数字图像处理

--- Something about OpenCV and Matlab

助教：陈舒雅
邮箱：413056870@qq.com
手机：18868130192

开发工具

- Visual Studio + OpenCV
- Matlab
- Python
- …

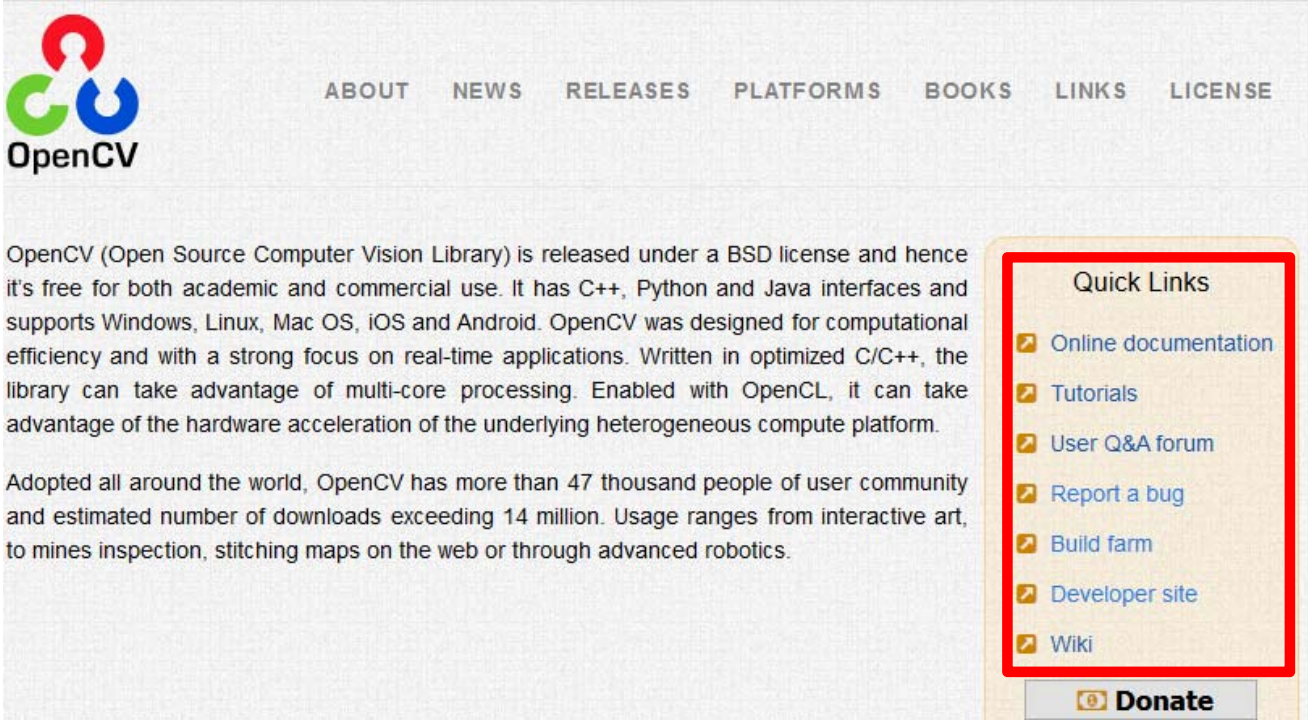
概述
搭建开发环境
基本结构
基本应用
常见应用

OpenCV 概述

- OpenCV是Intel开源计算机视觉库
- 由一系列C函数和C++类构成，实现了图像处理和计算机视觉方面很多的通用算法

- 特点：
  - 跨平台通用：Windows,Linux,Mac
  - Free
  - 速度快，使用方便

- 主页：https://opencv.org/



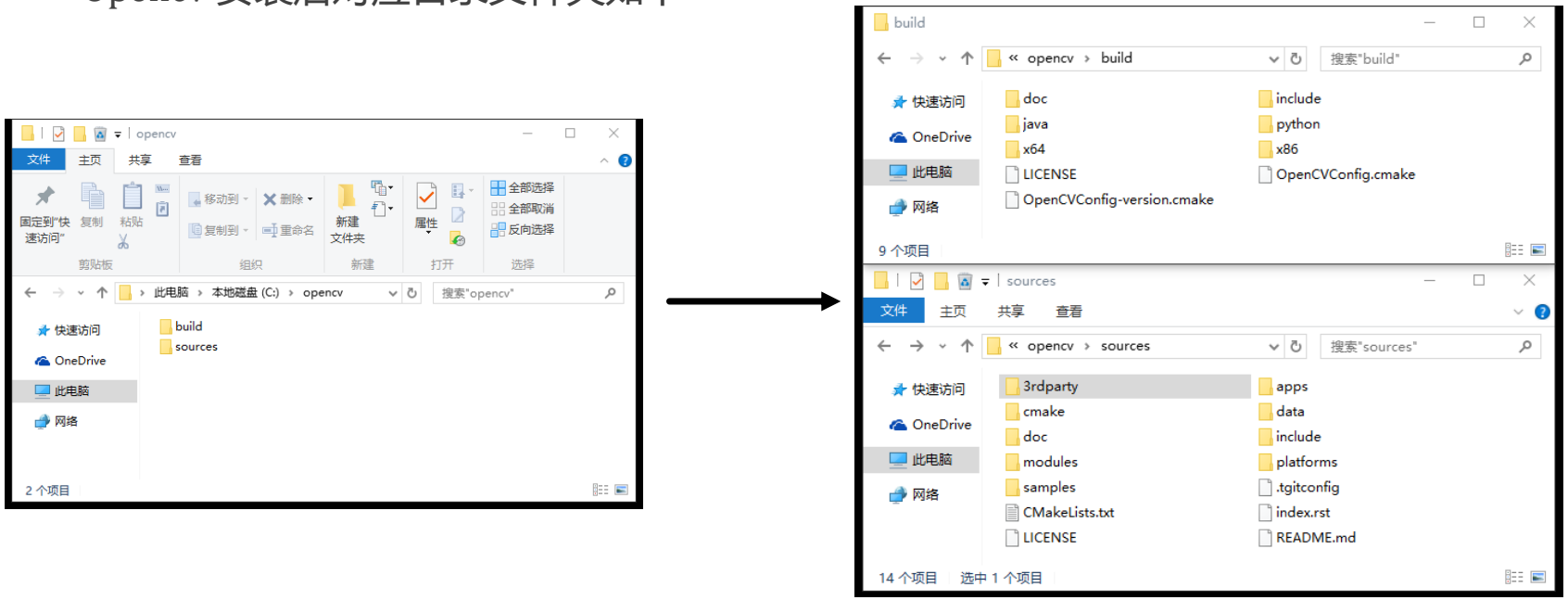
OpenCV 结构与内容

- CV: 基本的图像处理函数和高级的计算机视觉算法
- ML: 机器学习库
- HighGUI: 图像和视频的输入输出函数
- CXCore: 基本的数据结构和算法

- CvAux: 即将淘汰的算法和新出现的实验性算法和函数

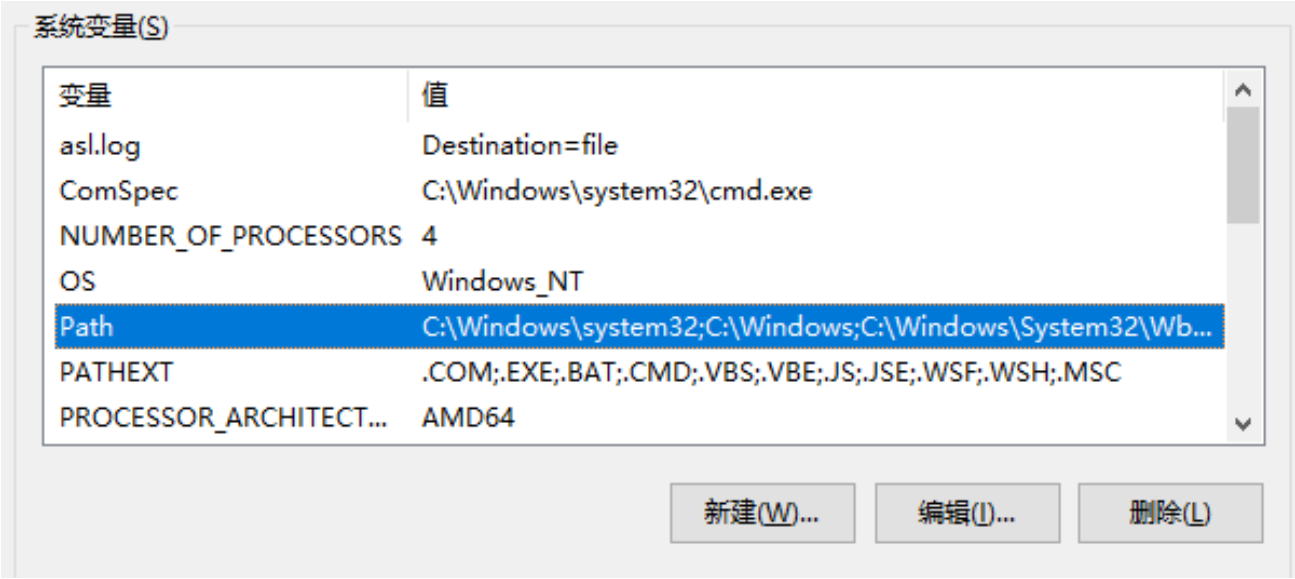
搭建VS+OpenCV 开发环境_01

- 1.1 下载并安装visual studio 以及 Opencv(以Visual studio2013+Opencv2.4.9为例):
  - 版本根据自己电脑的配置选定
  - x86代表32位操作系统 x64代表64位操作系统
  - Opencv 安装后对应目录文件夹如下



搭建VS+OpenCV 开发环境_01

- 1.2 设置环境变量:
  - “计算机”->”属性”->”高级系统设置”->”环境变量”
  - 添加路径到Path: “(opencv目录)\build\x86\vc12\bin;”
  - 其中 vcxx 对应vs版本：VS2010,VS2012,VS2013 对应vc10,11,12



搭建VS+OpenCV 开发环境_02

- 2.1 配置VS:
  - 新建空项目，进入属性页
  - “VC++目录”->”库目录”：添加路径：” D:\opencv\build\x86\vc12\lib;”
  - …“包含目录”： 添加路径：
    - D:\opencv\build\include\opencv2;
    - D:\opencv\build\include\opencv;
    - D:\opencv\build\include;
  - “链接器”->”输入”->”附加依赖项”：添加lib如以下列表：

opencv_ml249d.lib
opencv_calib3d249d.lib
opencv_contrib249d.lib
opencv_core249d.lib
opencv_features2d249d.lib
opencv_flann249d.lib
opencv_gpu249d.lib
opencv_highgui249d.lib
opencv_imgproc249d.lib
opencv_legacy249d.lib
opencv_objdetect249d.lib
opencv_ts249d.lib
opencv_video249d.lib
opencv_nonfree249d.lib
opencv_ocl249d.lib
opencv_photo249d.lib
opencv_stitching249d.lib
opencv_superres249d.lib
opencv_videostab249d.lib

测试

main.cpp

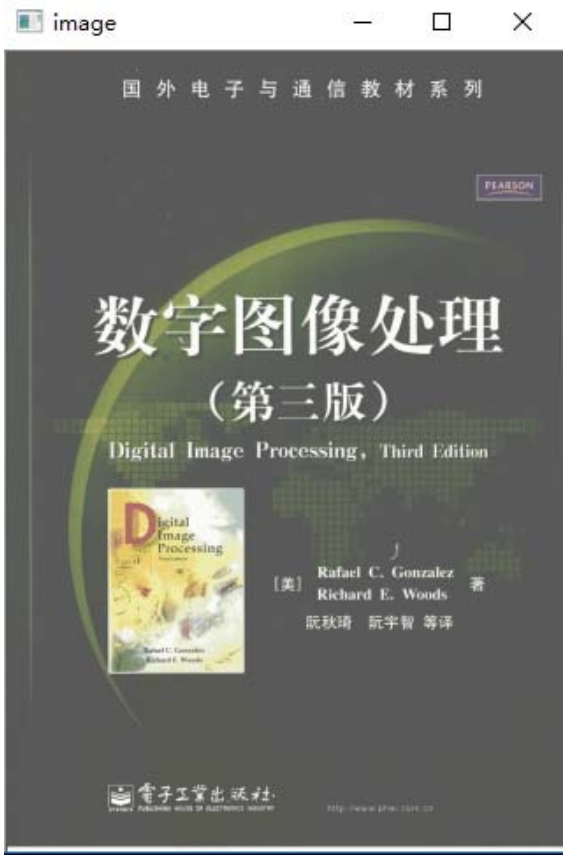
SGM | (全局范围) | main()

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

using namespace cv;


int main()
{
    // 读入一张图片
    Mat img = imread("D:\\pic\\process.jpg");
    // 创建一个名为 "image"的窗口
    namedWindow("image");
    // 在窗口中显示原画
    imshow("image", img);
    // 等待6000 ms后窗口自动关闭
    waitKey(6000);
}
```

解决方案资源管理器

搜索解决方案资源管理器(Ctrl+;)

- 解决方案"SGM"(1 个项目)
  - SGM
    - 头文件
    - 外部依赖项
    - 源文件
      - main.cpp
    - 资源文件



OpenCV 命名规则

- （1） 函数名
- （2） 矩阵数据类型
- （3） 图像数据类型

OpenCV 命名规则

- （1）函数名:
  - 数据结构几乎都是Cv开头，函数为cv
  - cvActionTargetMod(...)
    - Action = 核心功能（core functionality） (e.g. set, create)
    - Target = 目标图像区域（target image area） (e.g. contour, polygon)
    - Mod = （可选的）调整语（optional modifiers） (e.g. argument,type)
  - cvGet2D、cvCreateImage、cvNamedWindow

OpenCV 命名规则

- （2） 矩阵数据类型:
  - CV_<bit_depth>(S|U|F)C<number_of_channels>
    - S = 符号整型 U = 无符号整型 F = 浮点型

    E.g.:
    - CV_8UC1 是指一个8位无符号整型单通道矩阵,

      CV_32FC2是指一个32位浮点型双通道矩阵
    - CV_8UC1 CV_8SC1 CV_16U C1 CV_16SC1
      CV_8UC2 CV_8SC2 CV_16UC2 CV_16SC2
      CV_8UC3 CV_8SC3 CV_16UC3 CV_16SC3
      CV_8UC4 CV_8SC4 CV_16UC4 CV_16SC4
      CV_32SC1 CV_32FC1 CV_64FC1
      CV_32SC2 CV_32FC2 CV_64FC2
      CV_32SC3 CV_32FC3 CV_64FC3
      CV_32SC4 CV_32FC4 CV_64FC4

OpenCV 命名规则

- （3） 图像数据类型:
  - IPL_DEPTH_<bit_depth>(S|U|F)

    E.g.:

    - IPL_DEPTH_8U 图像像素数据是8位无符号整型

      IPL_DEPTH_32F图像像素数据是32位浮点型

    - IPL_DEPTH_1U　　IPL_DEPTH_8U

      IPL_DEPTH_8S　　IPL_DEPTH_16U

      IPL_DEPTH_16S　　IPL_DEPTH_32F

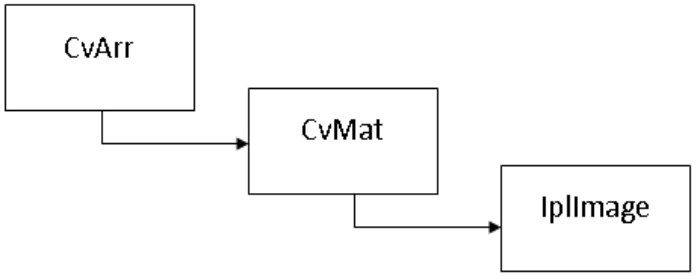
      IPL_DEPTH_64F

OpenCV 常用数据结构

结构	成员	意义
CvPoint	int x,y	图像中的点
CvPoint2D32f	float x,y	二维空间中的点
CvPoint3D32f	float x,y,z	三维空间中的点
CvSize	int width,height	图像的尺寸
CvRect	int x,y(左上坐标),width,height	图像的部分区域
CvScalar	double val[4]	RGBA 值

OpenCV 基本图像数据结构

- IplImage
  - 来源于Intel针对图像处理的函数库Intel Image Processing Library (IPL)。
  - 表示一个图像的结构体，也是从OpenCV1.0到目前最为重要的一个结构；
  - 基于 C 语言接口，用户来负责处理内存分配和解除分配。
- cv::Mat
  - OpenCV2.0封装的一个C++类，用来表示一个图像，自动内存管理
  - 和IplImage表示基本一致，但是Mat还添加了一些图像函数



- CvMat
  - 矩阵结构，由宽度（width）、高度（height）、类型（type）、行数据长度（step 用字节数来度量）和一个指向数据的指针构成。
- CvArr
  - 往往出现在函数原型中，作为参数出现，用来表示函数能处理任意的数组指针类型。

IplImage 数据结构

- typedef struct _IplImage {

int nSize;	/* IplImage大小 */
int ID; void *imageId; struct _IplTileInfo *tileInfo; int alphaChannel; char colorModel[4]; char channelSeq[4];	/* 版本 (=0)*/ /* 在 OpenCV中必须置NULL */ /* 同上*/ /* 被OpenCV忽略 */
int nChannels;	/* 大多数OPENCV函数支持1,2,3 或 4 个通道 */
int depth;	/* 像素的位深度: IPL_DEPTH_8U*/
int dataOrder;	/* 0 - 交叉存取颜色通道, 1 - 分开的颜色通道*/
int origin;	/* 0 - 顶—左结构,1 - 底—左结构 (BMP风格) */

int width; int height;	/* 图像宽\高像素数 */
int imageSize;	/* 图像数据大小，单位字节 (在交叉存取格式下imageSize=image->height*image->widthStep)*/
int widthStep;	/* 图像固定的行大小，以字节为单位，使处理过程更高效，因此行与行之间会有冗余字节*/
char *imageData;	/* 指向排列的图像数据 */
int align;	/* 图像行排列 (4 or 8). 用widthStep 代替 */
int BorderMode[4]; int BorderConst[4];	/* 边际结束模式, 被忽略*/
struct _IplROI *roi; struct _IplImage *maskROI;	/* 图像感兴趣区域. 当该值非空只对该区域进行处理 */
char *imageDataOrigin;	/* 指针指向一个不同的图像数据结构，是为了纠正图像内存分配准备的 */

}IplImage;

- int depth;
  - 图像通用数据类型：
  - IPL_DEPTH_<bit_depth>(S|U|F)

  IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16S, IPL_DEPTH_32S,IPL_DEPTH_32F ,IPL_DEPTH_64F
- int nChannels;
  - 图像的通道数
  - 例：
  - 灰度图为1个通道
  - 复值图像为2个通道，如傅里叶变换,一个通道为实数，一个通道为虚数
  - BGR图像为3个通道
  - BGRA图像是4个通道，RGB加A通道，A为alpha通道，表示透明度，赋值0到1，或者0到255，表示透明到不透明。 PNG图像是一种典型的4通道图像。
  - 大多数OpenCV函数支持1~4个通道 。

- int dataOrder;
  - 图像数据的存储格式
  - 0 :交叉存取颜色通道 1:分开存取颜色通道
  - OpenCV函数只支持交叉存取的图像。

B	G	R	B	G	R	...

- struct _IplROI *roi;
  - ROI：Region Of Interest（感兴趣区域）
    - roi==NULL：整幅图像参与运算
    - roi！=NULL：ROI区域代替图像参加运算
  - ROI的操作：
    - cvSetImageROI()：设置ROI区域
    - cvResetImageROI()：取消ROI区域
    - cvGetImageROI()：得到ROI区域

基本应用

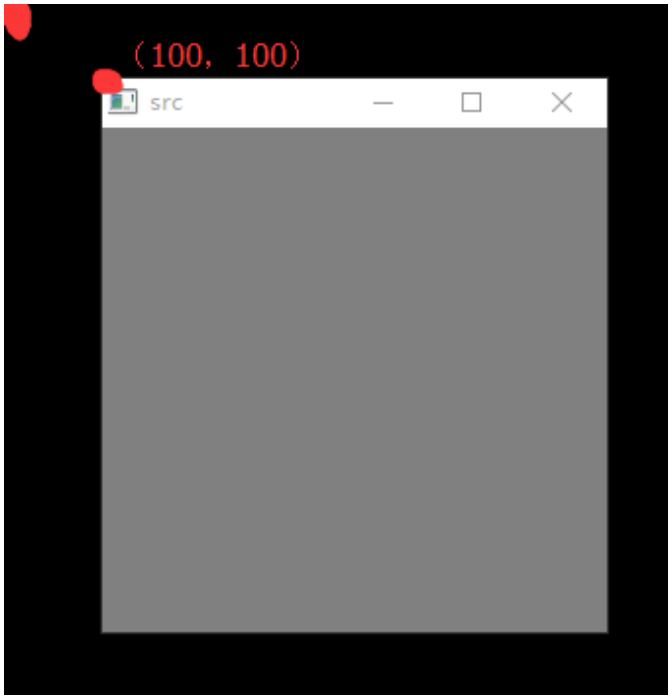
- GUI(Graphical User Interface)基本指令
- 图像处理基本指令

GUI基本指令

- （1） 创建和定位一个新窗口:
- （2） 载入图像:
- （3） 显示图像:
- （4） 关闭窗口:
- （5） 改变窗口大小:

(1) 创建和定位一个新窗口:

```
cvNamedWindow("src", CV_WINDOW_AUTOSIZE);
cvMoveWindow("src", 100, 100);
// offset from the corner of the screen
```



cvNamedWindow():

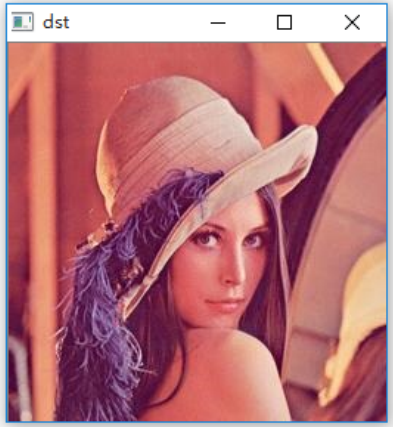
第一个参数表示新窗口的名称，显示在窗口的顶部，同时用作highgui中其他函数调用该窗口的句柄。

第二个参数是一个flags，设为0表示用户随意调整窗口大小，设为CV_WINDOW_AUTOSIZE，HighGUI会根据图像调整窗口大小。

（2） 载入图像:从文件中读入图像

IplImage* img =cvLoadImage(filename,flag);

```
#include <opencv.hpp>
#include <iostream>

using namespace std;
using namespace cv;

int main()
{
    cvNamedWindow("src", CV_WINDOW_AUTOSIZE);
    cvNamedWindow("dst", CV_WINDOW_AUTOSIZE);

    cvMoveWindow("src", 10, 10);

    IplImage * test_img = cvLoadImage("D:/lena.jpg",1);

    cvShowImage("dst", test_img);

    cvWaitKey();

    cvDestroyWindow("src");
    cvDestroyWindow("dst");
    cvReleaseImage(&test_img);

    return 0;
}
```



- Opencv默认将读入图像强制转换为一幅三通道彩色图像，可以按以下方法修改读入方式：
  - img=cvLoadImage(filename,flag)；
  - flag：
  - >0将读入的图像强制转换为一幅三通道彩色图像
  - =0将读入的图像强制转换为一幅单通道灰度图像
  - <0读入的图像通道数与 所读入的文件相同
- 支持的图像格式：bmp，dib，jpeg，jpg，jpe，png，pbm，pgm，ppm，sr，ras，tiff，tif

- （3） 显示图像:
  - cvShowImage(“dst",img);
  - 该函数可以显示彩色或灰度的字节型/浮点型图像。字节型图像像素值范围为[0-255]；浮点型图像像素值范围为[0-1]。彩色图像的三色元素按BGR（蓝-绿-红）顺序存储。
- （4） 关闭窗口:
  - cvDestroyWindow(“dst");
- （5） 改变窗口大小:
  - cvResizeWindow(“dst",100,100);

图像处理基本指令

- （1） 图像缩放:
- （2） 图像旋转:
- （3） 像素点操作:

(1) 图像缩放:

放大

缩小

（1） 图像缩放:

- void cvResize(const CvArr* src, CvArr* dst,
  int interpolation=CV_INTER_LINEAR);
- 函数 cvResize 将图像 src 改变尺寸得到与 dst 同样大小的图像。
- src: 输入图像.　dst: 输出图像.
- Interpolation有线性、最近邻、区域和三次样条插值。

```
IplImage * test_img = cvLoadImage("D:/lena.jpg",1);

cvShowImage("dst", test_img);

IplImage * dst = NULL;
```

定义目标大小

```
double fScale = 5;      //缩放倍数
CvSize czSize;              //目标图像尺寸

//计算目标图像大小
czSize.width = test_img->width * fScale;
czSize.height = test_img->height * fScale;

dst = cvCreateImage(czSize, test_img->depth, test_img->nChannels);

cvResize(test_img,dst,CV_INTER_LINEAR);
cvShow[illegible]

cvResize(test_img,dst,CV_INTER_NN);
cvShowImage("dst_nn", dst);

cvWaitKey();
```

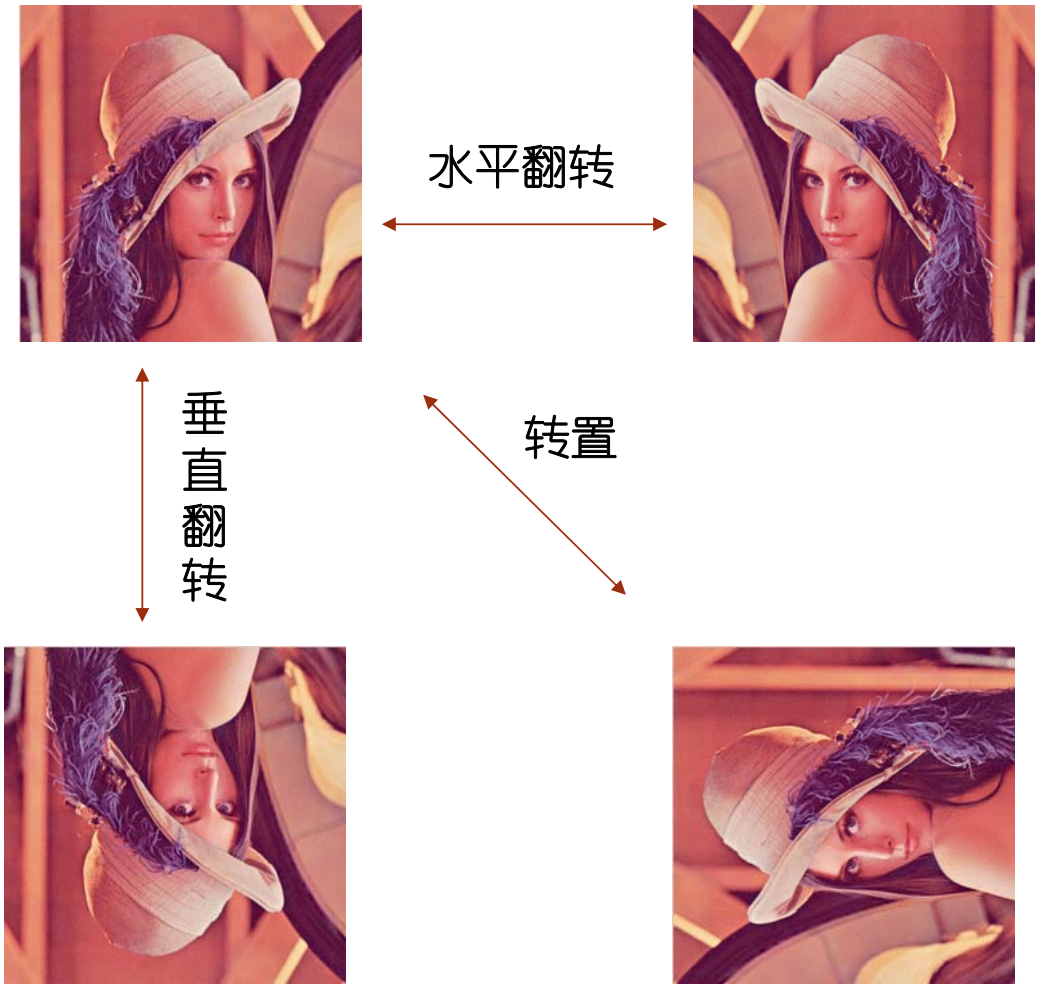
resize

- Interpolation: 插值方法:

  CV_INTER_LINEAR - 双线性插值

  CV_INTER_NN - 最近邻插值

(2) 图像旋转:



（2） 图像旋转:

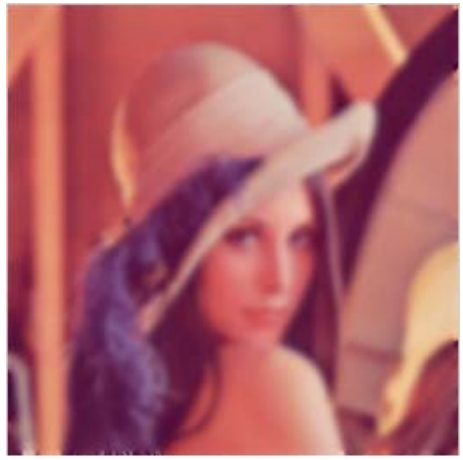
- void cvFlip(const CvArr* src,
  CvArr* dst=NULL,
  int flip_mode=0);

- Flip_mode: 翻转方法:
  flip_mode = 0 沿X-轴翻转,
  flip_mode > 0 (如 1) 沿Y-轴翻转
  flip_mode < 0 (如 -1) 沿X-轴和Y-轴翻转

- void cvTranspose(const CvArr* src, CvArr* dst);转置

(3) 像素点操作:

中心渐变



模糊

△访问图像像素:

- 坐标是从0开始的，图像原点通常是左上角，假设要访问第k通道、第i行、第j列的像素:
- ① 间接访问： 通用，效率低
- ② 直接访问：效率高，易出错

①访问像素 --- 间接访问

单通道	多通道

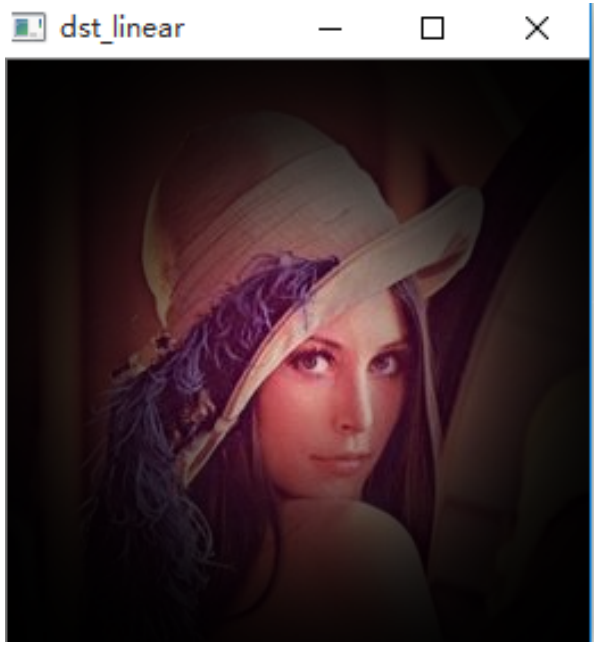
IplImage * img
=cvCreateImage（cvsize,
IPL_DEPTH_8U,1);
 	
CvScalar s;
 	
s=cvGet2D(img, i, j);
 	
s.val[0]=111;
 	
cvSet2D(img, i, j, s);
	IplImage * img
	=cvCreateImage（cvsize,
	IPL_DEPTH_32F,3);
	
	CvScalar s;
	
	s=cvGet2D(img, i, j);
	
	s.val[0]=111;
	s.val[1]=111;
	s.val[2]=111;
	
	cvSet2D(img,i,j,s);

②访问像素 --- 直接访问

单通道	多通道
IplImage * img =cvCreateImage（cvsize, IPL_DEPTH_8U,1); **(uchar*)** **(img->imageData** **+ img->widthStep * i))[j]=111;**	IplImage * img =cvCreateImage（cvsize, IPL_DEPTH_32F,3); **(uchar*)** **(img->imageData** **+ img->widthStep * i))[j*3+1]=111;** **(uchar*)** **(img->imageData** **+ img->widthStep * i))[j*3+2]=111;** **(uchar*)** **(img->imageData** **+ img->widthStep * i))[j*3+0]=111;**

例子：径向渐变

```
double scale = -3;
CvPoint center;
center = cvPoint(test_img->width/2,test_img->height/2);

for(int i=0;i<test_img->height;i++){
    for(int j=0;j<test_img->width;j++){
        double dx =(double)(j-center.x)/center.x;
        double dy =(double)(i-center.y)/center.y;

        double weight =exp((dx*dx+dy*dy)*scale);

        CvScalar s = cvGet2D(test_img,i,j);

        s.val[0] = cvRound(s.val[0]*weight);
        s.val[1] = cvRound(s.val[1]*weight);
        s.val[2] = cvRound(s.val[2]*weight);

        cvSet2D(dst,i,j,s);
    }
}

cvShowImage("dst", dst);
//cvResize(test_img,dst,CV_INTER_LINEAR);
```



常见应用

- Canny边缘检测
- 轮廓检测
- Harris 角点检测
- 直方图均衡
- 漫水填充法

Canny边缘检测

- void cvCanny(const CvArr* image, CvArr* edges, double threshold1,double threshold2, int aperture_size=3);
- 第二个参数表示输出的边缘图，需要和源图片有一样的尺寸和类型。
- 第五个参数表示Sobel 算子大小，默认为3即表示一个3*3的矩阵。
- Sobel 算子 :

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

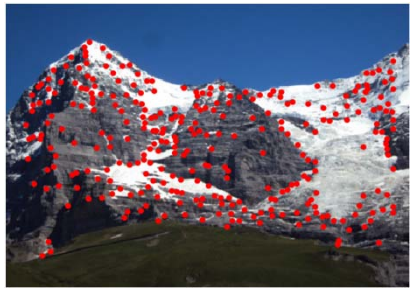
Canny边缘检测

- void cvCanny(const CvArr* image, CvArr* edges**, double threshold1,double threshold2**, int aperture_size=3);
- 滞后阈值(高阈值和低阈值):
  - 如果某一像素位置的幅值超过高阈值, 该像素被保留为边缘像素。
  - 如果某一像素位置的幅值小于低阈值, 该像素被排除。
  - 如果某一像素位置的幅值在两个阈值之间,该像素仅仅在连接到一个高于高阈值的像素时被保留。

Harris角点检测

void cornerHarris(InputArray src,OutputArray dst, int blockSize, int ksize, double k, intborderType=BORDER_DEFAULT)

- 角点的具体描述可以有几种：
  - 一阶导数(即灰度的梯度)的局部最大所对应的像素点；
  - 两条及两条以上边缘的交点；
  - 图像中梯度值和梯度方向的变化速率都很高的点；
  - 角点处的一阶导数最大，二阶导数为零，指示物体边缘变化不连续的方向。



- Harris角点检测是对于每一个像素(x,y)在某一邻域(blockSize× blockSize)内，计算梯度的协方差矩阵(M) ,然后计算 $dst(x,y) = det(M) - k\left(tr(M)\right)^2$，即可以找出输出图中的局部最大值，即找出了角点。

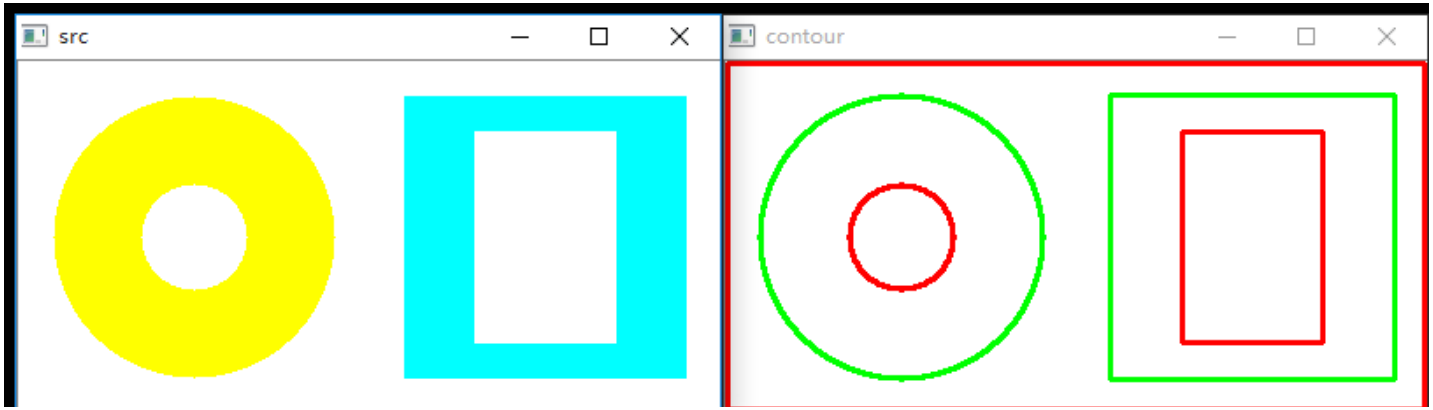
轮廓检测

- int cvFindContours(
  CvArr* img,
  CvMemStorage* storage,
  CvSeq** first_contour,
  int header_size=sizeof(CvContour),
  int mode=CV_RETR_LIST,
  int method=CV_CHAIN_APPROX_SIMPLE,
  CvPoint offset=cvPoint(0,0)
  );

对图像进行轮廓检测，生成一条链表以保存检测出的各个轮廓信息，并传出指向这条链表表头的指针。

- void cvDrawContours(
  CvArr *img,
  CvSeq* contour,
  CvScalar external_color,
  CvScalar hole_color,
  int max_level,
  int thickness=1,
  int line_type=8,
  CvPoint offset=cvPoint(0,0)
  );

在图像上绘制外部和内部轮廓。

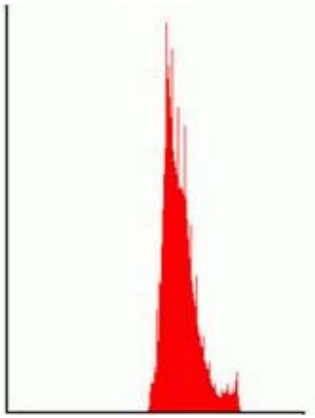


直方图均衡

- void cvCalcHist(IplImage** image, CvHistogram* hist, int accumulate CV_DEFAULT(0), const CvArr* mask CV_DEFAULT(NULL))

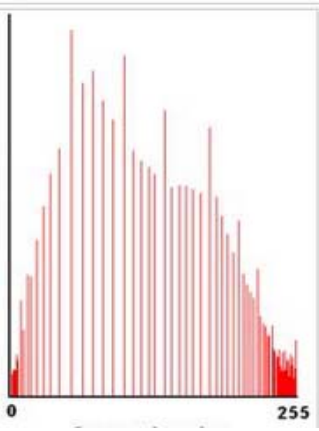
- void cvEqualizeHist(const CvArr* src, CvArr* dst);

第一个参数表示输入图像，必须为灰度图(8位，单通道图）。
第二个参数表示输出图像。



该函数采用如下法则对输入图像进行直方图均衡化：

1：计算输入图像的直方图H。
2：直方图归一化，因此直方块和为255。
3：计算直方图积分，H'(i) = Sum(H(j)) (0<=j<=i)。
4：采用H'作为查询表：dst(x, y) = H'(src(x, y))进行图像变换。



漫水填充法

自动选中和seedPoint相连的区域，将该区域替换成指定的颜色。当邻近像素点位于给定的范围（从loDiff到upDiff）内或在原始seedPoint像素值范围内时，FloodFill函数就会为这个点替换颜色。

int floodFill(InputOutputArray image, Point seedPoint, Scalar newVal, Rect* rect=0, Scalar loDiff=Scalar(), Scalar upDiff=Scalar(), int flags=4)

开发工具

- Visual Studio + OpenCV
- Matlab
- Python
- …

Matlab 概述

- MATLAB是matrix&laboratory两个词的组合，意为矩阵工厂（矩阵实验室）。
- 用于算法开发、数据可视化、数据分析以及**数值计算**，主要包括MATLAB和Simulink两大部分。

- 特点：
  - **高效的数值计算及符号计算功能**，能使用户从繁杂的数学运算分析中解脱出来；
  - 具有完备的**图形处理功能**，实现计算结果和编程的可视化；
  - 友好的用户界面及接近数学表达式的**自然化语言**，使学者易于学习和掌握；
  - 功能丰富的**应用工具箱**(如信号处理工具箱、通信工具箱等)，为用户提供了大量方便实用的处理工具。

界面组成

- **主页工具栏**

若干个功能模块，包括文件的新建、打开、查找等；数据的导入、保存工作区、新建变量等；代码分析、程序运行、命令清除等；窗口布局；预设MATLAB部分工作环境、设置当前工作路径；系统帮助；附加功能等。

- **命令行窗口**

主要功能为数值计算、函数参数设定、函数调用及其结果输出。一般来说，MATLAB的所有函数和命令都可以在命令行窗口中执行。

- **工作区窗口**

在工作区窗口中将显示所有目前保存在内存中的MATLAB变量的变量名及其对应的数据结构、字节数以及类型。

- **命令历史记录窗口**

该窗口记录着用户每一次开启MATLAB的时间，以及每一次开启MATLAB后，在MATLAB指令窗口中运行过的所有指令行。

- **当前文件夹窗口**

在当前文件夹窗口中可显示或改变当前文件夹，还可以显示当前文件夹下的文件，包括文件名、文件类型、最后修改时间以及该文件的说明信息等，并提供搜索功能。

- **当前已选择的文件详细信息**

有关命令行的操作

disp()
disp('输出字符串') //输出字符串注意使用单引号
disp(C) //输出变量

分号
当代码后面有分号时，按Enter键后，在命令窗口中不显示运行结果；如果无分号，则在命令窗口中显示运行结果。
如果需要同时执行输入的多条语句，则在输入下一条命令时，按Shift+Enter组合键进行。

Who，Whos，Which
Who，返回用过的变量名；whos，返回变量的信息；which 变量名，返回这个变量是否存在。

clc，close，close all，clear，clear all
clc，清除命令窗口的内容，对工作环境中的全部变量无任何影响；close，关闭当前的Figure窗口；close all，关闭所有的Figure窗口；clear，清除工作空间的所有变量；clear all，清除工作空间的所有变量，函数和MEX文件。

help
help，出现基本函数名及相关的介绍链接
help 函数名，出现该函数的说明。

基本函数

imread(filename,fmt) 从图像文件中读取（载入）图像
imwrite(A,filename,fmt) 把图像写入（保存）图像文件中
imshow(I,n) 显示图像
plot(X,Y) 绘制二维连续图像
plot3(X,Y,Z) 绘制三维曲线
stem(X,Y) 绘制离散图

imcrop(I) 剪切图像
imresize(A,m,method) 改变图像大小
imrotate(A,angle,method) 旋转图像

histeq(I,hgram) 直方图均衡
medfilt2(A,[m n]) 二维中值过滤
conv2(A,B) 二维卷积操作
edge(I,'sobel') 图像边缘检测
imcontour(I,n) 创建图像数据的轮廓图
……

谢谢！