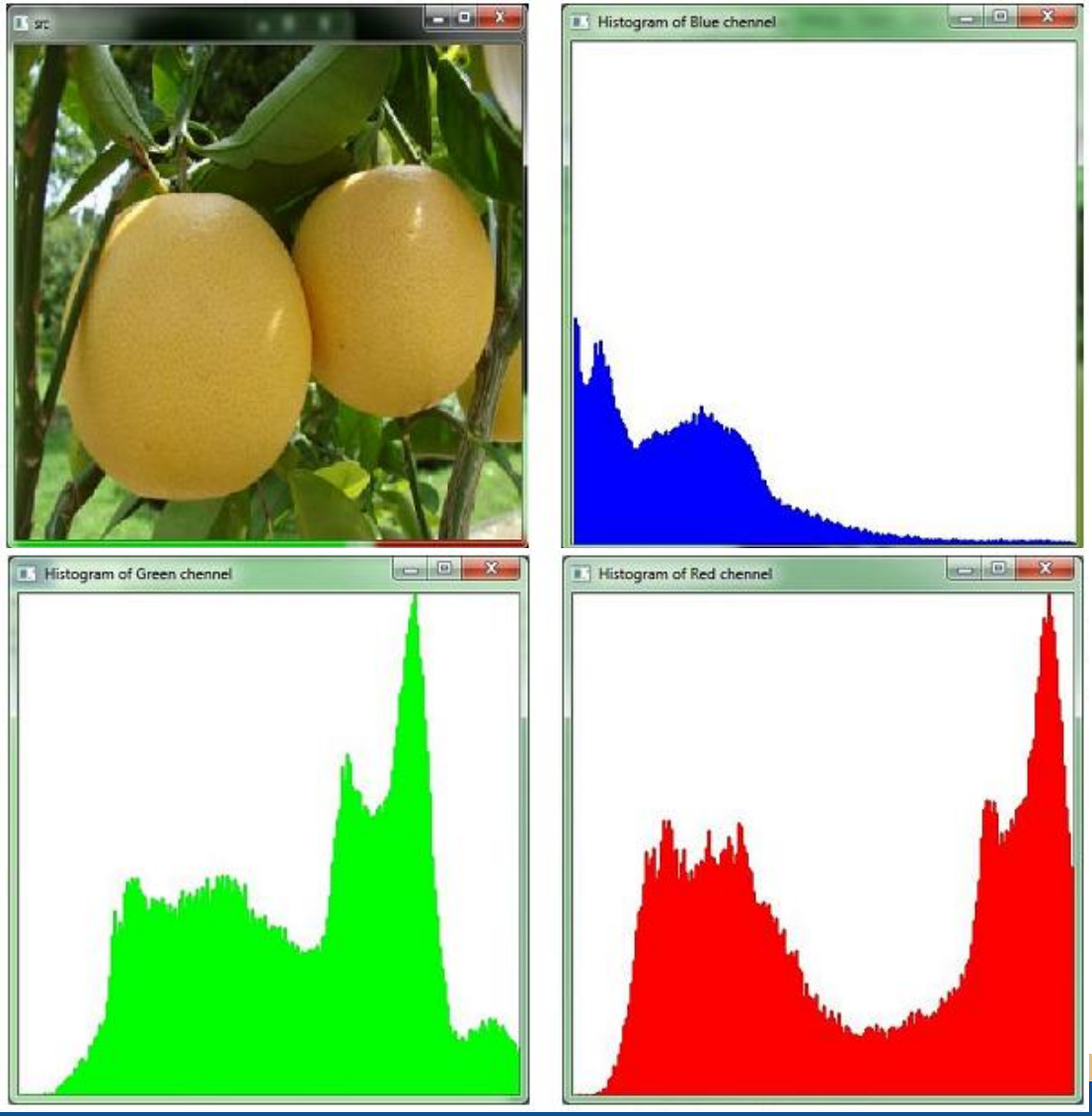
Ảnh nhị phân

TÍNH HISTOGRAM

```
#include <iostream>
#include <opencv2\highgui\highgui.hpp>
#include <opencv2\imgproc\imgproc.hpp>

using namespace std;
using namespace cv;

int main()
{
	std::cout<<"Tim histogram anh mau"<<std::endl;

	 Mat src = imread("buoi.jpg");
	 vector<Mat> img_rgb;
	 Mat img_r, img_g, img_b;
	 int w = 400, h = 400;
	 int size_hist = 255;
	float range[] = {0, 255};
	const float* hist_range = {range};

	 split(src, img_rgb);
	 calcHist(&img_rgb[0], 1, 0, Mat(), img_b, 1, &size_hist, &hist_range, true, false);
	 calcHist(&img_rgb[1], 1, 0, Mat(), img_g, 1, &size_hist, &hist_range, true, false);
	 calcHist(&img_rgb[2], 1, 0, Mat(), img_r, 1, &size_hist, &hist_range, true, false);

	 int bin = cvRound((double)w/size_hist);

	 Mat disp_r(w, h, CV_8UC3, Scalar( 255,255,255) );
	 Mat disp_g = disp_r.clone();
	 Mat disp_b = disp_r.clone();

	 normalize(img_b, img_r, 0, disp_b.rows, NORM_MINMAX, -1, Mat() );
	 normalize(img_g, img_g, 0, disp_g.rows, NORM_MINMAX, -1, Mat() );
	 normalize(img_r, img_b, 0, disp_r.rows, NORM_MINMAX, -1, Mat() );
	for( int i = 1; i < 255; i++ )
	{
	 line(disp_r, Point(bin*(i), h), Point(bin*(i), h - cvRound(img_r.at<float>(i))),
		Scalar(0, 0, 255), 2, 8, 0  );
	 line(disp_g, Point(bin*(i), h), Point(bin*(i), h - cvRound(img_g.at<float>(i))),
		Scalar( 0, 255, 0), 2, 8, 0  );
	 line(disp_b, Point( bin*(i), h), Point(bin*(i), h - cvRound(img_b.at<float>(i))),
		Scalar(255, 0, 0), 2, 8, 0  );
	}
	namedWindow("src", 0);
	imshow("src", src);
	imshow("Histogram of Blue chennel", disp_b);
	imshow("Histogram of Green chennel", disp_g);
	imshow("Histogram of Red chennel", disp_r);

	cv::waitKey(0);
	return 1;
}
```



TÍNH HISTOGRAM



CÂN BẰNG HISTOGRAM

```
#include <opencv2\core\core.hpp>
#include <opencv2\highgui\highgui.hpp>
#include <opencv2\imgproc\imgproc.hpp>

#include <iostream>

using namespace std;
using namespace cv;

int main()
{
	cout<<"Chuong trinh can bang histogram"<<endl;
	Mat src = imread("Cho_Ben_Thanh.jpg", CV_LOAD_IMAGE_COLOR);

	Mat hsv, disp;
	cvtColor(src, hsv, CV_BGR2HSV);
	vector<Mat> hsv_channels;
	// Tach hsv thanh 3 kenh mau
	split(hsv, hsv_channels);
	// Can bang histogram kenh mau v (value)
	equalizeHist(hsv_channels[2], hsv_channels[2]);
	// Tron anh
	merge(hsv_channels, hsv);
	// Chuyen doi hsv sang rgb de hien thi
	cvtColor(hsv, disp, CV_HSV2BGR);
	imshow("anh mau goc", src);
	imshow("anh mau sau khi can bang histogram", disp);
	waitKey(0);

}
```

CÂN BẰNG HISTOGRAM

PHÓNG TO VÀ THU NHỎ

- Biến đổi affine: p’(x’, y’) = Mp(x, y)

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

- Nếu $\gamma=\delta=0$ => phóng to hoặc thu nhỏ ảnh tùy theo giá trị của α,β
- Phép quay

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

- Vừa co giãn vừa quay

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

PHÓNG TO VÀ THU NHỎ

```
#include <opencv2\core\core.hpp>
#include <opencv2\highgui\highgui.hpp>
#include <opencv2\imgproc\imgproc.hpp>

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 = getRotationMatrix2D(center, angle, scale);
	warpAffine(src, dst, mat_rot, src.size());
	imshow("Anh goc", src);
	imshow("Anh sau phep bien doi", dst);
	waitKey(0);

	return 1;
}
```

PHÓNG TO VÀ THU NHỎ

Scale = 1.5 và angle = 45^{o}

LỌC ẢNH

$$M = \frac{1}{rows * cols}\begin{bmatrix} 1 & 1 & . & . & 1 \\ 1 & 1 & . & . & 1 \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & 1 & . & . & 1 \end{bmatrix}$$

Lọc Blur

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

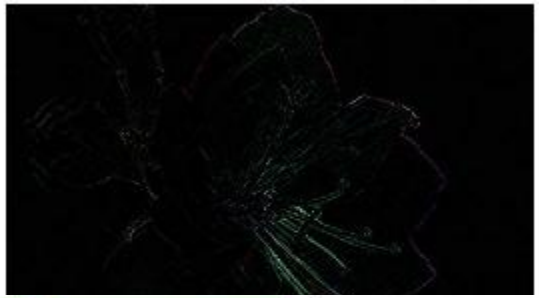
Lọc Laplace

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

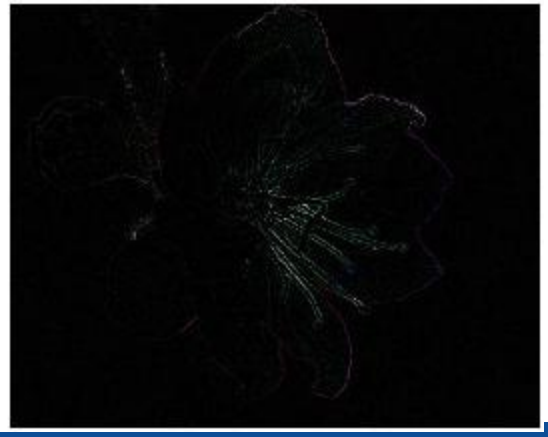
Lọc Sobel

LỌC ẢNH

Lọc blur: Làm mịn ảnh



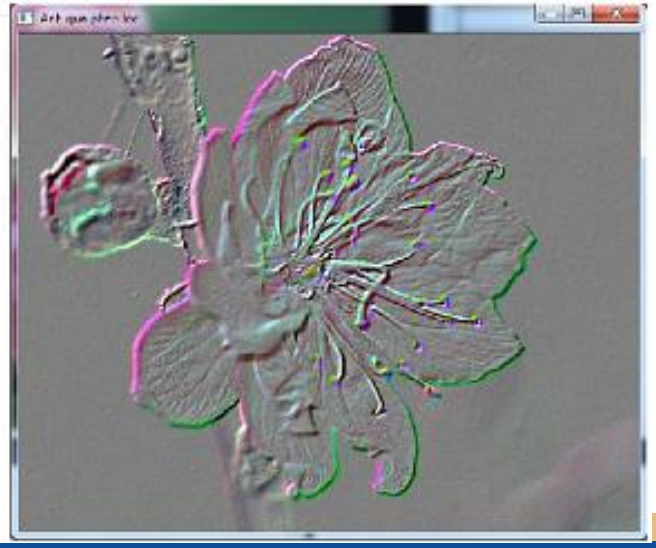
Lọc Sobel: Tách cạnh



Lọc Laplace

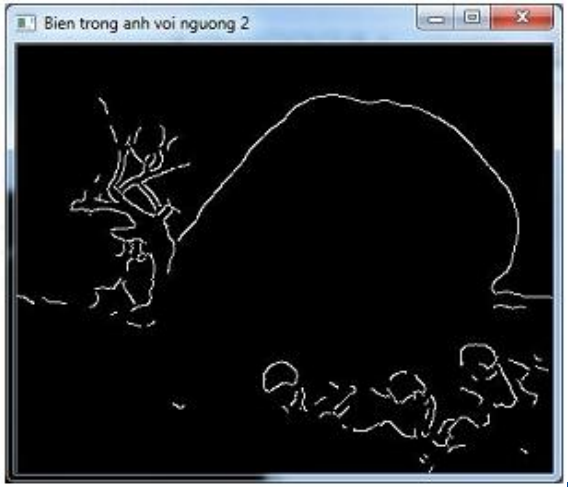
LỌC ẢNH

```
void main()
{
	Mat src = imread("HoaDao.jpg", CV_LOAD_IMAGE_COLOR);
	Mat dst;
	double m[3][3] = {  -1, -1,  0,
	                    -1,  0,  1,
	                     0,  1,  1
	                  };
	Mat M = cv::Mat(3,3,CV_64F, m, Point(-1,-1), 128.0);
	cv::filter2D(src, dst, src.depth(), M);
	imshow("Anh goc", src);
	imshow("Anh qua phep loc", dst);
	cv::waitKey(0);
}
```



TÌM BIÊN ẢNH

```
void main()
{
	Mat gray = cv::imread("TuoiTho.jpg", CV_LOAD_IMAGE_GRAYSCALE);
	Mat dst1, dst2;
	imshow("Anh xam", gray);
	GaussianBlur(gray, gray, Size(9,9), 2);
	double t1 = 30, t2 = 200;
	Canny(gray, dst1, t1, t2, 3, false);
	t1 = 100; t2 = 120;
	Canny(gray, dst2, t1, t2, 3, false);
	imshow("Bien trong anh voi nguong 1", dst1);
	imshow("Bien trong anh voi nguong 2", dst2);
	waitKey(0);
}
```



Thank You!