Opencv笔记3 读取摄像头、视频文件的数据并且保存

1. 读取生成视频数据（C++）

视频数据读取可以从摄像头和视频文件，打开之后调整适当的参数即可

视频的读取使用的是 VideoCapture，视频的生成使用的是VideoWriter

代码如下：

#include <iostream>

#include <string>

#include <opencv2\opencv.hpp>

using namespace std;

using namespace cv;

int main(){

VideoCapture cap;

cap.open(0);// 打开摄像头也可以打开视频文件，摄像头默认是0，然后1,2,3,

if (!cap.isOpened()){

cerr << "can not open camera or file" << endl;

return 1;

}

//获取摄像头拍摄图像对应的参数，主要是视频的长度和宽度，只有写入的视频文件和此参数一致，才可以生成正确的视频文件，不然不可以，这一个方面尝试了好久

Size refSize(cap.get(CV\_CAP\_PROP\_FRAME\_WIDTH), cap.get(CV\_CAP\_PROP\_FRAME\_HEIGHT));

//cout << refSize.width << " " << refSize.height << " " << cap.get(CV\_CAP\_PROP\_FPS) << endl;

int fps = 25;//视频的帧

VideoWriter writer("myvideo.avi", CV\_FOURCC('F', 'L', 'V', '1'), fps, refSize, true);

if (!writer.isOpened())

{

cerr << "can not create video file.\n" << endl;

return (1);

}

Mat frame(refSize,CV\_8UC3);

namedWindow("camera", 1);

for (int i = 0; i < 150; i++){

cap >> frame;

if (frame.empty()){

break;

}

cout << "frame " << i << endl;

writer << frame;

imshow("camera", frame);

cvWaitKey(1000 / fps);//等一段时间，让其看起来是正常的视频

}

frame.release();

writer.release();

cap.release();

}

其实我们这样生成的视频文件感觉非常大，每一帧视频保存的都是图片数据.

1. 对于图片文件的读写

对于c++方式，我们使用的是Mat来保存图像数据，使用的是imread来读取图像的数据，使用imwrite写入图像数据，生成文件。

Mat imread(const string &filename, int flag = 1)

对于参数flag：flag < 0 ,函数不会对图像进行转换，flag = 0 则表示将图像转成单通道的图像；flag > 0表示返回的为3通道图像。

bool imwrite(const string& filename , InputArray image, const vector<int>& params= vector<int>());

filename确定文件的名称包括后缀扩展名，推荐使用PNG无损压缩格式，BMP使用的是无损格式，文件比较大；JPEG文件小，但是是有损压缩格式，会丢信息；

params参数可以指定文件格式的一些细节信息，这个参数里面的数据跟文件格式有关：JPEG，默认是95，范围0-100，值越大，图片质量越高；PNG，0-9，数值越大，文件压缩越好，占用空间小，但是耗时较大；PPM,PGM,PBM，表示文件已二进制方式存储还是文本格式存储，1 表示二进制存储，为默认值。

string filename = "yang.jpg";

Mat image = imread(filename, 1);

if (image.empty() == true){

cout << "can not load image" << endl;

return -1;

}

Mat result;

Canny(image, result, 50, 200);

imwrite("yang\_canny.jpg", result);

1. 矩阵Mat学习

在程序中，使用Mat 保存图像的数据。Mat有多种构造函数：

Mat()

Mat(int rows, int cols, int type)

Mat(Size size, int type)

Mat(int rows, int cols, int type, const Scalar &s);

Mat(Size size, int type, const Scalar &s);

Mat(const Mat& image, const Rect&rect)

Mat(const Mat& image, const Range& rowRange, const Range& colRange)

上面的构造方式使用的图像是同一个内存区域，并没有在内存中重新开辟内存空间。如果需要独立的开辟内存空间，需使用create函数，但是无法设置图像像素的初始值。

矩阵基本元素的表示方法

对于彩色图像，矩阵的元素不在是基本的数据类型，而是向量，在opencv中预定义了一些小向量：

typedef Vec<uchar,2> Vec2b; Vec3b, Vec4b

typedef Vec<short,2> Vec2s; Vec2i, Vec2f;Vec2d;

Vec3b color;

Color[0] = 255; color[1] = 0; color[2] = 0;

读写像素值：

uchar value grayim.at<uchar>(i,j);

grayim.at<uchar>(i,j) = 128;

string filename = "yang\_canny.jpg";

Mat image = imread(filename, 1);

if (image.empty() == true){

cout << "can not load image" << endl;

return -1;

}

for (int i = 0; i < image.rows/2; ++i){

for (int j = 0; j < image.cols/2; ++j){

image.at<Vec3b>(i,j) = Vec3b(0, 0, 128);

}

}

imshow("image", image);

waitKey(0);

Mat A(image, Range::all(),Range(1,3))

核心部分：Mat内存管理知识

对于Mat内部数据结构：int\* refcount 是一个指向mat数据的引用次数的指针，表明该数据快有几个被引用；这个由opencv自动维护；然后一个uchar\* data指向矩阵数据的首地址。当refcount为0 的时候，就会释放掉data的内存空间。

IplImage CvMat 与Mat的转换

void fun(IplImage\* p,)

Mat image;

IplImage iplimage = image; fun(&iplimage,)

CvMat cvimg = image;这两个数据是没有关于refcount的数据结构，因此千万不要提前释放mat数据。

Mat(const CvMat\* m, bool copyData = false)

Mat(const IplImage\* img, bool copyData = false);

Tengfei Yang

2015.08.13 Guangzhou