OpenCV简介

- OPENCV是什么?
  - OPENCV= Intel (c) Open source computer vision library
  - OpenCV 是英特尔® 开源计算机视觉库，是Intel 公司面向应用程序开发者开发的计算机视觉库。
  - OpenCV中包含了300多种图像处理和计算机视觉方面的c/c++程序，其中包含大量的函数用来处理计算机视觉领域中常见的问题,例如运动分析和跟踪、人脸识别、3D 重建和目标识别等。

OpenCV简介 (续)

- OPENCV的优点?
  - 跨平台，可移植性好，无论windows/linux
  - 免费，无论商业用途与否
  - 代码效率高
  - 使用方便
  - 具备强大的图像和矩阵运算能力，可以大
  - 大减少开发者的编程工作量，有效提高开
  - 发效率和程序运行的可靠性
- OPENCV的缺点?
  - 过于特殊的数据结构
  - 完全不同于windows的窗口方式

OpenCV简介 (续)

- OPENCV的组成?

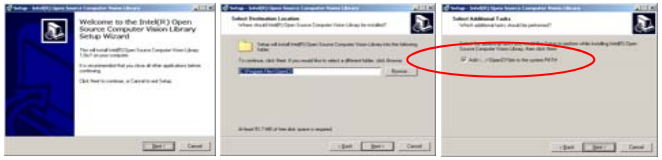
OpenCV API Reference
Introduction
core. The Core Functionality
imgproc. Image Processing
highgui. High-level GUI and Media I/O
video. Video Analysis
calib3d. Camera Calibration and 3D Reconstruction
features2d.
objdetect. Object Detection
ml. Machine Learning
flann. Clustering and Search in Multi-Dimensional Spaces
gpu. GPU-accelerated Computer Vision
stitching. Images stitching

VC6下安装与配置

- 安装Visual C++ 6.0
- 安装OpenCV
- 配置Windows环境变量
- 配置Visual C++ 6.0
  - 全局设置
  - 项目设置

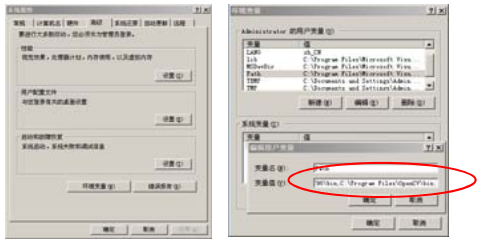
OpenCV的安装

- 安装OpenCV
  - 从http://www.opencv.org.cn 下载OpenCV安装程序。
  - 假如要将OpenCV安装到C:\Program Files\OpenCV。（下面附图为OpenCV 的安装界面，OpenCV 1.0安装界面与此基本一致。）
  - 在安装时选择"将\OpenCV\bin加入系统变量"



OpenCV的安装（续）

- 配置Windows环境变量
  - 检查C:\Program Files\OpenCV\bin是否已经被加入到环境变量PATH，如果没有，请加入。
  - 加入后需要注销当前Windows用户（或重启）后重新登陆才生效。



OpenCV的安装（续）

- 配置Visual C++ 6.0
  - 菜单Tools->Options->Directories：先设置lib路径，选择Library files，在下方填入路径：

    C:\Program Files\OpenCV\lib
  - 然后选择include files，在下方填入路径：

    C:\Program Files\OpenCV\cxcore\include

    C:\Program Files\OpenCV\cv\include

    C:\Program Files\OpenCV\cvaux\include

    C:\Program Files\OpenCV\ml\include

    C:\Program Files\OpenCV\otherlibs\highgui

    C:\Program Files\OpenCV\otherlibs\cvcam\include

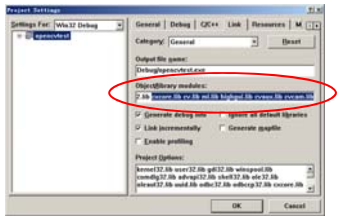
OpenCV的安装 （续）

❑项目设置

- 每创建一个将要使用OpenCV的VC Project，都需要给它指定需要的lib。

菜单：Project->Settings，然后将Setting for选为All Configurations，然后选择右边的link标签，在Object/library modules附加上

cxcore.lib cv.lib ml.lib cvaux.lib highgui.lib cvcam.lib



OpenCV的安装 （续）

- ❑ 安装Visual C 2010
- ❑ 安装OpenCV 2.4.2
- ❑ 配置Windows环境变量
- ❑ 配置Visual C2010
  - ❑ 全局设置
  - ❑ 项目设置

OpenCV的安装 （续）

1. http://www.opencv.org.cn/index.php/Download 下载 OpenCV for Windows（也即OpenCV-2.4.3.exe文件）。
2. 将OpenCV-2.4.3.exe解压并放到某个目录下，例如D:\OpenCV。
3. 打开VC++ 2010 Express，创建一个Win32控制台程序opencvtest；
4. 选择Property Manager里的Microsoft.cpp.win32.user，点击鼠标右键，选择Properties。
5. 依次选择“配置属性”->“VC++目录”->“包含目录”，包含

   D:\OpenCV\build\include；

   D:\OpenCV\build\include\opencv；

   D:\OpenCV\build\include\opencv2

OpenCV的安装 （续）

1. 配置lib路径.将库目录D:\OpenCV\build\x86\vc10\lib输入“属性”->“配置属性”->“VC++目录”->“库目录”。
2. 将OpenCV的dll文件所在的目录加入Path环境变量:32位系统 & VC2010，dll目录为：D:\OpenCV\build\x86\vc10\bin
3. 由于有些函数需要TBB，所以需要将tbb所在的目录也加入到环境变量Path中:D:\OpenCV\build\common\tbb\ia32\vc10
4. 选择Solution Explorer里的opencvtest项目，点击鼠标右键，选择Properties。

OpenCV的安装 （续）

- 在配置属性-->[链接器 LINKER]的[输入INPUT]中,为项目的Debug配置增加[附加依赖项 Additional Dependencies]:
  - opencv_calib3d243d.lib
  - opencv_contrib243d.lib
  - opencv_core243d.lib
  - opencv_features2d243d.lib
  - opencv_flann243d.lib
  - opencv_gpu243d.lib
  - opencv_highgui243d.lib
  - opencv_imgproc243d.lib
  - opencv_legacy243d.lib
  - opencv_ml243d.lib
  - opencv_objdetect243d.lib
  - opencv_ts243d.lib
  - opencv_video243d.lib

OpenCV的安装 （续）

- 为项目的Release配置增加[附加依赖项 Additional Dependencies]:
  - opencv_contrib243.lib
  - opencv_core243.lib
  - opencv_features2d243.lib
  - opencv_flann243.lib
  - opencv_gpu243.lib
  - opencv_highgui243.lib
  - opencv_imgproc243.lib
  - opencv_legacy243.lib
  - opencv_ml243.lib
  - opencv_objdetect243.lib
  - opencv_ts243.lib
  - opencv_video243.lib

OpenCV的安装 （续）

恭喜您，安装完毕!

OpenCV的特点

（1） 总体描述

- OpenCV是一个基于C/C++语言的开源图像处理函数库
- 其代码都经过优化，可用于实时处理图像
- 具有良好的可移植性
- 可以进行图像/视频载入、保存和采集的常规操作
- 具有低级和高级的应用程序接口（API）
- 提供了面向Intel IPP高效多媒体函数库的接口，可针对你使用
- Intel CPU优化代码，提高程序性能（译注：OpenCV 2.0版的代码已显着优化，无需IPP来提升性能，故2.0版不再提供IPP接口）

OpenCV的特点

(2) 功能

- 图像数据操作（内存分配与释放，图像复制、设定和转换）
- Image data manipulation (allocation, release, copying, setting, conversion).
- 图像/视频的输入输出（支持文件或摄像头的输入，图像/视频文件的输出）
- Image and video I/O (file and camera based input, image/video file output).
- 矩阵/向量数据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）
- Matrix and vector manipulation and linear algebra routines (products, solvers, eigenvalues, SVD).

OpenCV的特点

(2) 功能

- 支持多种动态数据结构（链表、队列、数据集、树、图）
- Various dynamic data structures (lists, queues, sets, trees, graphs).
- 基本图像处理（去噪、边缘检测、角点检测、采样与插值、色彩变换、形态学处理、直方图、图像金字塔结构）
- Basic image processing (filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids).
- 结构分析（连通域/分支、轮廓处理、距离转换、图像矩、模板匹配、霍夫变换、多项式逼近、曲线拟合、椭圆拟合、狄劳尼三角化）
- Structural analysis (connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation).

OpenCV的特点

(2) 功能

- 摄像头定标（寻找和跟踪定标模式、参数定标、基本矩阵估计、单应矩阵估计、立体视觉匹配）
- Camera calibration (finding and tracking calibration patterns, calibration, fundamental matrix estimation, homography estimation, stereo correspondence).
- 运动分析（光流、动作分割、目标跟踪）
- Motion analysis (optical flow, motion segmentation, tracking).
- 目标识别（特征方法、HMM模型）
- Object recognition (eigen-methods, HMM).
- 基本的GUI（显示图像/视频、键盘/鼠标操作、滑动条）
- Basic GUI (display image/video, keyboard and mouse handling, scroll-bars).
- 图像标注（直线、曲线、多边形、文本标注）
- Image labeling (line, conic, polygon, text drawing)

OpenCV的特点

(3) OpenCV模块

- OpenCV模块
- cv - 核心函数库
- cvaux - 辅助函数库
- cxcore - 数据结构与线性代数库
- highgui - GUI函数库
- ml - 机器学习函数库

OpenCV的特点

（4） 视频处理例程(在 <opencv-root>/samples/c/)：

- 颜色跟踪：camshiftdemo
- 点跟踪：lkdemo
- 动作分割：motempl
- 边缘检测：laplace

（5） 图像处理例程（在 <opencv-root>/samples/c/)：

- 边缘检测：edge
- 图像分割：pyramid_segmentation
- 形态学：morphology
- 直方图：demhist
- 距离变换：distrans
- 椭圆拟合：fitellipse

OpenCV 命名规则

(1) 函数名:

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)

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.

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.

OpenCV 命名规则

（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
```


GUI 指令

1） 创建和定位一个新窗口:

```
cvNamedWindow("win1", CV_WINDOW_AUTOSIZE);
cvMoveWindow("win1", 100, 100); // offset from the UL corner of the screen[编辑]
```

2） 载入图像:

```
IplImage* img=0;
img=cvLoadImage(fileName, CV_LOAD_IMAGE_COLOR);
if(!img) printf("Could not load image file: %s\n",fileName);
```


GUI 指令

（3） 显示图像:

cvShowImage("win1",img);

该函数可以在上面建立的窗口（win1）中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("win1");

（5） 改变窗口大小:

cvResizeWindow("win1",100,100); *// new width/heigh in pixels*

GUI 指令

处理键盘事件:

实际上对于键盘输入并没有专门的事件处理程序.
按一定间隔检测键盘输入（适用于循环体中）:

```
int key;

key=cvWaitKey(10); // wait 10ms for input 中止程序等待键盘输入:

key=cvWaitKey(0); // wait indefinitely for input 键盘输入的循环处理程序:

while(1)
{
key=cvWaitKey(10);
if(key==27) break;
switch(key){
 case 'h': ... break;
case 'i': ... break; }
}
```

GUI 指令

处理键盘事件:

实际上对于键盘输入并没有专门的事件处理程序.
按一定间隔检测键盘输入（适用于循环体中）:
int key;

key=cvWaitKey(10); *// wait 10ms for input* 中止程序等待键盘输入:

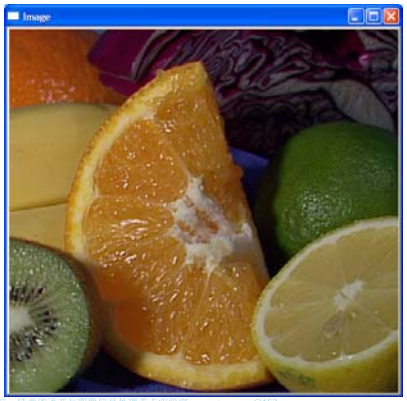
key=cvWaitKey(0); *// wait indefinitely for input* 键盘输入的循环处理程序:

```
while(1)
{
key=cvWaitKey(10);
if(key==27) break;
switch(key){
 case 'h': ... break;
case 'i': ... break; }
}
```


IplImage图像元素的访问

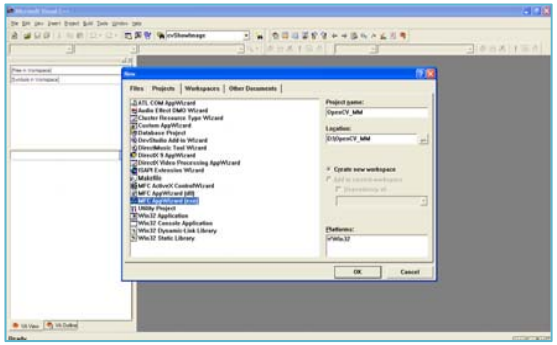
```
typedef struct _IplImage
 {
    int nSize;                  /* IplImage 大小 */
    int ID;                     /* 版本 (=0)*/
    int nChannels;              /* 大多数OPENCV函数支持1,2,3 或 4 个信道 */
    int alphaChannel;           /* 被OpenCV忽略 */
    int depth;                  /* 像素的位深度: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16U,
                                   IPL_DEPTH_16S, IPL_DEPTH_32S, IPL_DEPTH_32F and IPL_DEPTH_64F 可支持 */
    char colorModel[4];         /* 被OpenCV忽略 */
    char channelSeq[4];         /* 同上 */
    int dataOrder;              /* 0 - 交叉存取颜色信道, 1 - 分开的颜色信道. 有cvCreateImage可以创建交叉存取图像 */
    int origin;                 /* 0 - 顶—左结构, 1 - 底—左结构 (Windows bitmaps 风格) */
    int align;                  /* 图像行排列 (4 or 8). OpenCV 忽略它，使用 widthStep 代替 */
    int width;                  /* 图像宽像素数*/
    int height;                 /* 图像高像素数*/
    struct _IplROI *roi;        /* 图像感兴趣区域. 当该值非空只对该区域进行处理 */
    struct _IplImage *maskROI; /* 在 OpenCV中必须置NULL */
    void  *imageId;             /* 同上*/
    struct _IplTileInfo *tileInfo; /* 同上*/
    int imageSize;              /* 图像数据大小(在交叉存取格式下imageSize=image->height*image->widthStep），单位字节*/
    char *imageData;            /* 指向排列的图像数据 */
    int widthStep;              /* 排列的图像行大小，以字节为单位 */
    int BorderMode[4];          /* 边际结束模式, 被OpenCV忽略 */
    int BorderConst[4];         /* 同上 */
    char *imageDataOrigin; /* 指针指向一个不同的图像数据结构（不是必须排列的），是为了纠正图像内存分配准备的 */
 }
 IplImage;
```


示例5：图像的读取和显示



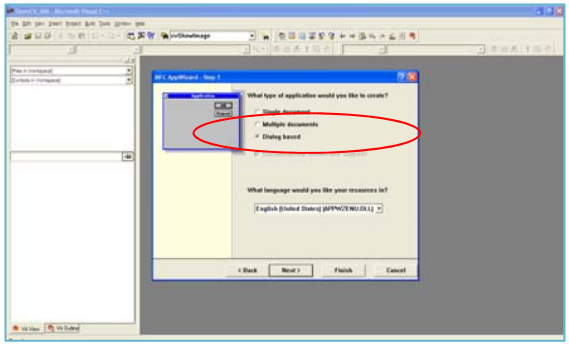
示例5：图像的读取和显示 (续)

- **第一步：创建一个基于对话框的工程OpenCV_MM**



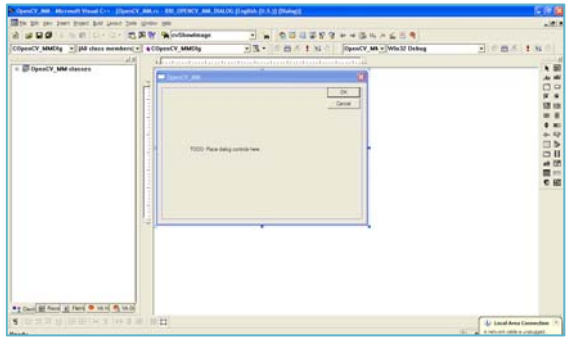
示例5：图像的读取和显示(续)

- 第一步：创建一个基于对话框的工程OpenCV_MM



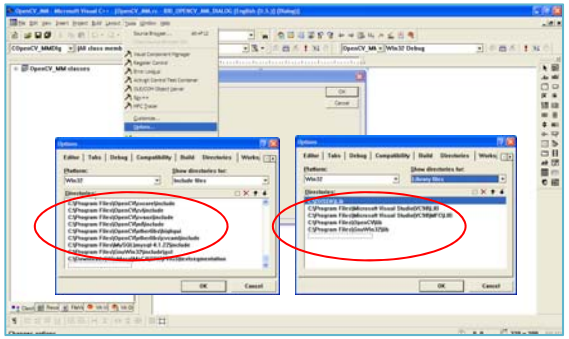
示例5：图像的读取和显示(续)

- 第一步：创建一个基于对话框的工程OpenCV_MM



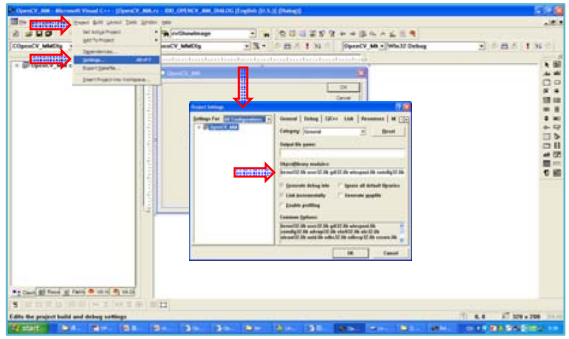
示例5：图像的读取和显示(续)

- 第二步：检查VC6的配置，OpenCV的include和lib目录



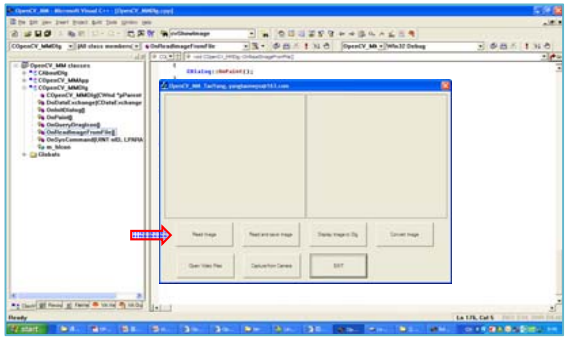
示例5：图像的读取和显示(续)

- 第二步：给该工程加入的OpenCV动态库



示例5：图像的读取和显示(续)

- 第三步：增加一个按钮ReadImage，并添加响应函数OnReadImageFromFile



示例5：图像的读取和显示(续)

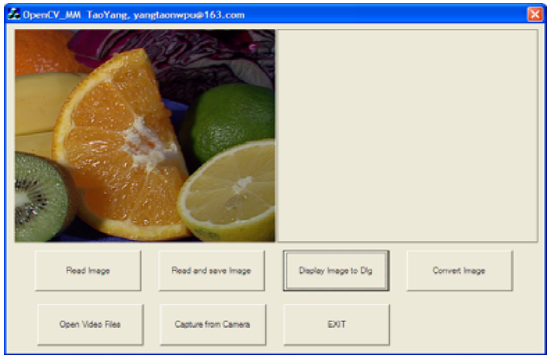
- 第四步：在响应函数中加入以下代码

```
#include "cv.h"
#include "highgui.h"

void COpenCV_MMDlg::OnReadImageFromFile()
{
    // TODO: Add your control notification handler code here

    IplImage* pImg;                          // 声明IplImage指针

    if((pImg = cvLoadImage("fruits.jpg")) != 0 )   // 载入图像
    {
        cvNamedWindow( "Image", 1 );          // 创建窗口

        cvShowImage( "Image", pImg );         // 显示图像

        cvWaitKey(0);                         // 等待按键

        cvDestroyWindow( "Image" );           // 销毁窗口

        cvReleaseImage( &pImg );              // /释放图像

    }

}
```

示例6：图像的创建、复制和保存

```
#include "cv.h"
#include "highgui.h"
void COpenCV_MMDlg::OnReadandSaveImage ()
{
    IplImage* pImg;                          // 声明IplImage指针
    if((pImg = cvLoadImage("Fruits.jpg")) != 0 )   // 载入图像
    {
        IplImage* pImg2 = cvCreateImage(cvGetSize(pImg), pImg->depth, pImg->nChannels);
        cvCopy(pImg, pImg2, NULL);
        cvSaveImage("copyImage.bmp", pImg2);
        cvNamedWindow( "Image", 1 );          // 创建窗口
        cvShowImage( "Image", pImg );         // 显示图像
        cvWaitKey(0);                         // 等待按键
        cvDestroyWindow( "Image" );           // 销毁窗口
        cvReleaseImage( &pImg );              // 释放图像
        cvReleaseImage( &pImg 2);             // 释放图像
    }
}
```

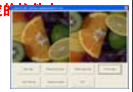
示例7：图像显示在VC对话框上



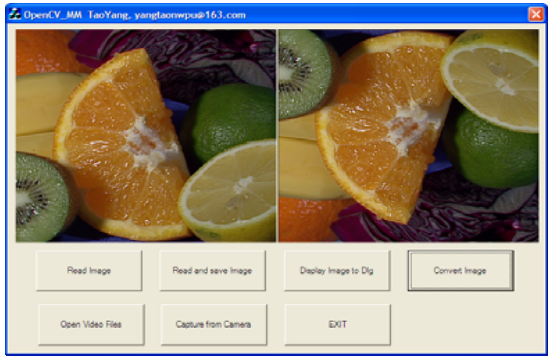
示例7：图像显示在VC对话框上(续)

❑ 增加一个新按钮和相应的函数OnDisplaytoDlg，用于显示图像

```
void COpenCV_MMDlg::OnDisplaytoDlg()
{
    // TODO: Add your control notification handler code here
    IplImage* pImg;                                  //声明IplImage指针
    if((pImg = cvLoadImage("Fruits.jpg")) != 0 )     //载入图像
    {
        CDC *pDC = GetDlgItem(IDC_Window)->GetDC();  //获取对话框指针

        HDC hDC=  pDC->GetSafeHdc();
        CRect rect;
        GetDlgItem(IDC_Window)->GetClientRect(&rect);
        CvvImage cimg;
        cimg.CopyOf(pImg);
        cimg.DrawToHDC(hDC,&rect);                   //将图像显示在指定的
        ReleaseDC(pDC);
    }
```



示例8：将图像旋转并显示



IplImage图像元素的访问

❑ 假设有 8-bit 1－通道的图像 I (IplImage* img):

I(x,y) ~ ((uchar*)(img->imageData + img->widthStep*y))[x]

❑假设有 8-bit 3-通道的图像 I (IplImage* img):

I(x,y)blue ~ ((uchar*)(img->imageData + img->widthStep*y))[x*3]
I(x,y)green ~ ((uchar*)(img->imageData + img->widthStep*y))[x*3+1]
I(x,y)red   ~ ((uchar*)(img->imageData + img->widthStep*y))[x*3+2]

❑现在，通用方法：假设有 N-通道，类型为 T 的图像：
I(x,y)c ~ ((T*)(img->imageData + img->widthStep*y))[x*N + c]

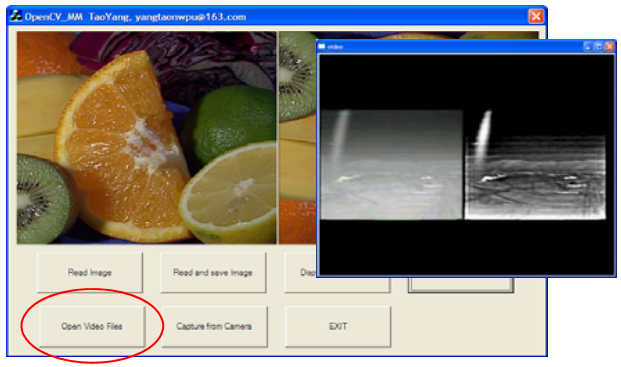
❑或者你可使用宏 CV_IMAGE_ELEM(image_header, elemtype, y, x_Nc)
I(x,y)c ~ CV_IMAGE_ELEM(img, T, y, x*N + c)

示例8：将图像旋转并显示

❑ 增加一个新按钮和相应的函数OnConvertImage，用于显示图像

```
void COpenCV_MMDlg::OnConvertImage()
{
…
    // 图像倒置
    for(int y=0;y<pImg->height;y++)
        for(int x=0;x<pImg->width;x++)
          {
        int r = ((uchar*)(pImg->imageData + pImg->widthStep*y))[x*3];
        int g = ((uchar*)(pImg->imageData + pImg->widthStep*y))[x*3+1];
        int b = ((uchar*)(pImg->imageData + pImg->widthStep*y))[x*3+2];
        ((uchar*)(pImg2->imageData + pImg2->widthStep*(pImg->height-y-1)))[x*3]=r;
        ((uchar*)(pImg2->imageData + pImg2->widthStep*(pImg->height-y-1)))[x*3+1]=g;
        ((uchar*)(pImg2->imageData + pImg2->widthStep*(pImg->height-y-1)))[x*3+2]=b;
        }
…
}
```

示例9：视频文件的读取



示例9：视频文件的读取 (续)

CvCapture* cvCaptureFromFile(const char* filename);

cvCaptureFromFile 初始化从文件中获取视频
Filename 视频文件名。
函数cvCaptureFromFile给指定文件中的视频流分配和初始化CvCapture结构。

IplImage* cvQueryFrame(CvCapture* capture);

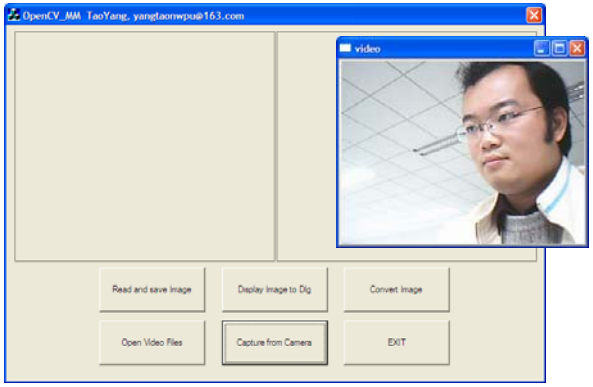
函数cvQueryFrame从摄像头或者文件中抓取一帧，然后解压并返回这一帧。

示例9：视频文件的读取 (续)

- 增加一个新按钮和相应的函数OnOpenVideoFile，用于显示图像

```
void COpenCV_MMDlg::OnOpenVideoFile()
{
    if( (pCapture = cvCaptureFromFile("myvideo2.avi")))          // 打开视频文件
     {
        int nFrmNum = 0;
        while(pFrame = cvQueryFrame( pCapture ))                 // 逐帧读取视频
        {
                nFrmNum++;
                cvShowImage("video", pFrame);                    // 显示图像
                if( cvWaitKey(2) >= 0 )                          // 等待键盘消息
                {
                        cvDestroyWindow("video");
                        break;
                }
        }
    }
    cvReleaseCapture(&pCapture);
}
```

示例10：摄像头数据的读取



示例10：摄像头数据的读取 (续)

cvCaptureFromCAM　初始化从摄像头中获取视频

CvCapture* cvCaptureFromCAM(int index);
要使用的摄像头索引。如果只有一个摄像头或者用哪个摄像头也无所谓，那使用参数-1应该便可以。函数cvCaptureFromCAM给从摄像头的视频流分配和初始化CvCapture结构。目前在Windows下可使用两种接口：Video for Windows（VFW）和Matrox Imaging Library（MIL）；Linux下也有两种接口：V4L和FireWire（IEEE1394）。

IplImage* cvQueryFrame(CvCapture* capture);

函数cvQueryFrame从摄像头或者文件中抓取一帧，然后解压并返回这一帧。

示例11：人脸检测

小　结

OPENCV 本质上只是一个开发工具或者平台而已，没有多大的神秘之处。了解OPENCV无非是在学习或者工作的过程中，能够偷点懒而已。毕竟站在别人的肩膀上会看得远一点。从这个角度上看，以轻松的心态来琢磨OPENCV应有最好的效果，也是学习OpenCV目的所在。

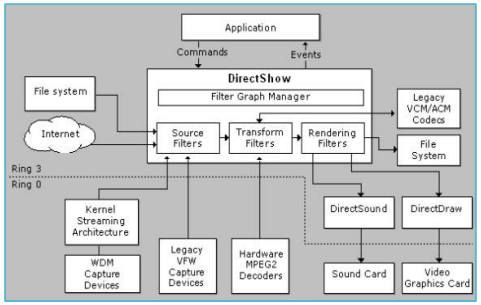
DirectShow

Microsoft Visual C++

Directshow简介

- 流媒体的处理，以其复杂性和技术性，一向广受工业界的关注。特别伴随着因特网的普及，流媒体在网络上的广泛应用，怎样使流媒体的处理变得简单而富有成效逐渐成为了焦点问题。选择一种合适的应用方案，事半功倍。
- DirectShow是微软公司提供的一套在Windows平台上进行流媒体处理的开发包
- DirectShow能够做些什么呢？且看，DirectShow为多媒体流的捕捉和回放提供了强有力的支持。运用DirectShow，我们可以很方便地从支持WDM(Windows Driver Model)驱动模型的采集卡上捕获数据，并且进行相应的后期处理乃至存储到文件中。它广泛地支持各种媒体格式，包括Asf、Mpeg、Avi、Dv、Mp3、Wave等等，使得多媒体数据的回放变得轻而易举。

Directshow简介 (续)



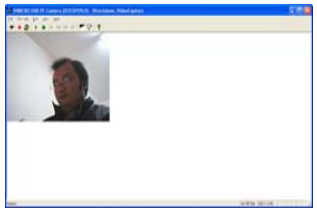
图中央最大的一块即是DirectShow系统。DirectShow使用一种叫Filter Graph的模型来管理整个数据流的处理过程；参与数据处理的各个功能模块叫做Filter；各个Filter在Filter Graph中按一定的顺序连接成一条“流水线”协同工作。大家可以看到，按照功能来分，Filter大致分为三类：Source Filters、Transform Filters和Rendering Filters。Source Filters主要负责取得数据，数据源可以是文件、因特网、或者计算机里的采集卡、数字摄像机等，然后将数据往下传输；Transform Fitlers主要负责数据的格式转换、传输；Rendering Filtes主要负责数据的最终去向，我们可以将数据送给声卡、显卡进行多媒体的演示，也可以输出到文件进行存储。

DirectX Video Processing AppWizard

Yunqiang Chen

Research Assistant in Image Formation Group (Thomas S. Huang)

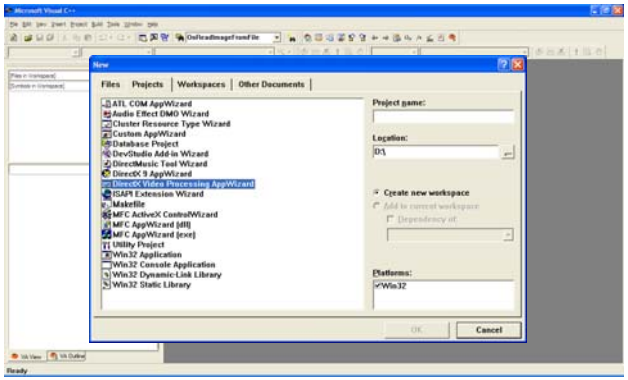
http://www.ifp.uiuc.edu/~chenyq/



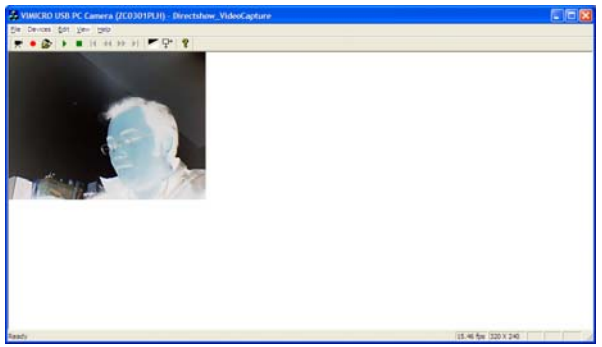
示例12：Directshow视频捕捉

- 第一步：安装 DirectX SDK 8.0 ，添加: DirectX SDK h文件和lib文件路径 (e.g. C:\DXSDK\include and C:\DXSDK\lib) 到 VC options->Directories
- 第二步：下载DxVideoAppWiz.awx and DXVIDEOAPPWIZ.HLP)，并复制到 vc6.0 的安装目录： C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Template.
- 第三步：编译并生成动态链接库(Debug version and Release version) C:\DXSDK\samples\Multimedia\DirectShow\BaseClasses\BaseClasses.dsw
- 第四步：用VC6.0创建新的工程时，选择 "DirectX Video Processing AppWizard”

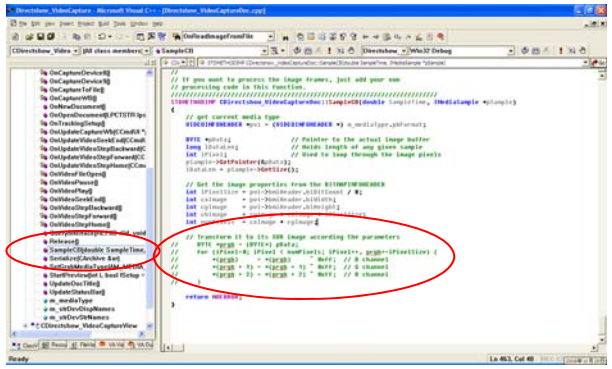
示例12：Directshow视频捕捉



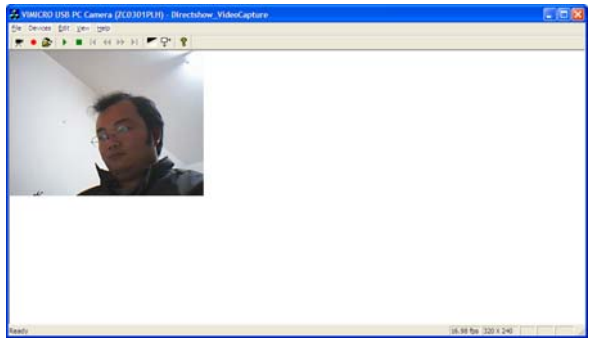
示例12：Directshow视频捕捉(续)



示例12：Directshow视频捕捉(续)



示例12：Directshow视频捕捉(续)



示例12：Directshow视频捕捉(续)

```
STDMETHODIMP CDirectshow_VideoCaptureDoc::SampleCB(double SampleTime, IMediaSample *pSample)
{
    …
    int*R,*G,*B;
    R = new int[cxImage*cyImage];                                   //申请动态内存
    G = new int[cxImage*cyImage];
    B = new int[cxImage*cyImage];
     //读取图像帧数据
    for(int j=0;j<cyImage;j++)
    {
        for(int i=0;i<cxImage;i++)
        {
            B[j*cxImage+i]=(int)pData[j*3*cxImage+3*i];                 // blue
            G[j*cxImage+i]=(int)pData[j*3*cxImage+3*i+1];           // green
            R[j*cxImage+i]=(int)pData[j*3*cxImage+3*i+2];           // red
        }
    }
    // 在这里增加你的视频内容分析函数

    // 释放内存
   …
  return NOERROR;
}
```

常用的视频编辑软件

常用软件

- **视频剪辑**

  VirtualDub 软件可以根据用户的需要，将分散在各个视频资源中的视频片段拼接在一起，然后可以对合成后的内容进行修改编辑.

- **Video (avi,mpg,…) -> Photos (bmp,jpg,…)**

  KMPlayer 高级截取功能

- **制作DEMO**

  SnagIt 著名屏幕捕获，抓图软件

  Camtasia studio 屏幕录制软件

总结

1. 学习几种常见的图像和视频读取编程工具，包括Matlab、OpenCV和 Directshow。详细给出各种开发工具读取单帧图像、多帧图像、视频文件(avi,mpg)、USB摄像头和图像采集卡的源代码和示例程序。介绍其它一些方便我们做研究和开发的软件工具。