**OpenCV读取TDA2x录像**

读取TDA2x录的视频Encode\_V1129\_167\_1.mp4，非常的卡。视频信息如下图：



while(1)

479 {

480 Mat frameVideo1;

481

482 long long timeProf = dpuGetTaskProfile(taskConv);

483 gettimeofday(&timeStart, nullptr);

484

485  **namedWindow("resized:");**//creat a window;

486

487 **capture1.read(frameVideo1);**//read current frame from video

488 gettimeofday(&timeEnd, nullptr);

489 double timeElapsed = (timeEnd.tv\_sec - timeStart.tv\_sec)\*1000. +

490 (timeEnd.tv\_usec - timeStart.tv\_usec)/1000.;

491 printf("capture read time:%lf (ms)\n", timeElapsed);

492

493 if(frameVideo1.empty())//only onece is enough.

494 {

495 break;

496 }

497 // resize(srcImg1,dstImg1, Size(WideSize, LengthSize), (0,0), (0,0), INTER\_LINEAR);

498 // cout << "frame:" << counter << endl;//display frame in order.

499 //run BSD\_left here:

500 //runBSD(taskConv, taskFC, dstImg1);

501 **cv::imshow("resized:", frameVideo1);**

502

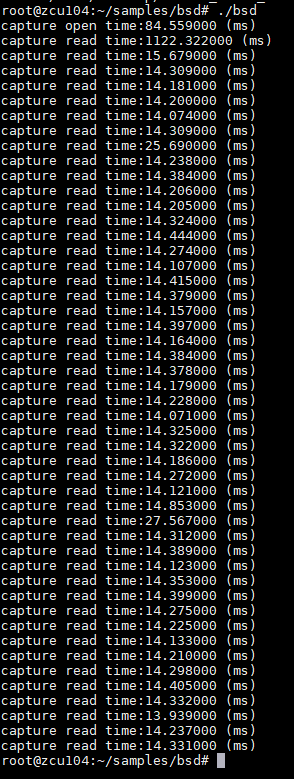
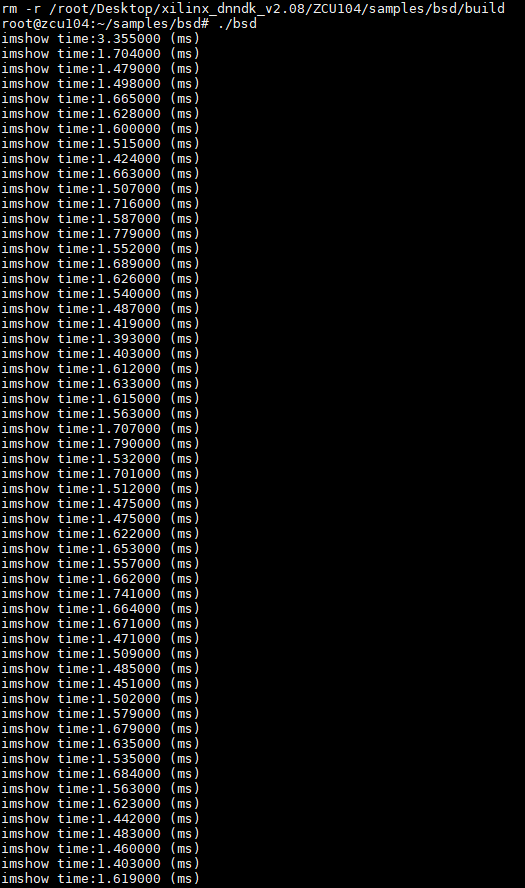
503 char c = cvWaitKey( 2 );//wait 33ms

504 if( c == 27 ) break;// ESC

505 }

506 capture1.release();

首先测试了红色代码的时间，读入视频帧。Capture read时间，如下图<1>：

<1> Capture read <2>imshow

然后测试imshow的时间，如上图<2>

**配置videocapture读取yuyv的视频：**



**存储、处理yuv图像数据：**

OpenCV读到的yuyv视频数据，直接都是Y/U/V的数据流。可以存放到FILE文件中，使用二进制方式读写。rb+

fread/fwrite



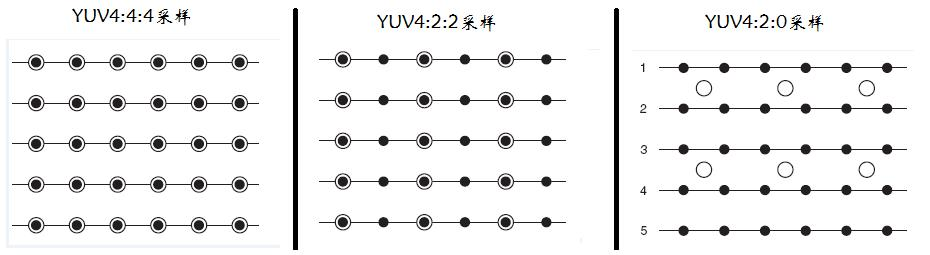
OpenCV处理yuv格式数据举例：

二. 读取以YUV形式保存的视频文件，并将其转为RGB图片，然后显示

首先先简单介绍下YUV数据数据格式。YUV，分为三个分量，“Y”表示明亮度（Luminance或Luma），也就是灰度值；而“U”和“V” 表示的则是色度（Chrominance或Chroma），作用是描述影像色彩及饱和度，用于指定像素的颜色。

YUV码流的存储格式其实与其采样的方式密切相关，主流的采样方式有三种，YUV4:4:4，YUV4:2:2，YUV4:2:0，关于其详细原理，可以通过网上其它文章了解，这里我想强调的是如何根据其采样格式来从码流中还原每个像素点的YUV值，因为只有正确地还原了每个像素点的YUV值，才能通过YUV与RGB的转换公式提取出每个像素点的RGB值，然后显示出来。

用三个图来直观地表示采集的方式吧，以黑点表示采样该像素点的Y分量，以空心圆圈表示采用该像素点的UV分量。



先记住下面这段话，以后提取每个像素的YUV分量会用到

1.YUV 4:4:4采样，每一个Y对应一组UV分量。

2.YUV 4:2:2采样，每两个Y共用一组UV分量。

3.YUV 4:2:0采样，每四个Y共用一组UV分量。

个人觉得YUV格式主要的优点就是可以压缩图片的数据量，节省内存。可能对单幅图来说节省的内存不是很多，但是对于一段视频来说，省下的内存就是很大了。

下面就以一副RGB24图片为例，如果将其以RGB形式保存的话，需要的size=width\*height\*3bit . 若是以YUV420保存的话，size=width\*height\*1.5bit 。

对于一副8\*4的图片来说，size=width\*height+1/2 \*width\*height=3/2\*width\*height=3/2\*8\*4



不过缺点就是在将YUV420数据转换为RGB时，会出现图像像素点信息的丢失，不能完全复现出原图。(YUV444可以完全复现，但是数据不会被压缩)

**具体代码如下：**

#include <opencv/cv.h>

#include <fstream>

#include <iostream>

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include <opencv2/video/background\_segm.hpp>

using namespace std;

using namespace cv;

const int width = 352;

const int height = 288;

const int framesize = width \* height \* 3 / 2; //一副图所含的像素个数

typedef struct planet

{

char name[framesize];

double population;

double g;

} PLANET;

int main()

{

/////////////////////////////////////////////////////////////

// 计算视频的帧数，怎样替换成c语言形式的？

PLANET pl;

ifstream fin;

fin.open("hall\_cif.yuv", ios\_base::in | ios\_base::binary);

if (fin.fail())

{

cout << "the file is error" << endl;

return -1;

}

fin.seekg(0, ios::end); //设置文件指针到文件流的尾部

streampos ps = fin.tellg(); //指出当前的文件指针

unsigned long NumberPixe = ps;

cout << "file size: " << ps << endl; //输出指针的位置

unsigned FrameCount = ps / framesize; //帧大小

cout << "frameNuber: " << FrameCount; //输出帧数

fin.close();

/////////////////////////////////////////////////////////////////

Ptr<BackgroundSubtractorMOG2> pBgmodel = createBackgroundSubtractorMOG2();

pBgmodel->setVarThreshold(20);

FILE\* fileIn = fopen("hall\_cif.yuv", "rb+");

unsigned char\* pYuvBuf = new unsigned char[framesize]; //一帧数据大小

//存储到图像

namedWindow("yuv", 1);

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

{

//从数据流中读取 按长度读取数据 下一次循环直接从上一帧的末尾开始读取

fread(pYuvBuf, framesize\*sizeof(unsigned char), 1, fileIn); //在整个yuv中截取一帧图像读入

Mat yuvImg;

yuvImg.create(height \* 3 / 2, width, CV\_8UC1); //创建新的图片的大小

memcpy(yuvImg.data, pYuvBuf, framesize\*sizeof(unsigned char));

Mat rgbImg;

cvtColor(yuvImg, rgbImg, CV\_YUV2BGR\_I420);

char key = waitKey(100);

if (key == 27)// 按键“ESC”退出

break;

imshow("yuv", yuvImg); //只显示y分量

imshow("rgbImg", rgbImg);

printf("第 %d 帧\n", i);

}

fclose(fileIn);

cvDestroyWindow("yuv");

cin.get();

cin.get();

return 0;

}

---------------------

原文：https://blog.csdn.net/qq\_24193303/article/details/80512134

**Videocapture类使用介绍**

https://blog.csdn.net/frosuccess/article/details/79589778