# Contents

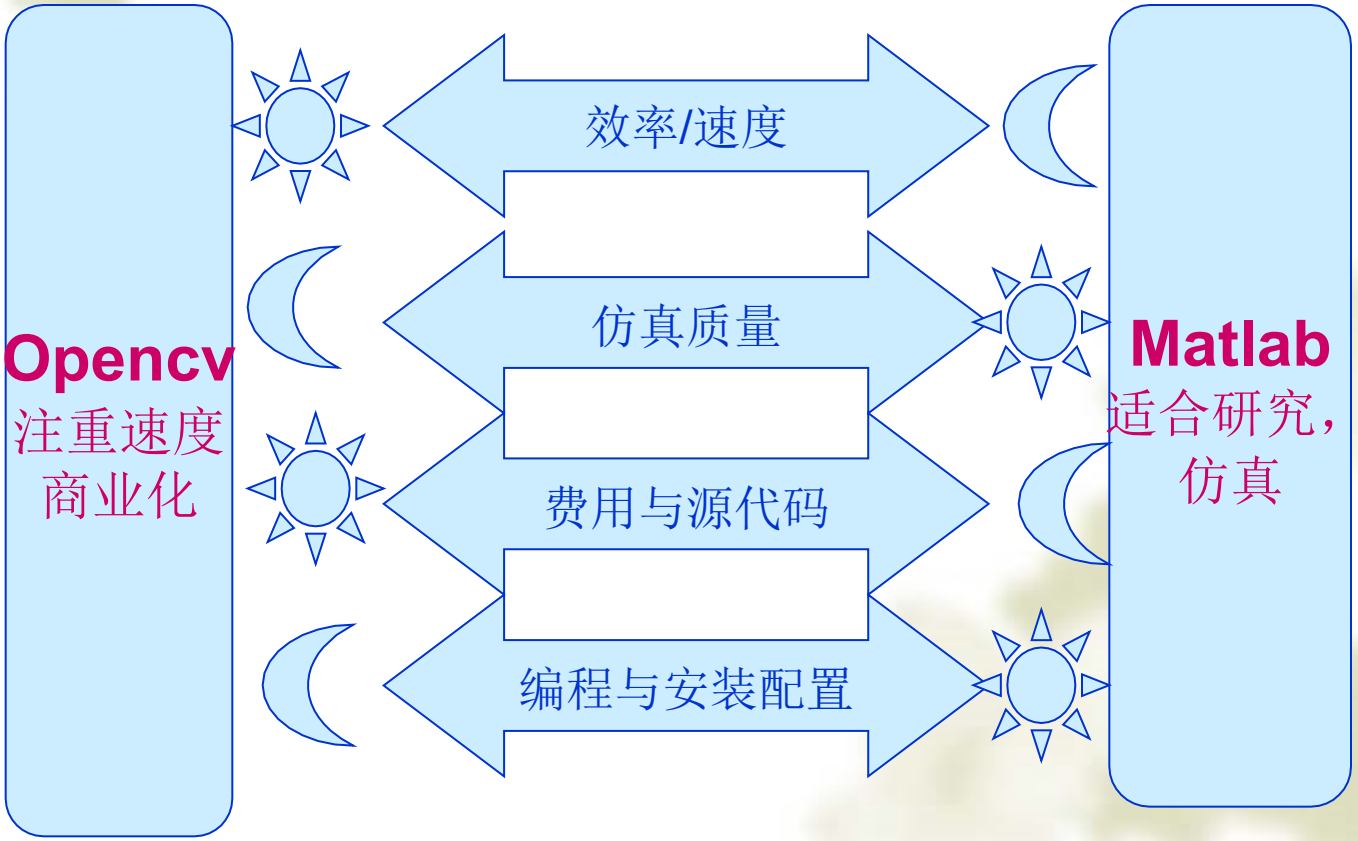
- 一、 OpenCV简介与特点
- 二、命名规则
- 三、 OpenCV模块与功能
- 四、例子：鼠标绘图

# Opencv简介

## Intel® Open Source Computer Vision Library

- 由一系列 C 函数和少许 C++ 类构成
- 实现了图像处理和计算机视觉方面的诸多通用算法。
- 涉及 300 多种C函数的跨平台的中、高层 API。
- 它不依赖于其他的外部库——尽管也能够使用某些外部库。

❖ Matlab，IPL， CxImage



# OpenCV特点

Intel® Open Source Computer Vision Library

OpenCV的几种明显优点：

- 跨平台,移植性好：Windows, Linux, Mac OS
- Open：免费，且源代码公开
- 实时性，商业化：代码优化，速度快
- 使用以便

缺陷：

Bug，安装配置麻烦

# OpenCV 功能

- 图像/视频的读写
- 数字图像的处理
- 目的辨认与跟踪
- 3D重建与标定



# 二. OpenCV的命名规则

OpenCV中使用大小写混合样式来标识外部函数、数据类型和类措施。

- 外部函数使用前缀cv：cvCreateImage
- 内部函数使用前缀Icv
- 数据构造(C构造体、枚举、联合体、类)使用前缀Cv：CvSize，CvPoint
- 外部或某些内部宏使用前缀CV_：CV_DEPTH_8U
- 内部宏使用前缀ICV_

# OpenCV 命名规则

（1）函数名:

- cvActionTargetMod(...)
  - Action = 关键功能 (e.g. set, create)
  - Target = 目的图像区域 (e.g. contour, polygon)
  - Mod = （可选的）调整语 (e.g. argument type)

  e.g.: cvGet2D、cvCreateImage、cvNamedWindow
- cv+算法/对象：cvSobel, cvCanny, cvRodrigues, cvSqrt, cvGoodFeaturesToTrack
- cv+容器元素+动作名

## （2）内部宏数据类型：

- 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 ： 一种8位无符号整型单通道矩阵
  - CV_32FC2： 一种32位浮点型双通道矩阵……
- 图像数据类型: IPL_DEPTH_<bit_depth>(S|U|F)
  - 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……
- 鼠标事件：

  event=CV_EVENT_<left/right>BUTTON<down/up>
  - CV_EVENT_LBUTTONUP,CV_EVENT_RBUTTONDOWN
  - CV_EVENT_LBUTTONDBLCLK

- (3).数据构造:

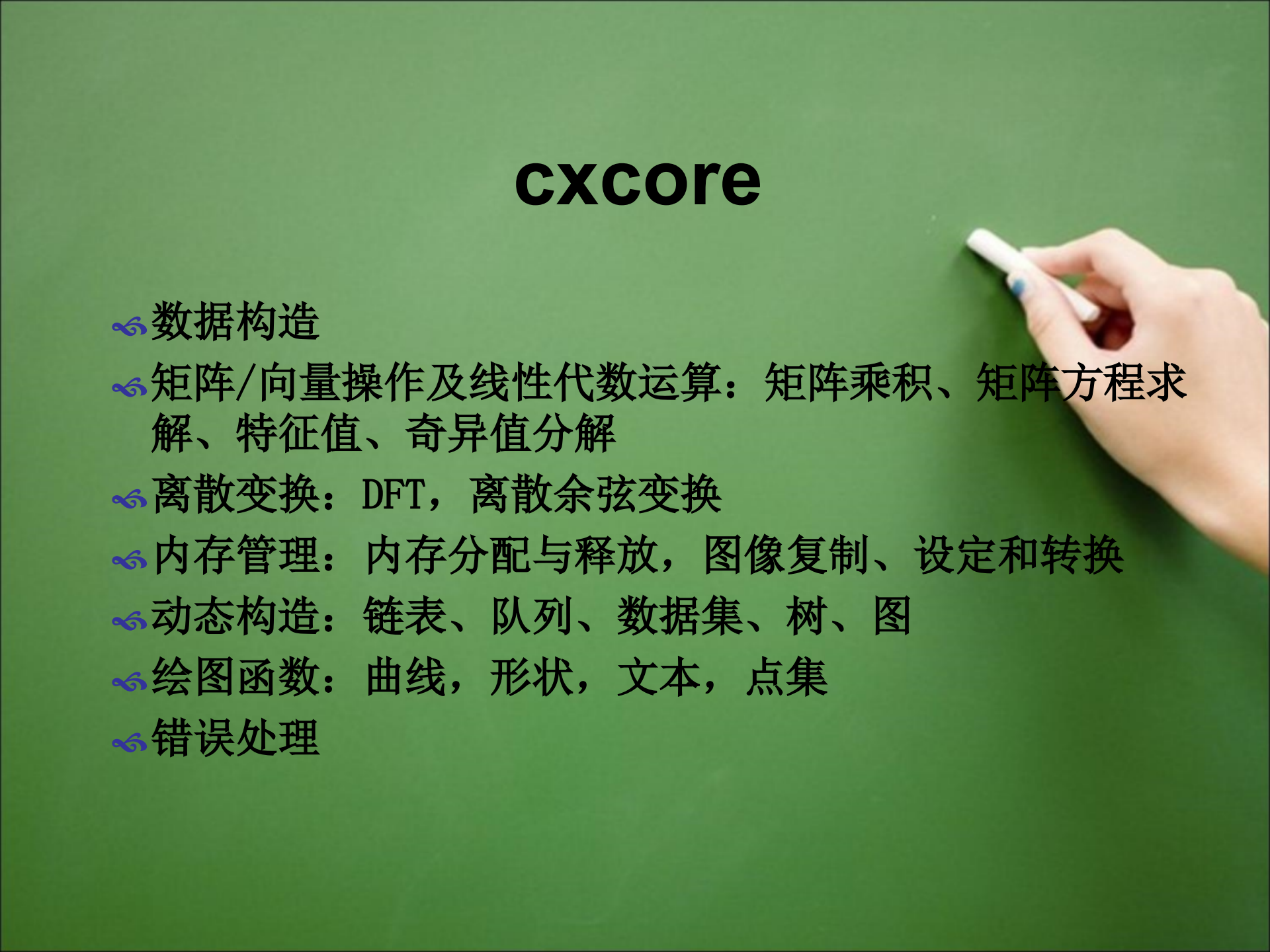
  CvMat（矩阵），CvPoint(点)，CvSize(矩形框)

- (4).头文件:
  - #include <cv.h>
  - #include <cvaux.h>
  - #include <highgui.h>
  - #include <ml.h>
  - #include <cxcore.h> // 一般不需要，cv.h 内已包含该头文件

# 三. OpenCV模块

- cv - 关键函数库
- cxcore - 数据构造与线性代数库
- cvaux - 辅助函数库
- highgui - GUI函数库
- ml - 机器学习函数库

# cxcore



- 数据构造
- 矩阵/向量操作及线性代数运算：矩阵乘积、矩阵方程求解、特征值、奇异值分解
- 离散变换：DFT，离散余弦变换
- 内存管理：内存分配与释放，图像复制、设定和转换
- 动态构造：链表、队列、数据集、树、图
- 绘图函数：曲线，形状，文本，点集
- 错误处理

# 基本数据构造

- 点：CvPoint, CvPoint2D32f, CvPoint3D32f
- 矩形框：CvSize，CvSize2D32f ，CvRect
- 矩阵：CvMat, CvMatND , CvSparseMat , CvArr*
- 向量：CvScalar
- IPL图像：IplImage（一种channel是一种CvMat ）

# 点

## CvPoint

```
typedef struct CvPoint
 {
  int x; /* x坐标, 通常以0为基点 */
  int y; /* y坐标, 通常以0为基点 */
 }
CvPoint;
```

## CvPoint2D32f

```
typedef struct CvPoint2D32f
 {
  float x; /* x坐标, 通常以0为基点*/
  float y; /* y坐标, 通常以0为基点*/
 }
CvPoint2D32f;
```

## CvPoint3D32f

```
typedef struct CvPoint3D32f
 {
  float x; /* x-坐标, 通常基于0 */
  float y; /* y-坐标, 通常基于0 */
  float z; /* z-坐标, 通常基于0 */
 }
 CvPoint3D32f;
```

# 矩形与向量

## CvSize

```
typedef struct CvSize
 {
  int width;   /* 矩形宽 */
  int height; /* 矩形高 */
 }
 CvSize;
```

## CvSize2D32f

```
typedef struct CvSize2D32f
 {
   float width;   /* 矩形宽 */
   float height; /* 矩形高 */
 }
 CvSize2D32f;
```

## CvRect

```
typedef struct CvRect
 {
  int x; /* 方形的最左角的x-坐标   */
  int y; /* 方形的最上或者最下角的y-坐标 */
  int width; /* 宽 */
  int height; /* 高 */
 }
 CvRect;
```

## CvScalar

可存储在1至4个捆绑数据的容器

```
typedef struct CvScalar
 {
  double val[4]
 }
 CvScalar;
```