

OPENCV



Pedro Luiz de Paula Filho

O que é?

- Biblioteca desenvolvida pela Intel (2000) possui + de 500 funções (**OPENCV - Open Computer Vision Library**)
- Idealizada para tornar a visão computacional acessível a usuários e programadores
- Disponível com o código fonte e os executáveis otimizados.
- A biblioteca está dividida em cinco grupos de funções:
 - ▣ Processamento de imagens;
 - ▣ Análise estrutural;
 - ▣ Análise de movimento e rastreamento de objetos;
 - ▣ Reconhecimento de padrões;
 - ▣ Calibração de câmera e reconstrução 3D.


O que é?

- Na versão 1.0 era escrito em C
- A partir da versão 2.0 os algoritmos são escritos em C++, mas com wrappers em Python e Java
- Roda em :
 - ▣ **Desktop** : (Windows, Linux, Android, MacOS, FreeBSD, OpenBSD)
 - ▣ **Mobile**: (Android, Maemo, iOS)
- A partir de 2010 utiliza CUDA
- A partir de 2011 utiliza OpenCL

Por onde começar?

- ❑ Site Oficial: <http://opencv.org/>
- ❑ Download: <http://opencv.org/downloads.html>
- ❑ Instalação:
[http://docs.opencv.org/doc/tutorials/introduction/table of content introduction/table of content introduction.html](http://docs.opencv.org/doc/tutorials/introduction/table_of_content_introduction.html)
- ❑ Documentação:
<http://opencv.org/documentation.html>

Instalando CodeBlocks no windows

- Faça download da ultima versão do OpenCv ([link](#))
 - ▣ Descompacte na pasta (ex. c:\opencv249)
- Faça download do cmake e o instale ([link](#))
- Coloque no path (variáveis de ambiente)
 - ▣ **Compilador:** C:\Program Files (x86)\CodeBlocks\MinGW\bin
- Rode o cmake (compilar openCv para sua maquina)
 - ▣ Acerte os diretórios 
 - ▣ Aperte “configure”
 - ▣ Aperte “generate”

Where is the source code:	C:/opencv249/opencv/sources
Where to build the binaries:	C:/opencv249/opencv/build4

Instalando CodeBlocks no windows

- ❑ Vá no diretório gerado pelo cmake →
“C:\opencv249\opencv\build4”
- ❑ Rode o “mingw32-make.exe” (demorado)
- ❑ Rode “mingw32-make install”

Instalando CodeBlocks no windows

- Entre no codeblocks
 - ▣ File → new Project → Console Application
 - ▣ Botão direito no projeto → Build Options
 - ▣ Search directories → Compiler → Add
 - C:\opencv249\opencv\build4\install\include
 - ▣ Search directories → Linker → Add
 - C:\opencv249\opencv\build4\install\x86\mingw\lib
 - ▣ Linker settings → Link libraries → Add
 - Adicione “todos” os programas do diretório
C:\opencv249\opencv\build4\install\x86\mingw\lib

Olá Mundo!

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>

using namespace cv;

int main() {
    Mat image;
    image = cv::imread("c:/cap.jpg");
    namedWindow( "Teste", CV_WINDOW_AUTOSIZE );
    imshow( "Teste", image);
    waitKey(0);
    return 0;
}
```

- Estrutura responsável pelo armazenamento de “imagens”
- Faz a leitura da imagem e atribui ao Mat
- Cria uma janela de nome “Teste”
- Apresenta a image (Mat) na janela “Teste”
- Para o processamento até ser apertada alguma tecla



Classe Mat

- Classe que representa um vetor n-dimensional
- Serve para armazenar vetores, matrizes, imagens coloridas e em tons de cinza, voxels, vetor de campos, nuvem de pontos, histogramas, ...
- Alguns atributos interessantes:
 - ▣ rows → informa a quantidade de linhas do Mat
 - ▣ cols → informa a quantidade de colunas do Mat
 - ▣ data → armazena a informação da imagem
 - ▣ Channels → quantidade de canais
 - ▣ at <Vec3b>(lin, col) → retorna um pixel no formato Vec3b

Classe Mat

- Criar uma Matriz Bidimensional
 - ▣ `double m[2][2] = {{1.0, 2.0}, {3.0, 4.0}};`
 - ▣ `Mat M(2, 2, CV_32F, m);`
- Copiar uma matriz M1 para M2
 - ▣ `Mat M2 = M1.Clone();`
 - ▣ `Mat M2; M1.copyTo(M2);`
- Matriz transposta \rightarrow `Mat M2 = M1.t();`
- Matriz inversa \rightarrow `Mat M2 = m1.inv();`
- Multiplica matriz \rightarrow `Mat M3 = M1 * M2;`

```
#include <stdio.h>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
using namespace cv;
int main() {
    Mat matriz(10,10,CV_8UC1);
    //Mat matriz(10,10,CV_8UC3);
    int canais = matriz.channels();
    printf("%d\n\n", canais);
    for(int i = 0; i < matriz.rows; i++) {
        for(int j = 0; j < matriz.cols; j += canais) {
            matriz.at<int>(i,j) = i * 20;
            printf("%03d ", matriz.at<int>(i,j));
        }
        printf ("\n");
    }
    waitKey(0);
    return 0;
}
```

imread

- Carrega uma imagem de um arquivo
- Sintaxe:

Mat imread(const string& filename, int flags=1)

- ▣ **Filename:** nome do arquivo a ser lido
- ▣ **Flags:** Especifica o tipo da cor ao carregar a imagem
 - CV_LOAD_IMAGE_ANYDEPTH
 - Carrega a imagem como foi gravada (16 bits ou 32 bits).
 - CV_LOAD_IMAGE_COLOR
 - Carrega a imagem como colorida
 - CV_LOAD_IMAGE_GRAYSCALE
 - Carrega a imagem como em tons de cinza
 - As opções podem ser combinadas
 - `image = cv::imread("c:/cap.jpg", CV_LOAD_IMAGE_ANYCOLOR | CV_LOAD_IMAGE_ANYDEPTH);`

namedWindow

- Cria uma janela

- Sintaxe:

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

- ▣ **Winname:** nome da janela a ser criada

- ▣ **Flags:** característica da abertura da janela

- **WINDOW_NORMAL** → Permite ao usuario alterar o tamanho da janela

- **WINDOW_AUTOSIZE** → abre a janela do tamanho da imagem e não permite alteração do tamanho dela

imshow

- Mostra uma imagem dentro de uma janela
- Sintaxe:
void imshow(const string& winname, InputArray mat)
 - ▣ **Winname:** nome da janela
 - ▣ **Mat:** imagem a ser mostrada

waitKey

- Aguarda que uma tecla seja pressionada
- Sintaxe:

int waitKey(int delay=0)

- delay

- Aguarda “n” milissegundos por uma tecla, se nada for teclado da continuidade ao processo.
- Se “n” for zero o tempo de parada é infinito

Conversão de espaço de cores

- Quando carrega-se uma imagem com o `imread`, ela vem no formato BGR (*blue, green e red*)
- Para converte-la para outro espaço de cores usa-se o comando `cvtColor`

- Sintaxe:

`void cvtColor(InputArray src, OutputArray dst, int code, int dstCn=0)`

- ▣ **Code:** representação da transformação desejada
- ▣ **dstCn:** numero de canais de cor, na imagem destino

Code : Conversão de espaço de cores

- RGB \leftrightarrow GRAY
 - ▣ CV_BGR2GRAY, CV_RGB2GRAY, CV_GRAY2BGR, CV_GRAY2RGB
- RGB \leftrightarrow XYZ
 - ▣ CV_BGR2XYZ, CV_RGB2XYZ, CV_XYZ2BGR, CV_XYZ2RGB
- RGB \leftrightarrow YCrCb
 - ▣ CV_BGR2YCrCb, CV_RGB2YCrCb, CV_YCrCb2BGR, CV_YCrCb2RGB
- RGB \leftrightarrow HSV
 - ▣ CV_BGR2HSV, CV_RGB2HSV, CV_HSV2BGR, CV_HSV2RGB
- RGB \leftrightarrow HLS
 - ▣ CV_BGR2HLS, CV_RGB2HLS, CV_HLS2BGR, CV_HLS2RGB
- RGB \leftrightarrow CIE L*a*b*
 - ▣ CV_BGR2Lab, CV_RGB2Lab, CV_Lab2BGR, CV_Lab2RGB
- RGB \leftrightarrow CIE L*u*v*
 - ▣ CV_BGR2Luv, CV_RGB2Luv, CV_Luv2BGR, CV_Luv2RGB

```
#include <opencv2/opencv.hpp>
using namespace cv;
int main() {
    Mat original, nova;
    original = imread("c:/flores.jpg");
    namedWindow( "Original", CV_WINDOW_AUTOSIZE );
    imshow( "Original", original);
    cvtColor(original, nova, CV_BGR2HSV);
    namedWindow( "Convertida", CV_WINDOW_AUTOSIZE );
    imshow( "Convertida", nova);
    waitKey(0);
    return 0;
}
```

split

- Divide um vetor multi-canal em vários vetores de um canal
 - ▣ Transforma uma imagem 24 bits (RGB) em 3 de 8 bits
- Sintaxe:

void split(InputArray m, OutputArrayOfArrays mv)

```
vector<Mat> canaisCor(3);  
split(nova, canaisCor);  
namedWindow( "Canal1", CV_WINDOW_AUTOSIZE );  
imshow( "Canal1", canaisCor[0] );  
namedWindow( "Canal2", CV_WINDOW_AUTOSIZE );  
imshow( "Canal2", canaisCor[1] );  
namedWindow( "Canal3", CV_WINDOW_AUTOSIZE );  
imshow( "Canal3", canaisCor[2] );
```

imwrite

- Salva uma imagem para um arquivo específico
- Sintaxe:
 - ▣ `bool imwrite(const string& filename, InputArray img, const vector<int>& params=vector<int>())`

```
imwrite( "c:/opencv249/convertida.jpg", nova );  
imwrite( "c:/opencv249/convertida.png", nova );  
imwrite( "c:/opencv249/convertida.bmp", nova );
```

Copiar uma imagem pixel-a-pixel

```
#include <opencv2/opencv.hpp>
using namespace cv;
int main() {
    Mat original;
    original = imread("c:/flores.jpg");
    Mat nova(original.rows, original.cols, CV_8UC3, Scalar( 0,0,0));
    for (int l=0; l < original.rows; l++)
        for (int c=0; c < original.cols; c++)
            if (original.channels() == 3) {
                nova.at<Vec3b>(l,c)[0] = original.at<Vec3b>(l,c)[0];
                nova.at<Vec3b>(l,c)[1] = original.at<Vec3b>(l,c)[1];
                nova.at<Vec3b>(l,c)[2] = original.at<Vec3b>(l,c)[2];
            }
    namedWindow( "Original", CV_WINDOW_AUTOSIZE );
    imshow( "Original", original);
    namedWindow( "Convertida", CV_WINDOW_AUTOSIZE );
    imshow( "Convertida", nova);
    waitKey(0);
    return 0;
}
```

Histograma

- Para se fazer um histograma deve-se:
 - ▣ Carregar a imagem
 - ▣ Segmentar a imagem em 3 canais (split)
 - ▣ Calcular o histograma para cada canal, através da função `calHist`
 - ▣ Normaliza o histograma `minMaxLoc`
 - ▣ Plotar os histogramas usando linhas

calcHist

- Calcula o histograma de um conjunto de dados

- Sintaxe:

```
void calcHist(const Mat* images, int nimages, const int*  
channels, InputArray mask, OutputArray hist, int dims, const int*  
histSize, const float** ranges, bool uniform=true, bool  
accumulate=false)
```


- ▣ Images → Fonte de dados
- ▣ Hist → histograma de saída
- ▣ Dims → Dimensionalidade do histograma
- ▣ Histsize → Tamanho do histograma em cada dimensão
- ▣ Ranges → Vetor com a faixa de cada histograma

Exemplo de Histograma

```
MatND geraHistograma (const Mat imagem, Mat *Histograma){
    MatND hist;
    int channels[] = {0, 1};
    int histSize[1];
    float hranges[2] = {0, 255};
    const float* ranges[1];
    double maxVal=0, minVal=0;
    ranges[0] = hranges;

    histSize[0] = 256;
    calcHist(&imagem, 1, channels, Mat(), hist, 1, histSize, ranges);
    /*
    calcHist(&imagem,
    // Normaliza o histograma
    minMaxLoc(hist, &minVal, &maxVal, 0, 0);
    // define o maior ponto com 90% do tamanho
    int hpt = static_cast<int>(0.9*histSize[0]);
    // Desenha uma linha para cada faixa
    for( int h = 0; h < histSize[0]; h++ ) {
        float binVal = hist.at<float>(h);
        int intensity = static_cast<int>(binVal*hpt/maxVal);
        line(*Histograma, Point(h, histSize[0]),
            Point(h, histSize[0]-intensity),
            Scalar::all(0));
    }
    return hist;
}
```

```
calcHist(&imagem,
    1, // histograma de uma unica imagem
    channels, //canal usado
    Mat(), //sem usar mascara
    hist, // histograma de saida
    1, //define histograma 1D
    histSize, //quantidade faixas do histograma
    ranges //
);
```



Exemplo de Histograma

```
int main() {
    Mat original, histograma(256,256,CV_8U, Scalar(255));
    original = imread("c:/flores.jpg", CV_LOAD_IMAGE_GRAYSCALE);
    MatND histo = geraHistograma(original, &histograma);
    //mostra o vetor do histograma
    for (int i=0; i<256; i++)
        cout << "Valor " << i << " = " <<
            histo.at<float>(i) << endl;
    namedWindow( "Source", 1 );
    imshow( "Source", original );
    namedWindow( "Histograma", 1 );
    imshow( "Histograma", histograma );
    waitKey();
}
```

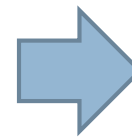
Outro Exemplo de Histograma

```
Mat original;  
original = imread("c:/flores.jpg");  
// Separa a imagem em 3 canais (B,G,R)  
vector<Mat> bgr_planes;  
split( original, bgr_planes );  
int histSize = 256;  
float range[] = { 0, 256 } ;  
const float* histRange = { range };  
bool uniform = true; bool accumulate = false;  
Mat b_hist, g_hist, r_hist;  
// Calcula os histogramas de cada canal  
calcHist(&bgr_planes[0],1,0,Mat(),b_hist,1,  
        &histSize,&histRange,uniform,accumulate);  
calcHist(&bgr_planes[1],1,0,Mat(),g_hist,1,  
        &histSize,&histRange,uniform,accumulate);  
calcHist(&bgr_planes[2],1,0,Mat(),r_hist,1,  
        &histSize,&histRange,uniform,accumulate);
```

Outro Exemplo de Histograma

```
// Desenha os histogramas B, G, R
int hist_w = 512; int hist_h = 400;
int bin_w = cvRound( (double) hist_w/histSize );
Mat histImage( hist_h, hist_w, CV_8UC3, Scalar( 0,0,0) );
// Normaliza o resultado de o até o tamanho de linhas do histograma
normalize(b_hist, b_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat() );
normalize(g_hist, g_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat() );
normalize(r_hist, r_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat() );
// Desenha cada canal
for( int i = 1; i < histSize; i++ ) {
    line(histImage, Point(bin_w*(i-1), hist_h-cvRound(b_hist.at<float>(i-1))) ,
          Point(bin_w*(i), hist_h-cvRound(b_hist.at<float>(i))),
          Scalar(255, 0, 0), 2, 8, 0);
    line(histImage, Point(bin_w*(i-1), hist_h-cvRound(g_hist.at<float>(i-1))),
          Point(bin_w*(i), hist_h - cvRound(g_hist.at<float>(i)) ),
          Scalar(0, 255, 0), 2, 8, 0 );
    line(histImage, Point(bin_w*(i-1), hist_h-cvRound(r_hist.at<float>(i-1))),
          Point(bin_w*(i), hist_h - cvRound(r_hist.at<float>(i)) ),
          Scalar(0, 0, 255), 2, 8, 0 );
}
namedWindow( "Source", 1 );
imshow( "Source", original );
namedWindow( "Histograma", 1 );
imshow( "Histograma", histImage );
```

equalizeHist



- Equaliza o histograma de uma imagem em tons de cinza, normaliza o brilho e aumenta o contraste
- Sintaxe:

`void equalizeHist(InputArray src, OutputArray dst)`

```
Mat original, equalizada, histograma(256,256,CV_8U, Scalar(255));  
Mat histograma2(256,256,CV_8U, Scalar(255));  
original = imread("c:/flores.jpg", CV_LOAD_IMAGE_GRAYSCALE);  
equalizeHist(original, equalizada);  
MatND histo      = geraHistograma(original, &histograma);  
MatND histoEqua  = geraHistograma(equalizada, &histograma2);  
namedWindow( "Original", 0 );  
imshow( "Original", original );  
namedWindow( "Histograma", 0 );  
imshow( "Histograma", histograma );  
namedWindow( "Equalizada", 0 );  
imshow( "Equalizada", equalizada );  
namedWindow( "Histograma Equalizado", 0 );  
imshow( "Histograma Equalizado", histograma2 );
```

Exercício

- Faça um programa que peça ao usuário o nome de um arquivo (imagem) e uma opção de menu, e faça a conversão das imagens para o espaço de cor selecionado.
 - ▣ Mostre também cada um dos canais do espaço de cor escolhido.
 - ▣ Gere os histogramas de cada um dos canais
 - ▣ Salve todas as imagens geradas