OpenCV Tutorial

潘　纲

浙江大学计算机学院

OpenCV起源

- **Intel Performance Lib**
  - IPL，MKL，RPL，SPL
  - IPP: Integrated Performance Primitives
    - Video/Audio, Speech, Cryptography, IP, CV,

- **Open Source Computer Vision Library**
  - **http://www.opencv.org.cn**
  - **http://www.opencv.org**
  - http://sourceforge.net/projects/opencvlibrary/

OpenCV的特点

- Open sources: BSD license
- 采用C/C++/STL编写 (+ Python、Java、MATLAB)
- 跨平台：Win/Linux/MacOS
- 支持移动平台：iOS/Android
- 独立性：不依赖于其他的外部库
- 通用的图像、视频输入、输出与保存
- 2500+ 个算法
- 适用于开发实时应用程序
- 47K users

OpenCV的版本

Version core b1.5，	2001年4月
Version Beta 3/3.1，	2002-2003
Version Beta 4 (0.9.6)	2004.8
Version Beta 5 (0.9.7)	2005.7
Version 1.0：	2006年11月
Version 1.1pre1a:	2008.10
Version 2.0:	2009年10月
Version 2.1:	2010年4月
Version 2.2 -2.4:	
Version 2.4.3:	2012年11月2日
Version 2.4.7:	2013年11月7日
Version 2.4.9	2014.04.16
Version 2.4.10:	2014.10.02
Version 3.0 beta:	2014.11.11
Version 3.0:	2015.06.04
Version 3.1:	2015.12.21
Version 2.4.13:	2016.05.19
Version 2.4.13.4:	2017.10.12
Version 3.3.1:	2017.10.23
Version 4.0.0:	2018.11.18
Version 2.4.13.6:	2019.02.26
Version 3.4.7:	2019.07.26
Version 4.1.1:	2019.07.26

OpenCV的版本

- **Version 1.x：   2006年11月**
  - C为主。SURF、RANSAC、Face detection、random trees/boosted trees…
- **Version 2.x: 2009年10月**
  - C++为主。C API很少更新与新加。CMake构建。
  - FAST、LBP、Grabcut。GPU加速。 opencv_contrib尚未成熟
- **Version 3.x: 2015年6月**
  - 3.x与2.x不完全兼容，大部分用了OpenCL加速
  - 有专利保护的技术也归到opencv_contrib
  - opencv_dnn独立出来
- **Version 4.x: 2018年11月**
  - 全面采用C++11
  - 加入QR code识别、Kinect fusion算法等

OpenCV的基本功能

- 图像操作
- 视频操作
- 矩阵、向量、部分常用线性代数算法
- 动态数据结构：**list/set/sequence/tree/graph**
- 基本的图像处理功能
- 结构分析
- 摄像头定标
- 运动分析
- 目标识别
- 基本的**GUI**功能

（仅供本课程内部学习，勿上载外网）

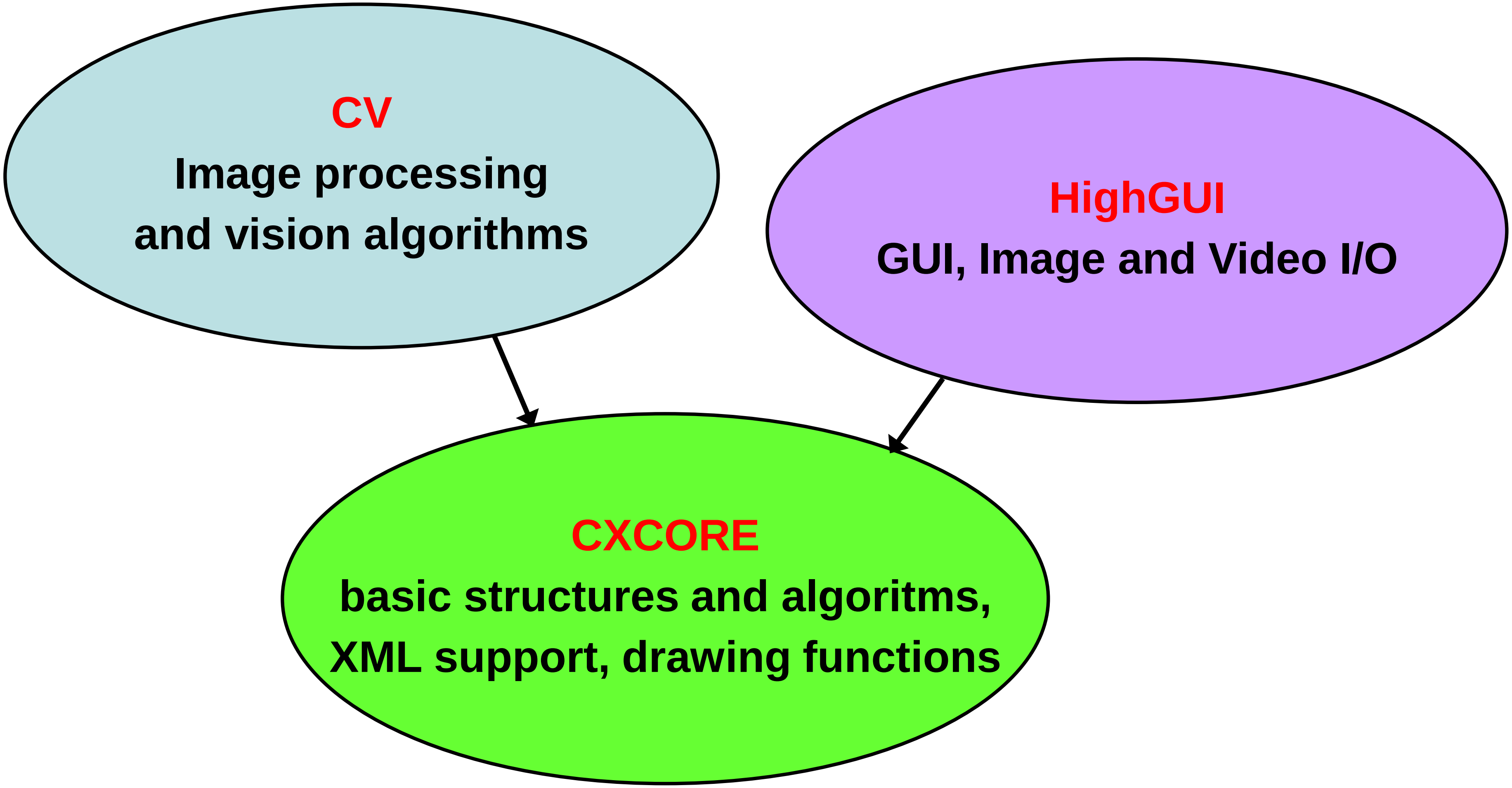
OpenCV v.s. Matlab

- **OpenCV与Matlab中的图像处理工具箱相比：**
  - OpenCV用编程语言调用，效率高；Matlab使用脚本语言，直观方便。
  - OpenCV适合开发实时系统；Matlab适合算法仿真与算法测试。
  - OpenCV开发源码；Matlab商业产品，核心算法代码无法获得。

OpenCV 2.1 模块划分

- **cv**
  - 主要的ip/cv算法函数
- **cxcore**
  - 数据结构、线性代数
- **highgui**
  - 外部界面库
- **cvaux**
  - 辅助的（实验性的）函数
- **ml**
  - Machine learning, 机器学习

三个基本模块的关系



CV

Image processing and vision algorithms

HighGUI

GUI, Image and Video I/O

CXCORE

basic structures and algoritms, XML support, drawing functions

VC.net 下的设置

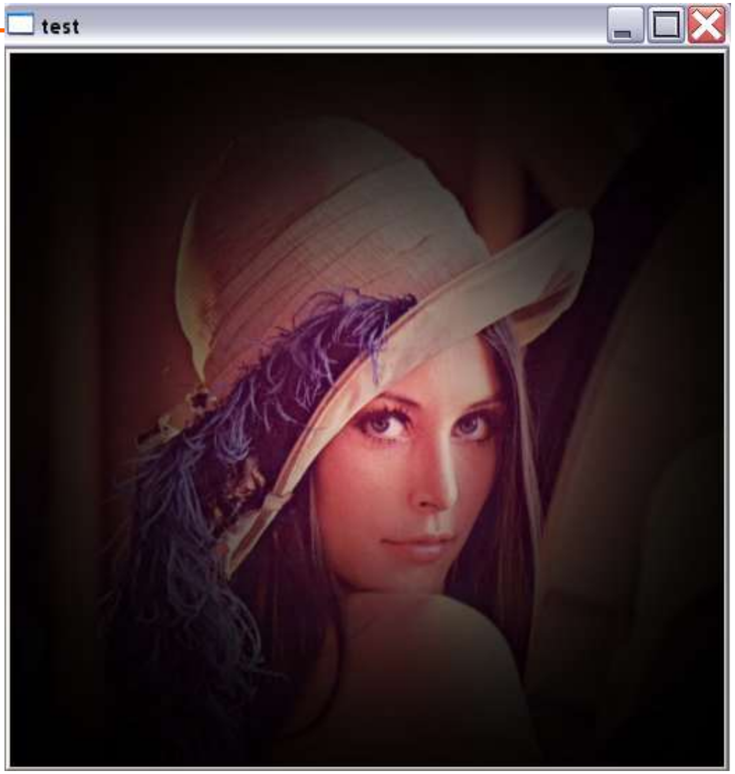
- **路径设置：**
  - Tools|Options → Projects|VC++Directories
    - Library files
    - Include files

- **新建Project后**
  - Additional dependencies加：

    cxcore.lib highgui.lib cv.lib cvaux.lib …

Sample

```
#include <cxcore.h>
#include <highgui.h>
#include <math.h>
int main( int argc, char** argv ) {
    CvPoint center;
    double scale=-3;
    IplImage* image = cvLoadImage(argv[1]) ;
    if(!image) return -1;
    center = cvPoint(image->width/2,image->height/2);
    for(int i=0;i<image->height;i++)
        for(int j=0;j<image->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);
            uchar* ptr = &CV_IMAGE_ELEM(image,uchar,i,j*3);
            ptr[0] = cvRound(ptr[0]*weight);
            ptr[1] = cvRound(ptr[1]*weight);
            ptr[2] = cvRound(ptr[2]*weight);  }
    cvSaveImage( “copy.png”, image );
    cvNamedWindow( "test", 1 );
    cvShowImage( "test", image );
    cvWaitKey();
    return 0; }
```



```
#include <cxcore.h>
#include <highgui.h>
#include <math.h>
int main( int argc, char** argv ) {
    CvPoint center;
    double scale=-3;
    IplImage* image = cvLoadImage(argv[1]) ;
    if(!image) return -1;
    center = cvPoint(image->width/2,image->height/2);
    for(int i=0;i<image->height;i++)
        for(int j=0;j<image->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);
            uchar* ptr = &CV_IMAGE_ELEM(image,uchar,i,j*3);
            ptr[0] = cvRound(ptr[0]*weight);
            ptr[1] = cvRound(ptr[1]*weight);
            ptr[2] = cvRound(ptr[2]*weight);  }
    cvSaveImage( “copy.png”, image );
    cvNamedWindow( "test", 1 );
    cvShowImage( "test", image );
    cvWaitKey();
    return 0; }
```

- 基本GUI
- 图像操作
- 视频操作
- 高级GUI
- 简单数值计算

窗口

- 创建及定位
  - cvNamedWindow(“w1”, CV_WINDOW_AUTOSIZE)
  - cvMoveWindow(“w1”,50,100)
- 显示图像
  - IplImage* img=cvLoadImage(“lena.jpg”)
  - cvShowImage(“w1”, img)
- 关闭窗口
  - cvDestroyWindow(“w1”)
- 缩放窗口
  - cvResizeWindow(“w1”, 100,200)

（仅供本课程内部学习，勿上载外网）

键盘

- **cvWaitKey**
  - int key
  - key=cvWaitKey(0)
  - key=cvWaitKey(2000)

```
…
for(…) {
    …
    int c = cvWaitKey(100);
    if( c >= 0 )
        // key_pressed
        break;
}
```

Mouse

```
CV_EXTERN_C_FUNCPTR (   void (*CvMouseCallback )
                         (int event, int x, int y, int flags, void* param) );

void cvSetMouseCallback ( const char* window_name,
                         CvMouseCallback on_mouse,
                         void* param=NULL );
```

Mouse

Event

```
#define CV_EVENT_MOUSEMOVE      0
#define CV_EVENT_LBUTTONDOWN    1
#define CV_EVENT_RBUTTONDOWN    2
#define CV_EVENT_MBUTTONDOWN    3
#define CV_EVENT_LBUTTONUP      4
#define CV_EVENT_RBUTTONUP      5
#define CV_EVENT_MBUTTONUP      6
#define CV_EVENT_LBUTTONDBLCLK  7
#define CV_EVENT_RBUTTONDBLCLK  8
#define CV_EVENT_MBUTTONDBLCLK  9
```

Flag

```
#define CV_EVENT_FLAG_LBUTTON   1
#define CV_EVENT_FLAG_RBUTTON   2
#define CV_EVENT_FLAG_MBUTTON   4
#define CV_EVENT_FLAG_CTRLKEY   8
#define CV_EVENT_FLAG_SHIFTKEY  16
#define CV_EVENT_FLAG_ALTKEY    32
```

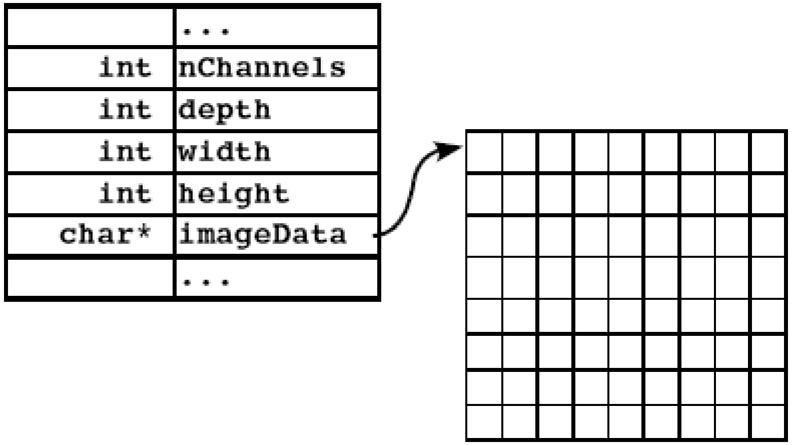
Mouse: 自拟函数mouseHandler

```
void mouseHandler (int event, int x, int y, int flags, void* param)
 {
   switch(event) {
       case CV_EVENT_LBUTTONDOWN:
           if(flags & CV_EVENT_FLAG_CTRLKEY)
           printf("Left button down with CTRL pressed\n");
           break;

       case CV_EVENT_LBUTTONUP:
           printf("Left button up\n");
           break;
   }
 }

void main ()
{  …
   mouseParam=5;
   cvSetMouseCallback("win1",mouseHandler,&mouseParam);
…}
```

- 基本GUI
- 图像操作
- 视频操作
- 高级GUI
- 简单数值计算

图像数据结构 IplImage

图像：二维（单通道）或者三维（多通道）的矩阵。
在OpenCV中，图像数据结构类行为IplImage，其定义如下图：



	...
int	nChannels
int	depth
int	width
int	height
char*	imageData
	...

支持的像素深度：IPL_DEPTH_8U, ...8S, ...16U, ...16S, ...32S, ...32F和...64F

图像的读写

iscolor > 0
iscolor = 0
Iscolor < 0

cvLoadImage

IplImage* cvLoadImage(const char* filename, int iscolor=1)
支持的图像格式有：BMP, JPEG, PNG, PBM, PGM, PPM, SR, TIFF, JPEG2000.

cvSaveImage

int cvSaveImage(const char* filename, const CvArr* image)

- CvArr可以是IplImage, cvMat, cvSeq。参数image类型在这个函数里只可能是IplImage或者CvMat。
- 保存的图像文件格式由filename的扩展名确定。
- 只有8bit的单通道或者3通道（只可BGR顺序）的图像可以被保存，不支持alpha通道。

图像创建

cvCreateImage

IplImage* cvCreateImage(CvSize size, int depth, int channels)

这个函数等价于:

header = cvCreateImageHeader(size,depth,channels);

cvCreateData(header);

IPL_DEPTH_8U
IPL_DEPTH_8S
IPL_DEPTH_16U
IPL_DEPTH_16S
IPL_DEPTH_32S
IPL_DEPTH_32F
IPL_DEPTH_64F

其他相关函数

cvReleaseImage
cvCreateMat
cvReleaseMat
cvCloneImage
cvClonemat

访问图像的某个像素

- **CV_IMAGE_ELEM(img, T, y, x*N + c)**
  - ((T*)(img->imageData + img->widthStep*y))[x*N + c]
  - blue: c=0 green: c=1 red:c=2

```
uchar* ptr = &CV_IMAGE_ELEM(image,uchar,i,j*3);
```

简单数据结构

- CvPoint --2D integer
- CvPoint2D32f --2D float
- CvPoint2D64f --2D double
- CvPoint3D32f --3D float
- CvPoint3D64f --3D double

- CvSize --2D int, pixel
- CvSize2D32f --2D float, subpixel

- CvRect --int

- CvScalar --4-tuples of numbers (double)

色彩空间的转换

- 彩色→ 灰度
  - cvCvtColor (cimg, gimg, CV_BGR2GRAY)

- **Color space conversion**
  - **cvCvtColor** (c1,c2, code)
  - code = CV_{**X**}2{**Y**}

    X,Y: RGB, BGR, GRAY,HSV,YCrCb,XYZ,Lab,Luv,HLS

    如：CV_BGR2Lab, CV_BGR2HSV

图像增强

图像平滑（低通滤波）

```
void cvSmooth( const CvArr* src, CvArr* dst, int
smoothtype=CV_GAUSSIAN, int param1=3, int param2=0,
double param3=0, double param4=0 );
```

可用的平滑图像的方法有：

- CV_BLUR_NO_SCALE：临域求和
- CV_BLUR：平均法
- CV_GAUSSIAN：高斯滤波
- CV_MEDIAN：中值法
- CV_BILATERAL：双向滤波（Bilateral filter）

图像增强

直方图均衡化

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

例子equalizehist：直方图均衡化

例子demhist：亮度和对比度分别跟直方图的关系

直方图相关的操作函数：cvNormalizeHist, cvThreshHist, cvCalcHist, cvCalcBackProject, cvCalcBackProjectPatch· · ·

图像缩放

- **cvResize(const CvArr* src, CvArr* dst, int interpolation=CV_INTER_LINEAR);**

CV_INTER_NN
CV_INTER_LINEAR
CV_INTER_AREA
CV_INTER_CUBIC

图像变换

- 傅立叶变换 ▶
  - cvDFT
- 离散余弦变换
  - cvDCT
- 主元分析
  - cvEigenVV
  - cv::PCA

Morphology

```
IplConvKernel* cvCreateStructuringElementEx( int cols, int rows,
                        int anchor_x, int anchor_y, int shape, int* values=NULL );
  CV_SHAPE_RECT, a rectangular element;
  CV_SHAPE_CROSS, a cross-shaped element;
  CV_SHAPE_ELLIPSE, an elliptic element;
  CV_SHAPE_CUSTOM, a user-defined element. （这时，values指定一个mask）
void cvReleaseStructuringElement( IplConvKernel** element );

void cvErode( const CvArr* src, CvArr* dst,
                         IplConvKernel* element=NULL, int iterations=1 );
void cvDilate( const CvArr* src, CvArr* dst,
                         IplConvKernel* element=NULL, int iterations=1 );
void cvMorphologyEx( const CvArr* src, CvArr* dst, CvArr* temp,
                         IplConvKernel* element, int operation, int iterations=1 );
  CV_MOP_OPEN – opening    (erode → dilate)
  CV_MOP_CLOSE – closing    (dilate → erode )
  CV_MOP_GRADIENT - morphological gradient    (dilate - erode)
  CV_MOP_TOPHAT - "top hat“  (src - open)
  CV_MOP_BLACKHAT - "black hat“ (close - src)
```

- 基本GUI
- 图像操作
- 视频操作
- 高级GUI
- 简单数值计算

视频I/O—从文件或摄像头读取

- CvCapture* cvCaptureFromCAM(camera_id=0);
  initializes capturing from the specified camera
- CvCapture* cvCaptureFromFile(videofile_path);
  initializes capturing from the video file.

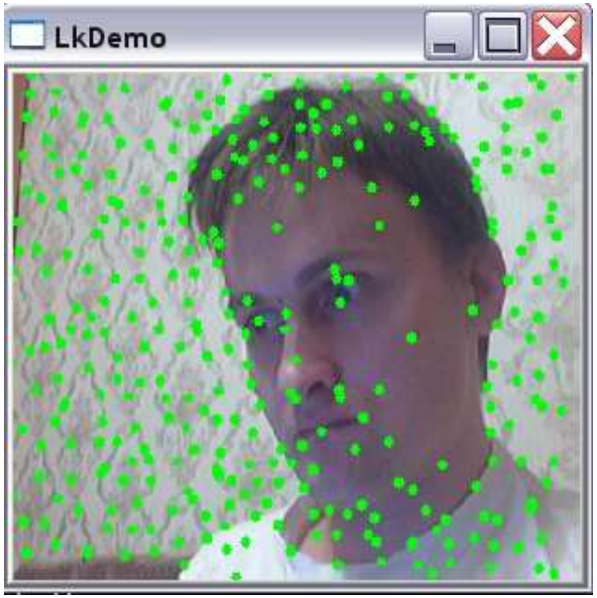
- IplImage* cvQueryFrame(capture);
  retrieves the next video frame (do not alter the result!), or NULL if there is no more frames or an error occured.

- cvGetCaptureProperty(capture, property_id);
  cvSetCaptureProperty(capture, property_id, value);
  retrieves/sets some capturing properties (camera resolution, position within video file etc.)

- cvReleaseCapture(&capture);
  do not forget to release the resouces at the end!

Video I/O: Sample Code

```
// opencv/samples/c/lkdemo.c
int main(…){
…
CvCapture* capture = <…> ?
     cvCaptureFromCAM(camera_id) :
     cvCaptureFromAVI(path);
if( !capture ) return -1;
for(;;) {
   IplImage* frame=cvQueryFrame(capture);
   if(!frame) break;
   // … copy and process image
   cvShowImage( “LkDemo”, result );
   c=cvWaitKey(30); // run at ~20-30fps speed
   if(c >= 0) {
       // process key
 }}
cvReleaseCapture(&capture);}
```



lkdemo.c, 190 lines

(needs camera to run)

Video I/O: 生成

```
CV_FOURCC('P','I','M','1') = MPEG-1 codec
CV_FOURCC('M','J','P','G') = motion-jpeg codec
CV_FOURCC('M', 'P', '4', '2') = MPEG-4.2 codec
CV_FOURCC('D', 'I', 'V', '3') = MPEG-4.3 codec
CV_FOURCC('D', 'I', 'V', 'X') = MPEG-4 codec
CV_FOURCC('U', '2', '6', '3') = H263 codec
CV_FOURCC('I', '2', '6', '3') = H263I codec
CV_FOURCC('F', 'L', 'V', '1') = FLV1 codec
```

1. 初始化一个VideoWriter

```
   CvVideoWriter *writer =
    int isColor = 1;
    int fps = 25; // or 30
    int frameW = 640;
    int frameH = 480;
    writer=cvCreateVideoWriter("out.avi",CV_FOURCC('P','I','M','1'),
                                 fps,cvSize(frameW,frameH),isColor);
```

2. 写入视频帧

```
    IplImage* img = 0;
    int nFrames = 50;
    for(i=0;i<nFrames;i++) {
      img=cvQueryFrame(capture); // retrieve the captured frame
      cvWriteFrame(writer,img); // add the frame to the file
    }
```

3. 释放/保存

```
   cvReleaseVideoWriter(&writer);
```

- 基本GUI
- 图像操作
- 视频操作
- 高级GUI
- 简单数值计算

Drawing

- 画线段
  - cvLine(img, cvPoint(100,100), cvPoint(200,200), cvScalar(0,255,0), 1);
- 画矩形
  - cvRectangle(img, cvPoint(100,100), cvPoint(200,200), cvScalar(255,0,0), 1);
- 画圆
  - cvCircle(img, cvPoint(100,100), 20, cvScalar(0,255,0), 1);

Drawing

- 画多边形

```
CvPoint curve1[]={10,10, 10,100, 100,100, 100,10};
CvPoint curve2[]={30,30, 30,130, 130,130, 130,30, 150,10};
CvPoint* curveArr[2]={curve1, curve2};
int nCurvePts[2]={4,5};
int nCurves=2;
int isCurveClosed=1;
int lineWidth=1;
cvPolyLine(img,curveArr,nCurvePts,nCurves,isCurveClosed,
                                  cvScalar(0,255,255),lineWidth);
```

- 画填充的多边形

```
cvFillPoly(img,curveArr,nCurvePts,nCurves,cvScalar(0,255,255));
```

Drawing

- 写文字（汉字支持问题）

```
CvFont font;
double hScale=1.0;
double vScale=1.0;
int lineWidth=1;

cvInitFont(&font,CV_FONT_HERSHEY_SIMPLEX|CV_FONT_ITALIC,
hScale,vScale,0,lineWidth);

cvPutText (img,"My comment",cvPoint(200,400), &font,
                                        cvScalar(255,255,0));
```

▶

- 基本GUI
- 图像操作
- 视频操作
- 高级GUI
- 简单数值计算

简单数学函数

- cvRound
- cvFloor
- cvCeil

- cvSqrt
- cvInvSqrt --1./sqrt(x)
- cvCbrt --cubic root

- cvLog： 绝对值的自然对数
- cvExp： 自然指数
- cvPow： 求幂

简单数学函数

- 求解一元三次函数的实根
  - cvSolveCubic(CvArr* coeffs, CvArr* roots)
  - coeffs: 3-4个元素的数值
  - roots: 输出，三个元素

$$x^3 + a_0 x^2 + a_1 x + a_2 = 0$$

$$a_0 x^3 + a_1 x^2 + a_2 x + a_3 = 0$$

矩阵

- 创建
  - CvMat* **cvCreateMat**(int rows, int cols, int type);
  - type: CV_<depth>(S|U|F)C<number_of_channels>.

    CV_32UC1
- 释放
  - CvMat* M = cvCreateMat(4,4,CV_32FC1);
    cvReleaseMat(&M);
- **访问元素---single channel**
  - CvMat* mat
  - M(i,j):   CV_MAT_ELEM(mat, float, i, j)

矩阵和向量操作

- 矩阵间
  - CvMat *Ma, *Mb, *Mc;
  - cvAdd(Ma, Mb, Mc); // Ma+Mb -> Mc
  - cvSub(Ma, Mb, Mc); // Ma-Mb -> Mc
  - cvMatMul(Ma, Mb, Mc); // Ma*Mb -> Mc
- 矩阵元素间
  - CvMat *Ma, *Mb, *Mc;
  - cvMul(Ma, Mb, Mc); // Ma.*Mb -> Mc
  - cvDiv(Ma, Mb, Mc); // Ma./Mb -> Mc
  - cvAddS(Ma, cvScalar(-10.0), Mc); // Ma.-10 -> Mc

矩阵与向量操作

- 向量乘
  - double va[] = {1, 2, 3};
  - double vb[] = {0, 0, 1};
  - double vc[3];
  - CvMat Va=cvMat(3, 1, CV_64FC1, va);
  - CvMat Vb=cvMat(3, 1, CV_64FC1, vb);
  - CvMat Vc=cvMat(3, 1, CV_64FC1, vc);
  - double res=cvDotProduct(&Va,&Vb); // dot product: Va . Vb -> res
  - cvCrossProduct(&Va, &Vb, &Vc); // cross product: Va x Vb -> Vc
- 单矩阵
  - cvTranspose(Ma, Mb); // transpose(Ma) -> Mb (cannot transpose onto self)
  - CvScalar t = cvTrace(Ma); // trace(Ma) -> t.val[0]
  - double d = cvDet(Ma); // det(Ma) -> d
  - cvInvert(Ma, Mb); // inv(Ma) -> Mb

对称矩阵特征值求解

```
CvMat* A = cvCreateMat(3,3,CV_32FC1);
CvMat* E = cvCreateMat(3,3,CV_32FC1);
CvMat* l = cvCreateMat(3,1,CV_32FC1);

cvEigenVV(&A, &E, &l);
// l = eigenvalues of A (descending order)
// E = corresponding eigenvectors (rows)
```

Inpaint

```
void cvInpaint ( const CvArr* src, const CvArr* mask, CvArr* dst,
                                                        int flags, double inpaintRadius );
```

mask:

8-bit 1-channel image, 大小与src一样，非零像素表示需要inpainted

flags:

CV_INPAINT_NS - Navier-Stokes based method.
CV_INPAINT_TELEA - The method by Alexandru Telea [Telea04]

inpaintRadius:

算法考虑的领域半径 (=3)

Demo

- adaptiveskindetector.exe
- camshiftdemo.exe
- contours.exe
- convexhull.exe
- demhist.exe
- dft.exe
- drawing.exe
- fback.exe: 光流
- facedetect.cmd
- find_obj.exe: 对应点
- fitellipse.exe
- houghlines.exe
- inpaint.exe
- lkdemo.exe
- …

References

- OpenCV自带文档与手册
- http://opencvlibrary.sourceforge.net/
- http://www.opencv.org.cn
- Gary Bradski, Adrian Kaebler: 《Learning OpenCV：Computer Vision with the OpenCV Library》, O'Reilly, 2008
- 刘瑞祯、于仕琪，《OpenCV教程—基础篇》，北京航空航天大学出版社