# openCV

## Mat

### 显示

using namespace cv;

Mat img = imread("d:\\1.png");

if(!img.data)

return ;

namedWindow("window", CV\_WINDOW\_AUTOSIZE);

imshow("window", img);

### 构造

cv::Mat mat(height, width, CV\_16UC1);

mat.data[pos] = val;

16UC1表示16位深度1通道

### 转换

cv::Mat image = cv::cvarrToMat(ipl);

imwrite("D:\\111.jpg",image);

IplImage result=I;

## 绘制到windows窗口

class DrawImg

{

public:

DrawImg()

{

m\_hbrush = CreateSolidBrush(RGB(0,0,0));

}

void Init(CWnd \*pWnd)

{

pBmpInfo = (BITMAPINFO \*)chBmpBuf;

pBmpInfo->bmiHeader.biSize = sizeof(BITMAPINFOHEADER);

pBmpInfo->bmiHeader.biPlanes = 1;

pBmpInfo->bmiHeader.biCompression = BI\_RGB;

for (int i = 0;i<256;i++)

{

pBmpInfo->bmiColors[i].rgbBlue = i;

pBmpInfo->bmiColors[i].rgbGreen = i;

pBmpInfo->bmiColors[i].rgbRed = i;

pBmpInfo->bmiColors[i].rgbReserved = i;

}

pWnd->GetClientRect(wrect);

CDC \*pdc = pWnd->GetDC();

hdc = pdc->GetSafeHdc();

}

void Draw(IplImage \*pImg)

{

pBmpInfo->bmiHeader.biWidth = pImg->width;

pBmpInfo->bmiHeader.biHeight = pImg->height;

pBmpInfo->bmiHeader.biBitCount = pImg->depth \* pImg->nChannels;

int x = 0, y = 0, width = pBmpInfo->bmiHeader.biWidth, height = pBmpInfo->bmiHeader.biHeight;

if (NULL != pImg->roi) { x = pImg->roi->xOffset; y = pImg->roi->yOffset; width = pImg->roi->width; height = pImg->roi->height; }

FillRect(hdc,wrect,m\_hbrush);

int DesX =0;

int DesY =wrect.Height();

int DesWidth = wrect.Width();

int DesHeight = wrect.Height();

double ratio = pImg->width/(double)pImg->height;

int WidthRequest = ratio \* wrect.Height();

int HeightRequest = wrect.Width()/ratio;

if (wrect.Width() < WidthRequest)

{

DesHeight = DesWidth / ratio;

DesY = (wrect.Height() - DesHeight)/2 + DesHeight;

}

else if (wrect.Height() < HeightRequest)

{

DesWidth = DesHeight\*ratio;

DesX = (wrect.Width() - DesWidth)/2;

}

::SetStretchBltMode(hdc, COLORONCOLOR);

::StretchDIBits(hdc,

DesX, DesY, DesWidth, -DesHeight,

x, y, width, height,

pImg->imageData, pBmpInfo,

DIB\_RGB\_COLORS, SRCCOPY);

}

private:

char chBmpBuf[2048];

BITMAPINFO \*pBmpInfo;

CRect wrect;

HDC hdc;

HBRUSH m\_hbrush;

};

=========================使用

DrawImg m\_DrawImg;

m\_DrawImg.Init(this);

m\_DrawImg.Draw(pImg);

Mat img;

int depth = img.depth();

if(CV\_8U == depth || CV\_8S == depth) depth = 8;

else if(CV\_16U == depth || CV\_16S == depth) depth = 16;

else if(CV\_32S == depth || CV\_32F == depth) depth = 32;

else if(CV\_64F == depth) depth = 64;

## 视频

### 播放avi

cvNamedWindow("aviWindow",CV\_WINDOW\_AUTOSIZE);

CvCapture\* capture=cvCreateFileCapture("d:\\1.avi");

IplImage\* frame;

while(1)

{

frame=cvQueryFrame(capture);

if(!frame)

break;

cvShowImage("aviWindow",frame);

char c=cvWaitKey(33); //相当于设置帧速率

if(c==27)break; //表示按esc退出

}

cvReleaseCapture(&capture);

cvDestroyWindow("aviWindow");

### avi设置

获取avi总帧数

double frameCount=cvGetCaptureProperty(capture,CV\_CAP\_PROP\_FRAME\_COUNT);

第二个参数可以是：

CV\_CAP\_PROP\_POS\_MSEC - 影片目前位置，为毫秒数或者视频获取时间戳

CV\_CAP\_PROP\_POS\_FRAMES - 将被下一步解压/获取的帧索引，以0为起点

CV\_CAP\_PROP\_POS\_AVI\_RATIO - 视频文件的相对位置（0 - 影片的开始，1 - 影片的结尾)

CV\_CAP\_PROP\_FRAME\_WIDTH - 视频流中的帧宽度

CV\_CAP\_PROP\_FRAME\_HEIGHT - 视频流中的帧高度

CV\_CAP\_PROP\_FPS - 帧率

CV\_CAP\_PROP\_FOURCC - 表示codec的四个字符

CV\_CAP\_PROP\_FRAME\_COUNT - 视频文件中帧的总数

设置播放位置

cvSetCaptureProperty(capture,CV\_CAP\_PROP\_POS\_FRAMES,frameCount/2);

第二个参数可以是：

CV\_CAP\_PROP\_POS\_MSEC - 从文件开始的位置，单位为毫秒

CV\_CAP\_PROP\_POS\_FRAMES - 单位为帧数的位置（只对视频文件有效）

CV\_CAP\_PROP\_POS\_AVI\_RATIO - 视频文件的相对位置（0 - 影片的开始，1 - 影片的结尾)

CV\_CAP\_PROP\_FRAME\_WIDTH - 视频流的帧宽度（只对摄像头有效）

CV\_CAP\_PROP\_FRAME\_HEIGHT - 视频流的帧高度（只对摄像头有效）

CV\_CAP\_PROP\_FPS - 帧率（只对摄像头有效）

CV\_CAP\_PROP\_FOURCC - 表示codec的四个字符（只对摄像头有效）

### 捕获摄像头

cvNamedWindow("aviWindow",CV\_WINDOW\_AUTOSIZE);

//CvCapture\* capture=cvCreateCameraCapture(0);

CvCapture\* capture=cvCaptureFromCAM(0);//代表第一个摄像头,该函数必须在主线程中执行

if(!capture){exit(0);}

IplImage\* frame;

while(1)

{

frame=cvQueryFrame(capture);

if(!frame)

break;

cvShowImage("aviWindow",frame);

char c=cvWaitKey(33); //相当于设置帧速率

if(c==27)break; //表示按esc退出

}

cvReleaseCapture(&capture);

cvDestroyWindow("aviWindow");

### 保存为avi

程序运行前确保avi文件名不存在，保存文件的路径存在，否则创建writer会失败。

如果要将保存avi和摄像头捕捉显示同时进行，注意避免同时访问frame的冲突。

(int)cvGetCaptureProperty(cap,CV\_CAP\_PROP\_FOURCC)可以获取编码模式

全局变量：

CvVideoWriter \*writer=NULL;

保存过程：

int fps = 12;

CvSize size=cvSize((int)cvGetCaptureProperty(capture,CV\_CAP\_PROP\_FRAME\_WIDTH),(int)cvGetCaptureProperty(capture,CV\_CAP\_PROP\_FRAME\_HEIGHT));

CTime time=CTime::GetCurrentTime();

CString str=time.Format("%Y-%m-%d %H-%M-%S");

CString file="d:\\avi\\"+str+".avi";//ANCI时

writer=cvCreateVideoWriter(file,CV\_FOURCC('I', 'Y', 'U', 'V'),fps,size);//此参数传递-1，可以查看和选择已有的编码模式，也可以传入其他数字进行测试。

while((frame=cvQueryFrame(capture))!=NULL)

{

Sleep(fps);

cvWriteFrame(writer, frame);

}

析构：

cvReleaseVideoWriter(&writer);

## IplImage

### IplImage构造

int depth = IPL\_DEPTH\_8U;

int channels = 1; //黑白 1，RGB 3

IplImage \*pImg = cvCreateImage(cvSize(Width,Height),depth,channels);

for (int row = Height -1 ; row >= 0; --row)

{

for (int col = 0; col < Width; ++col)

{

int pos = pImg->widthStep \* row + col;//如果depth为IPL\_DEPTH\_16U int pos = pImg->widthStep \* row + col\*2;

pImg->imageData[pos] = dat;

}

}

### IplImage结构体赋值

void InitIpl(IplImage &img,char\* DibBuf,int width,int height,int depth,int nchannel)

{

img.nSize = sizeof(IplImage);

img.ID = 0;

img.alphaChannel =0;

img.depth = depth;

strcpy\_s(img.colorModel,4,"RGB");

strcpy\_s(img.channelSeq,4,"BGR");

img.dataOrder = 0;

img.origin = 0;

img.align = 4;

img.roi = NULL;

img.maskROI = NULL;

img.imageId = NULL;

img.tileInfo = NULL;

memset(img.BorderMode,4,0);

memset(img.BorderConst,4,0);

img.imageDataOrigin = 0;

img.width = width;

img.height = height;

img.nChannels = nchannel;

img.imageSize = img.width \* img.height \* img.nChannels;

img.widthStep = img.width \* img.nChannels;

img.imageData = DibBuf;

}

### 读取图像

IplImage \*pImg = cvLoadImage("D:\\1.png",CV\_LOAD\_IMAGE\_ANYCOLOR|CV\_LOAD\_IMAGE\_ANYDEPTH);

第二个参数可以不写，默认是将图像以8位3通道载入。

CV\_LOAD\_IMAGE\_GRAYSCALE代表以单通道载入

第二个参数iscolor

>0 强制以彩色（3通道）

=0 灰度

<0 取决于图像本身

### 弹窗显示

可以在app类initInstance创建对话框前调用函数，取代对话框。

IplImage\* img=cvLoadImage("d:\\1.jpg"); //将图像加载到内存中

cvNamedWindow("picWindow",CV\_WINDOW\_AUTOSIZE);

//窗口名为picWindow,第二个参数默认0，表示任意拉伸。CV\_WINDOW\_AUTOSIZE表示窗口大小是图像的大小，不能拉伸

cvShowImage("picWindow",img);

//使用创建的窗口显示图片

cvWaitKey(0);//使程序暂停，等待用户按下键盘，参数0代表一直等待，如果参数为正数x，代表等待x毫秒

cvReleaseImage(&img);//释放内存中的图像

cvDestroyWindow("picWindow");//销毁创建的窗口

### ROI

cvSetImageROI(img,cvRect(x,y,width,height));

cvResetImageROI(img);

在 cvSetImageROI和cvCopy函数时，一定要注意图像的大小（或ROI大小），避免越界和图像大小不一。

### 保存图片

可以将img保存为任意格式图片，保存图片的目录必须存在

cvSaveImage("d:\\x.png",img);

### 创建与销毁

IplImage\* img = cvCreateImage(cvSize(width,height),IPL\_DEPTH\_8U,3);

cvReleaseImage(&img);

创建时，也可以使用cvGetSize(img)作为参数

IplImage\* img = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);

### 归零

cvZero(img);

### 设置颜色

cvSet(pImg, cvScalar(0x00,0xDB, 0x00));

### 翻转

cvFlip(img,img,0);

IplImage中的origin成员为0，代表上下翻转，1代表左右翻转。

### 复制

cvCopy(src, dst);//将src拷贝到dst中

IplImage\* dst=cvCloneImage(img);//将img完整复制一份

### 获取尺寸

int cx=cvGetSize(img).width;

int cy=cvGetSize(img).height;

或者

int cx=img->width;

int cy=img->height;

### 调整大小

将src变为dst的大小，dst为缩放后的图像。

IplImage\* dst = cvCreateImage(cvSize(200,60),src->depth,src->nChannels);

cvResize(src,dst);

cvReleaseImage(&dst);//放在函数最后（处理图像结束后）释放内存

### 裁剪

将src进行裁剪，裁剪后的图像为dst。

int x=0,y=0,width=200,height=200;

IplImage\* dst = cvCreateImage(cvSize(width,height),src->depth,src->nChannels);

cvSetImageROI(src,cvRect(x,y,width,height));//设置感兴趣的区域，也就是用来操作的区域

cvCopy(src,dst);

### 分离通道

cvSplit(src,r,g,b,NULL);

将src的rgb分别放到三个通道图中,r,g,b三个IplImage都是单通道的

### 平滑处理

将src平滑处理后放到dst

cvSmooth(src,dst,CV\_GAUSSIAN ,3,3);

第三个以后的参数都有默认值。

第三个参数可以是:

CV\_BLUR\_NO\_SCALE (简单不带尺度变换的模糊) - 对每个象素的 param1×param2 领域求和。如果邻域大小是变化的，可以事先利用函数 cvIntegral 计算积分图像。

CV\_BLUR 对每个象素param1×param2邻域 求和并做尺度变换 1/(param1.param2).

CV\_GAUSSIAN对图像进行核大小为 param1×param2 的高斯卷积

CV\_MEDIAN对图像进行核大小为param1×param1 的中值滤波(邻域是方的).

CV\_BILATERAL (双向滤波) - 应用双向 3x3 滤波，彩色sigma=param1，空间 sigma=param2. 平滑操作的第一个参数.

### 融合

两个图象可以是任何象素类型，只要它们的类型相同。它们可以是单通道或是三通道，只要它们相符。必须有相同的象素类型。这些图象可以是不同的尺寸，但它们的ROI必须有相同的大小。

融合时候，以图像本身大小为基准，而不是以显示界面的大小为基准

IplImage \*dst, \*src;

dst存储合成后的图

int width =200,height =300; //设置ROI大小相同

cvSetImageROI(dst, cvRect(0,0,width,height));

cvSetImageROI(src, cvRect(0,0,width,height));

//前两个参数代表图像左上角位置

double alpha =0.5;

double beta =1;//设置为0和1可以让上面的图盖在底图上

double gamma=0;

cvAddWeighted(dst, alpha, src,beta,gamma,dst);

//运算是dst=dst\*alpha+src\*beta+gamma;

cvResetImageROI(dst); //因为ROI变小，所以重置回原来的大小

### 去白色融合

去掉src2的白色，融合图放在src3

IplImage\* src3 = cvCreateImage(cvSize(width,height),src1->depth,src1->nChannels);

cvAnd(src1,src2,src3);

cvAndS(src,cvScalar(255,255,0),src3); //可以用指定颜色进行位与

### 调色

为图片增加红色（红通道加了100，如果是负数代表减少颜色），cvScalar可以有四个参数BGRA

cvAddS(img,cvScalar(0,0,100),img);

### 反色

for(int i=0;i != height; ++ i)

{

for(int j=0;j != width; ++ j)

{

for(int k=0;k != channels; ++ k)

{

data[i\*step+j\*channels+k]=255-data[i\*step+j\*channels+k];

}

}

}

### 边缘检测

IplImage\* img=cvLoadImage("d:\\1.jpg",CV\_LOAD\_IMAGE\_GRAYSCALE); //灰度图像

IplImage\* CannyImg=cvCreateImage(cvGetSize(img),IPL\_DEPTH\_8U,1);

cvCanny(img,CannyImg,50,150,3);//输出为CannyImg

### 遍历IplImage处理图像

将img中蓝色最大化

IplImage\* img = cvCreateImage(cvSize(src->width,src->height),IPL\_DEPTH\_8U,3);

cvCopy(src, img);

for(int y=0;y<img->height;y++)

{

uchar\* ptr=(uchar\*)(img->imageData+y\*img->widthStep);//ptr指向一行的数据长度，也就是指向的内存大小为一行的内存大小

for(int x=0;x<img->width;x++)

{

ptr[3\*x+0]=255;//0，1，2分别代表蓝绿红，3\*x是因为这是3通道的图像

}

}

ptr指向的行内存排列应该是：

B G R B G R 。。。。。。。。。

3\*0+0 3\*1+1

### 替代ROI方式

使img的部分区域红色增加100

int x=10,y=10,width=400,height=400;//感兴趣区域

IplImage\* sub\_img=cvCreateImageHeader(cvSize(width,height),img->depth,img->nChannels);

sub\_img->origin=img->origin;

sub\_img->widthStep=img->widthStep;

sub\_img->imageData=img->imageData + img->widthStep\*y + img->nChannels\*x;

cvAddS(sub\_img,cvScalar(0,0,100),sub\_img);//让红通道增加100

cvReleaseImageHeader(&sub\_img);

ROI只能串行处理，且不断设置和重置，该方式可以保持一副图像的多个子区域处于活动状态下。

### 拉伸变换

不能对一幅图进行两次拉伸，需要多次拉伸时，用多个dst接收src的变换，然后设置ROI进行合成即可。

CvPoint2D32f srcTir[4],dstTir[4];

CvMat \*warp\_mat=cvCreateMat(3,3,CV\_32FC1);

IplImage \*src=cvLoadImage("d:\\1.jpg");

IplImage \*dst=cvCloneImage(src);

dst->origin=src->origin;

cvZero(dst);

///src中要进行变换的四个点坐标

srcTir[0].x=52;

srcTir[0].y=0;//LT

srcTir[1].x=src->width-52;

srcTir[1].y=0;//RT

srcTir[2].x=0;

srcTir[2].y=300;//LB

srcTir[3].x=src->width;

srcTir[3].y=300;//RB

//变换后的四个点坐标

dstTir[0].x=0;

dstTir[0].y=0;

dstTir[1].x=src->width;

dstTir[1].y=0;

dstTir[2].x=0;

dstTir[2].y=300;

dstTir[3].x=src->width;

dstTir[3].y=300;

cvGetPerspectiveTransform(srcTir,dstTir,warp\_mat);

cvWarpPerspective(src,dst,warp\_mat);

////对dst进行显示///////

cvReleaseImage(&dst);

cvReleaseMat(&warp\_mat);

### 转换颜色类型

#pragma comment(lib,"opencv\_objdetect248d")

#pragma comment(lib,"opencv\_imgproc248d")

IplImage\* gray = cvCreateImage(cvSize(img->width, img->height), 8, 1);

cvCvtColor(img, gray, CV\_BGR2GRAY);

## 绘图

### 矩形

在两个点定位处绘制一个白色正方形,最后一个参数是线条宽度，如果是-1，代表画一个填充矩形。

cvRectangle(img,cvPoint(10,10),cvPoint(50,50),cvScalar(255,255,255),-1);

### 圆

cvCircle(pImg, cvPoint(x,y), 8,cvScalar(0,0,255), CV\_FILLED);

//最后一个参数是线条宽度，cvScalar(0,0,255) BGR

//8代表圆的半径，最后一个参数是线条宽度

### 椭圆

cvEllipse(img,cvPoint(x,y),cvSize(width,height),0,0,360,cvScalar(255,255,255),1);

//第二个参数代表中心点，第三个参数代表了椭圆的两个半径，第五个是偏离主轴的角度，六七代表弧线开始和结束角度，一般都是0和360。最后一个是线条宽度

### 直线

cvLine(img,cvPoint(10,10),cvPoint(50,50),cvScalar(255,255,255),1);

最后一个参数是线宽，默认为1

### 文字

只能绘制英文

CvFont font;

cvInitFont(&font,CV\_FONT\_HERSHEY\_SCRIPT\_SIMPLEX,hscale,vscale,0,thick);

cvPutText(img,"3",cvPoint(100,200),&font,cvScalar(0,0,255));

hscale和vscale代表字体宽和高的间距，thick代表字体线条宽度,thick的前一个参数代表是否字体倾斜

## 计时

获取秒差:

double timeSpan = (GetTickCount() - time1 )/GetTickFrequency();

## 图像处理

### 腐蚀膨胀

cvErode()腐蚀后cvDilate()膨胀，叫作开操作，那些离散点或游丝线、毛刺就被过滤  
cvDilate()膨胀后cvErode()腐蚀，叫作闭操作，那些断裂处就被缝合。

函数的输入和输出可以是一个指针

IplConvKernel \*ConvKernel = cvCreateStructuringElementEx(3, 3, 1, 1, CV\_SHAPE\_RECT);

//表示3\*3矩形进行操作

cvErode(pImg,pImg, ConvKernel, 1);//腐蚀, 最后一个参数代表腐蚀次数

cvDilate(pImg, pImg, ConvKernel, 4);//膨胀

### 查找轮廓

cvFindContours

需要二值图像，非0作为1，0保持不变。识别1

0是背景，1是图案

会按照边缘连续性，图案中空心则会识别两个轮廓

cvThreashold对单通道操作，一般是对灰度图转换为2值图

cvThreshold(pImg,pImg, 128, 1, CV\_THRESH\_BINARY);//将大于128的灰度值（白色）变为1，小于128的变成0

第三个参数代表阈值，第四个代表转换后最大值(0,1)

CV\_THRESH\_BINARY: value > threshold ? max\_value : 0

CV\_THRESH\_BINARY\_INV则相反处理

CvMemStorage\* storage = cvCreateMemStorage(0);

CvSeq\* contour = 0;

cvFindContours(gray, storage, &contour);//gray为2值图像

cvZero(gray);

for (; contour != 0; contour = contour->h\_next)

{

double contourSize = fabs(cvContourArea(contour));//轮廓面积

///CvRect rect = cvBoundingRect(contour, 0);//轮廓外接矩形

if(contourSize > 200) continue;//挑选面积小的

cvDrawContours(pImg, contour,

cvScalar(255,255,255),cvScalar(255,255,255),

-1,CV\_FILLED);

//pImg是原始图，gray是二值图

//CV\_FILLED代表完全填充，cvScalar参数可以是1到4个，和图像位数相关

}

cvReleaseMemStorage(&storage);

int GetCountersNum(CvSeq\* contour)

{//获取轮廓数量

int total = 0;

CvSeq\* contour\_bak = contour;

for (; contour != 0; contour = contour->h\_next) total++;

return total;

}

## gamma增亮

void Gamma(int width,int height,BYTE \*pImg)

{

static const double gamma = 0.45;

static const double offset = 0.055;

static const double threshold = 0.0031308;

static const double maxgray = 255;

static const double s = ((1 + offset)\*pow(threshold, gamma) - offset) / threshold;

for (int row=0; row< height; ++row)

{

for (int col = 0 ; col < width;++col)

{

BYTE \*pVal = pImg + row\*width + col;

double gray = (\*pVal) / maxgray;

double SI = 0;

if (gray <= threshold)

SI = s\*gray\*maxgray;

else

SI = maxgray\*((1 + offset)\*pow(gray, gamma) - offset);

\*pVal = (BYTE)SI;

}

}

}

## OpenCVSharp

### 图像显示

Mat src = new Mat("D:\\1.jpg", ImreadModes.AnyColor);

//Cv2.ImShow("im", src);

//Cv2.WaitKey();

Bitmap bitmap = OpenCvSharp.Extensions.BitmapConverter.ToBitmap(src);//using System.Drawing;

using (MemoryStream stream = new MemoryStream())

{

bitmap.Save(stream, System.Drawing.Imaging.ImageFormat.Jpeg); //不带透明度

stream.Position = 0;

BitmapImage result = new BitmapImage();

result.BeginInit();

result.CacheOption = BitmapCacheOption.OnLoad;

result.StreamSource = stream;

result.EndInit();

result.Freeze();

img.Source = result;

}

### 获取点阵数据

IntPtr datpr = src.Data;

### 创建图像

Mat.Zeros(row, col, OpenCvSharp.MatType.CV\_8UC1);

### 获取设置像素

int pix = src.At<byte>(i, j);

ret.SetArray(i, j, r);

### 获取图像中最大最小值

double min, max;

src.MinMaxIdx(out min, out max);

### 图像合并

Mat src = new Mat("D:\\a.jpg", ImreadModes.AnyColor | ImreadModes.AnyDepth);

Mat src2 = new Mat("D:\\b.jpg", ImreadModes.AnyColor | ImreadModes.AnyDepth);

if (src.Rows != src2.Rows) return;

Mat MatrixCom = new Mat(src.Rows, src.Cols + src2.Cols, src.Type());

Mat temp = MatrixCom.ColRange(0, src.Cols);

src.CopyTo(temp);

temp = MatrixCom.ColRange(src.Cols, src.Cols + src2.Cols);

src2.CopyTo(temp);

Cv2.ImWrite("D:\\c.jpg", MatrixCom);