**EasyEdge新增vk\_video\_inference案例操作记录**

2021-12-29

EasyEdge平台Linux系统SDK开发文档-C++：

<https://ai.baidu.com/ai-doc/EASYEDGE/bk3fj86ad>

物体检测Jetson专用SDK集成文档：<https://ai.baidu.com/ai-doc/EASYDL/eka82sbuj>

这里使用的是jetson nano,jetpack4.6，SDK为安全帽检测模型

**!!!在使用SDK之前，需要从百度获得序列号来激活设备，可以从智能边缘控制台 IEC上面激活，也可以从运行demo时在命令行传入序列号，如果不传序列号，SDK会自动寻找本地激活过的序列号（license）**

**大概流程：**

一、先跑通sdk中的demo

二、再参考demo\_video\_inference,按照需求修改得到vk\_video\_inference案例

1.抛弃demo\_video\_inference中用VideoConfig类读取video\_src的方法，改为用opencv读取video\_src(视频文件、usb或csi摄像头)

2.改变命令行入参：参数1：res路径

参数2：video\_type(视频文件：1，csi摄像头：2，usb摄像头：3)

参数3：video\_src(视频文件路径或摄像头设备号)

参数4（可选）：serial\_num(如果设置为空，SDK会自动寻找本地已经激活过的license)

3.将摄像头或视频文件的每一帧都显示在窗口（原案例为保存视频）

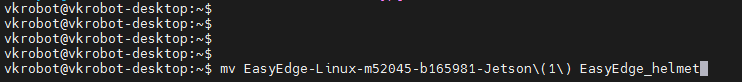
1. 增加背景，将每一帧复制到背景特定的ROI中（注意背景的刷新）
2. 在背景中将识别结果和当前时间显示出来，并调整好字体的大小格式和行距等
3. 将背景图编码成base64储存在头文件的变量内，再从头文件读取base64进行解码，最后转化成cv::Mat类型

7.将vkbg.cpp编译成动态链接库libvkbg.so

# 一、跑通sdk中的demo

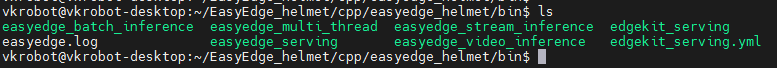
将sdk压缩包拉到jetson平台上，并解压相应jetpaack版本的sdk（这里是jetpack4.6的）

由于解压后名字会比较长，所以这里重命名为EasyEdge\_helmet



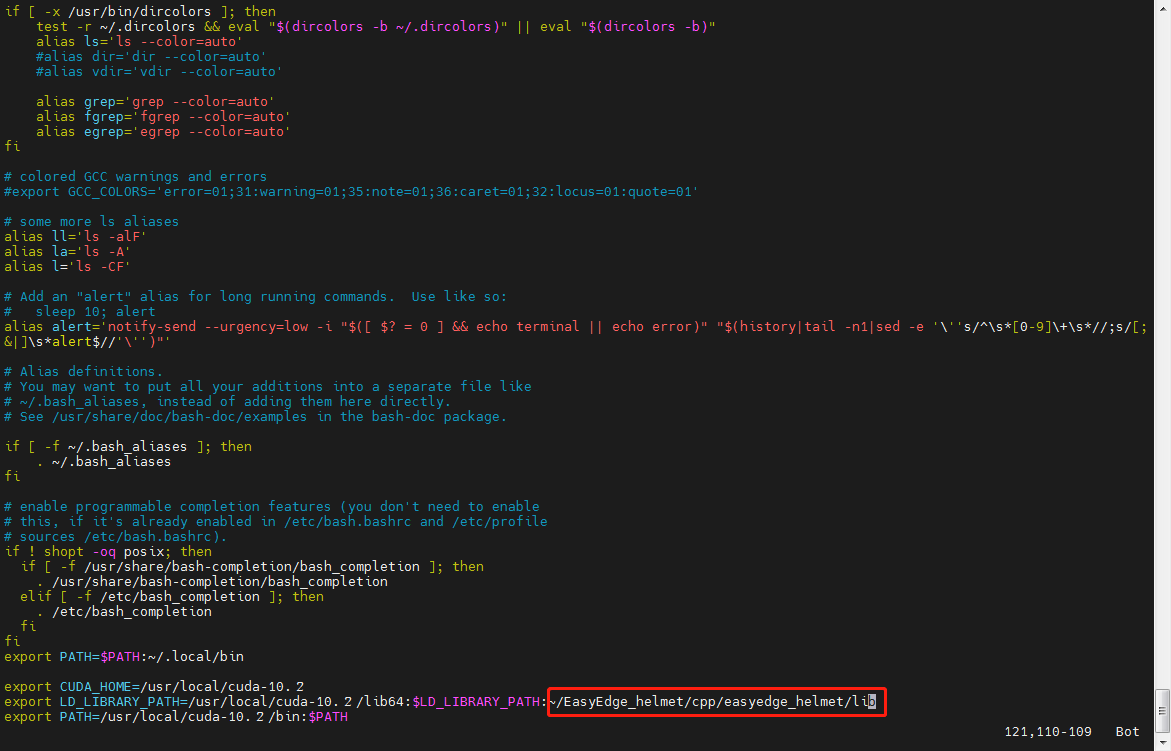
## 1.直接运行已经编译好的demo

在~/EasyEdge\_helmet/cpp/easyedge\_helmet/bin 路径（其中的easyedge\_helmet也是重命名了）下有已经编译好的demo



但在运行之前，需将lib所在路径添加为环境变量：

将sdk中的lib目录的路径添加到~/.bashrc文件下的LD\_LIBRARY\_PATH环境变量里，也可在build目录下运行sudo make install代替以上操作。



然后要source ~/.bashrc

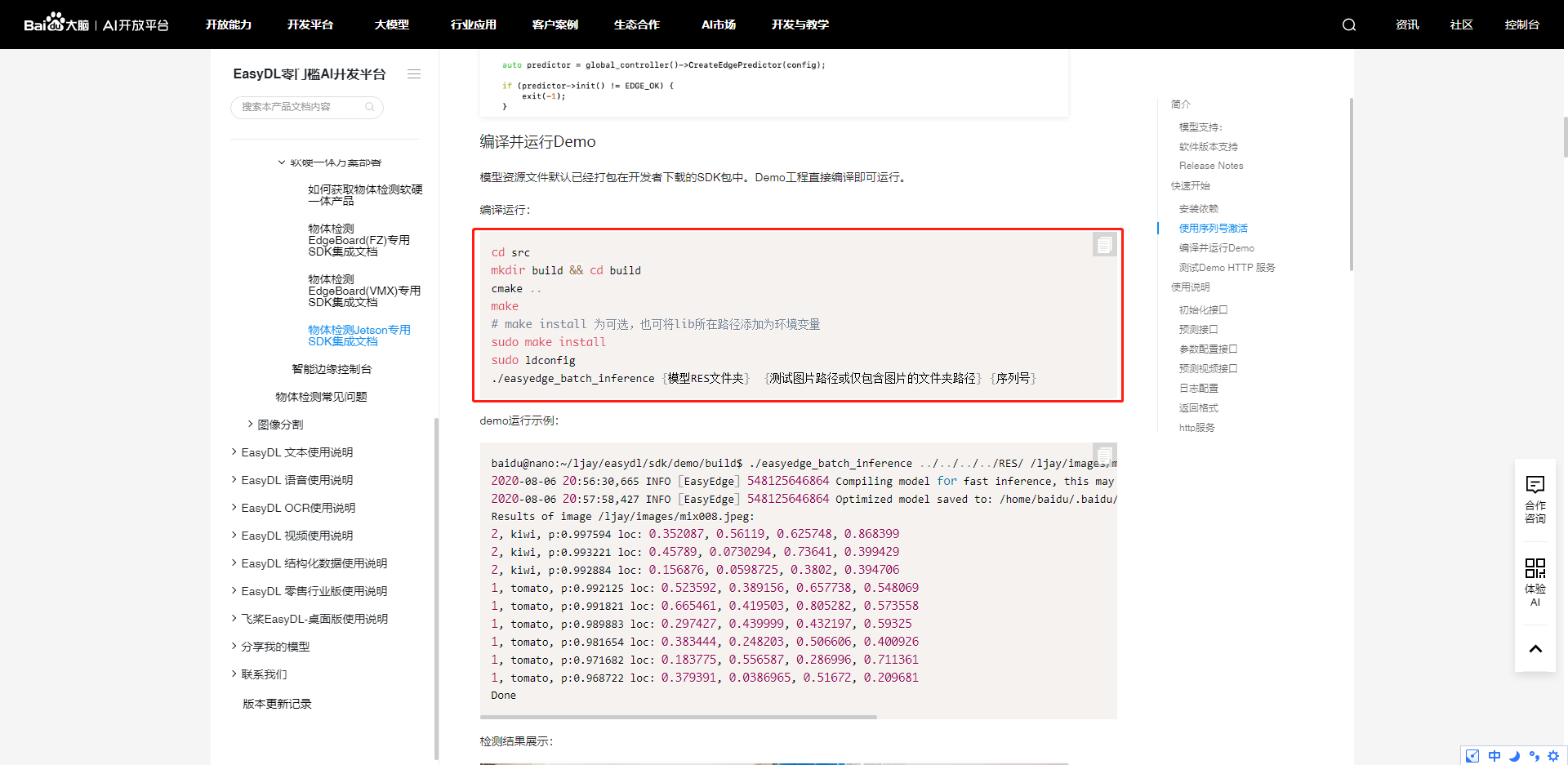
source ~/.bashrc

然后即可运行~/EasyEdge\_helmet/cpp/easyedge\_helmet/bin 路径下的demo

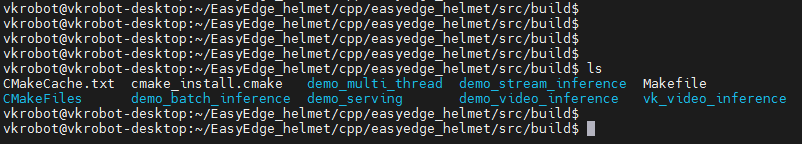
## 2.编译并运行demo

在工程的src目录下新建build目录并在build目录编译：

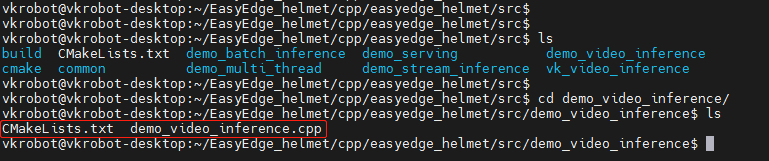
|  |
| --- |
| $ cd src  $ mkdir build && cd build  $ cmake ..  $ make  # make install 为可选，也可将lib所在路径添加为环境变量  $ sudo make install  $ sudo ldconfig  $ ./easyedge\_batch\_inference {模型RES文件夹} {测试图片路径或仅包含图片的文件夹路径} {序列号} |



编译完成后在build目录下会有各个demo的执行文件和CMakeLists.txt



至此，可以修改src目录下demo的cpp文件，再到build目录下编译，达到修改案例的效果

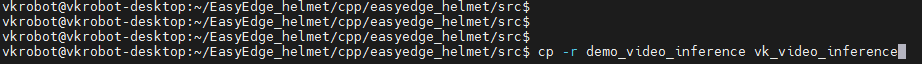


# 新增vk\_video\_inference案例

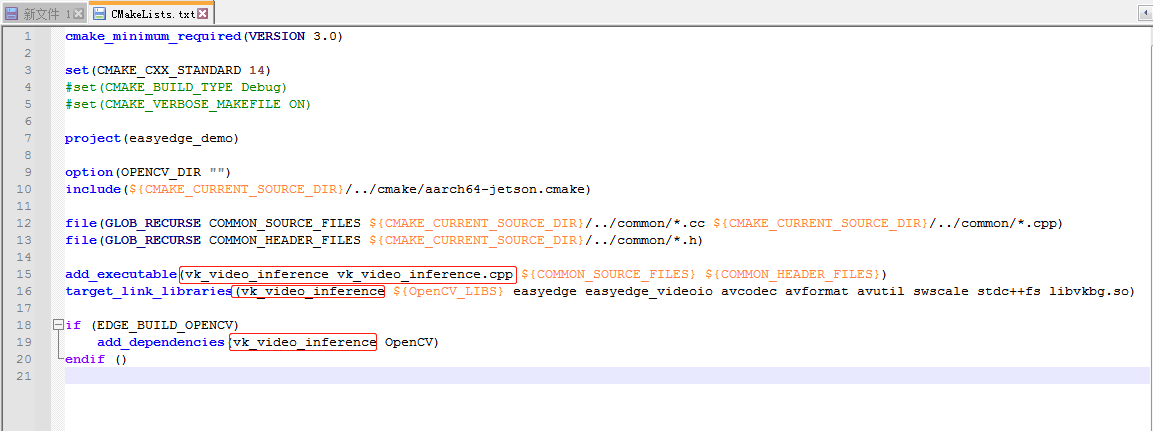
参考demo\_video\_inference,按照需求修改得到**vk\_video\_inference**案例

**准备工作：**

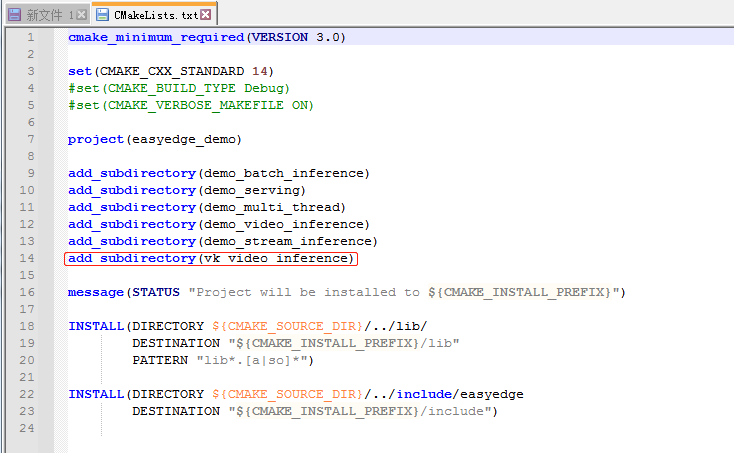
在src目录下，将demo\_video\_inference拷贝一份并重命名为vk\_video\_inference



修改src/vk\_video\_inference下的CMakeLists.txt：



修改src目录下的CMakeLists.txt：



将src目录下的build先删掉，再新建build目录，并在build目录中编译：

|  |
| --- |
| $ rm -rf build  $ mkdir build && cd build  $ cmake .. && make |

编译成功，此时，到src/build/vk\_video\_inference路径下，执行vk\_video\_inference，应该也能执行成功，效果和demo\_video\_inference案例一样

|  |
| --- |
| # 执行命令行传参和demo\_video\_inference一样  $ sudo ./vk\_video\_inference ../../../../../RES 2 2 |

接下来，我们就可以在src路径下的vk\_video\_inference做修改了

## 改用opencv读取video\_src

抛弃demo\_video\_inference中用VideoConfig类读取video\_src的方法，改为用opencv读取video\_src(视频文件、usb或csi摄像头)，详细内容和第2点一起说明

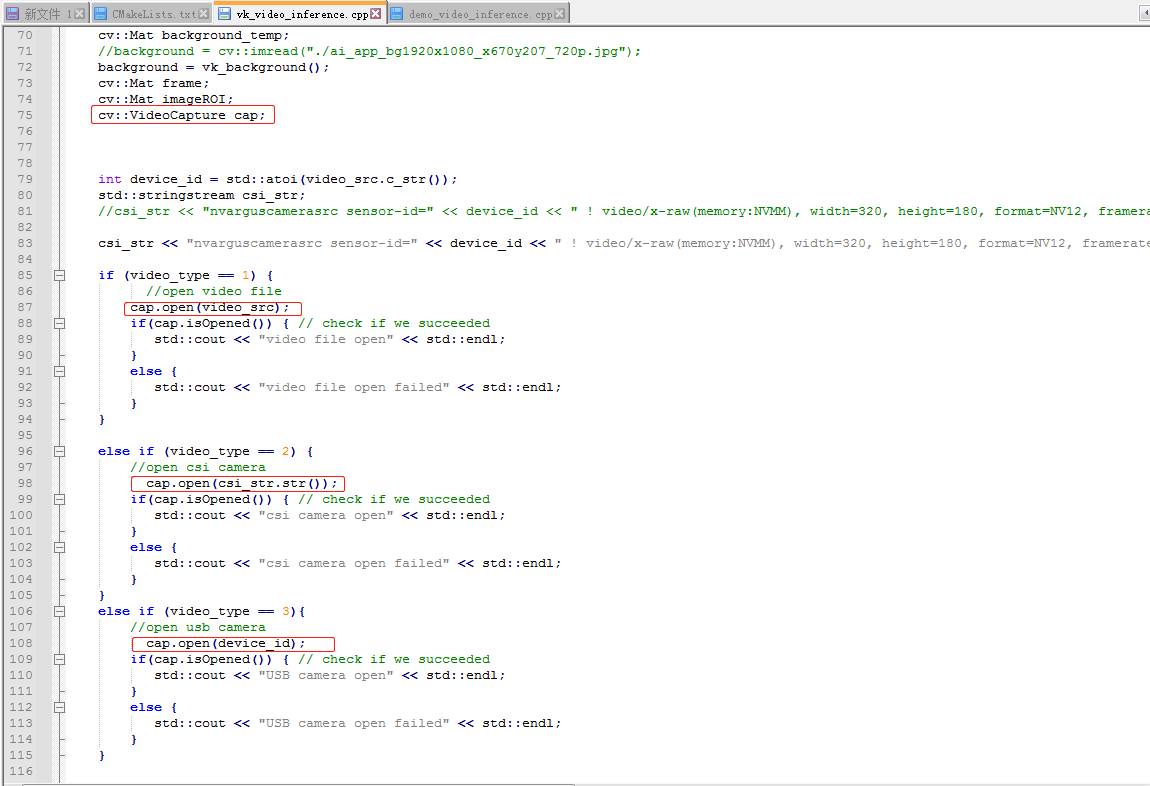
## 2.改变命令行传入参数

这两点放到一起说明

先导入使用到的 opencv 头文件：



用opencv打开摄像头或视频文件，抛弃原有用VideoConfig类的视频打开方法，并通过命令行传参选择传入视频类型（video\_type）和设备号或文件路径



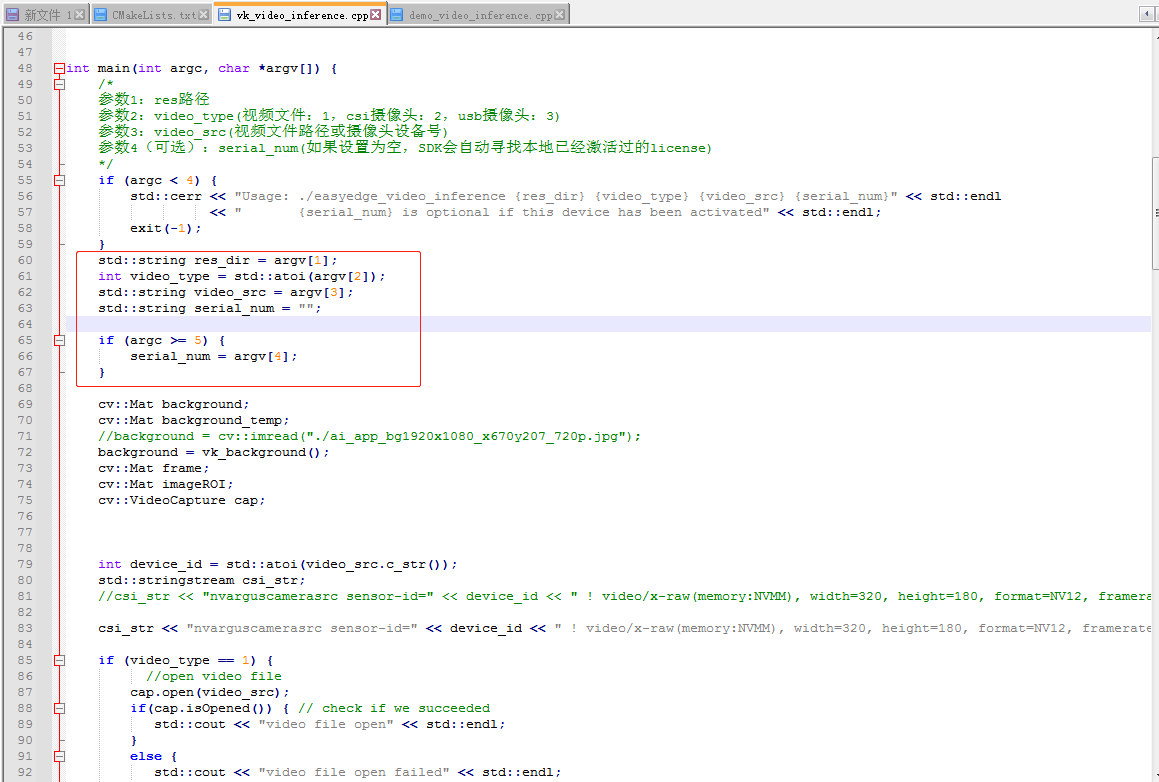
改变后的命令行传参为：

参数1：res路径

参数2：video\_type(视频文件：1，csi摄像头：2，usb摄像头：3)

参数3：video\_src(视频文件路径或摄像头设备号)

参数4（可选）：serial\_num(如果设置为空，SDK会自动寻找本地已经激活过的license)



## 将每一帧都显示在窗口

将摄像头或视频文件的每一帧都显示在窗口（原案例为保存视频）

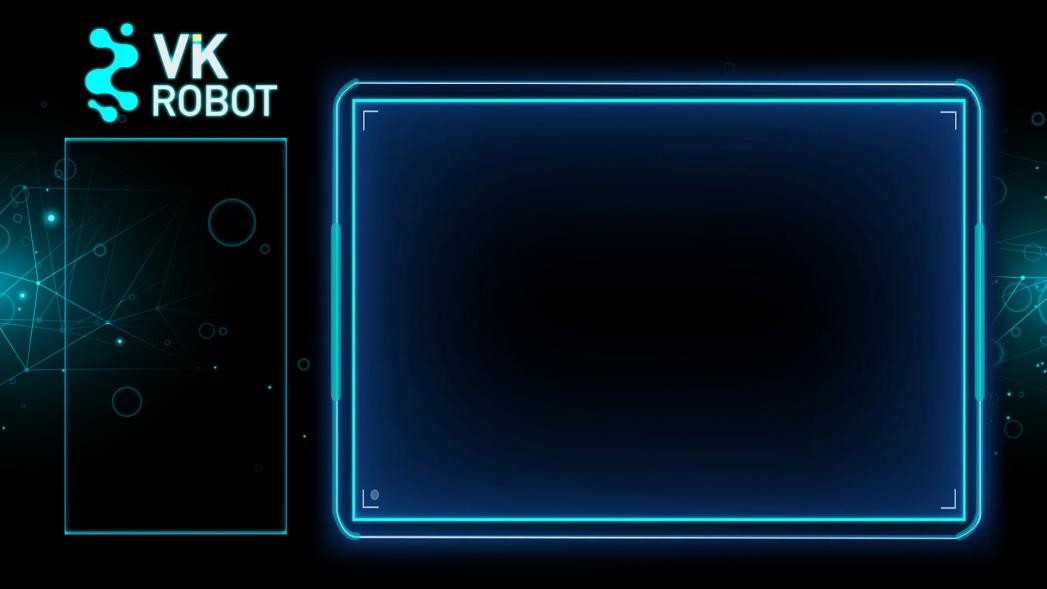
修改逻辑很简单，在代码的while中：用cap.read()每读取摄像头一帧画面，就给模型推理一帧，然后再显示一帧（可以把无关的代码注释掉）

|  |
| --- |
| cv::namedWindow("vkrobot",cv::WINDOW\_NORMAL);  while(true){  cap.read(frame);  // 垂直翻转  //cv::flip(frame,frame,0);  predictor->infer(frame, results);  // 应用识别结果到frame  render(frame, results, predictor->model\_info().kind);    cv::imshow("vkrobot",frame);  if (cv::waitKey(1) >= 0) break;    if (quit.load()) {  cap.release();  break;  }  } |

## 为显示窗口增加背景

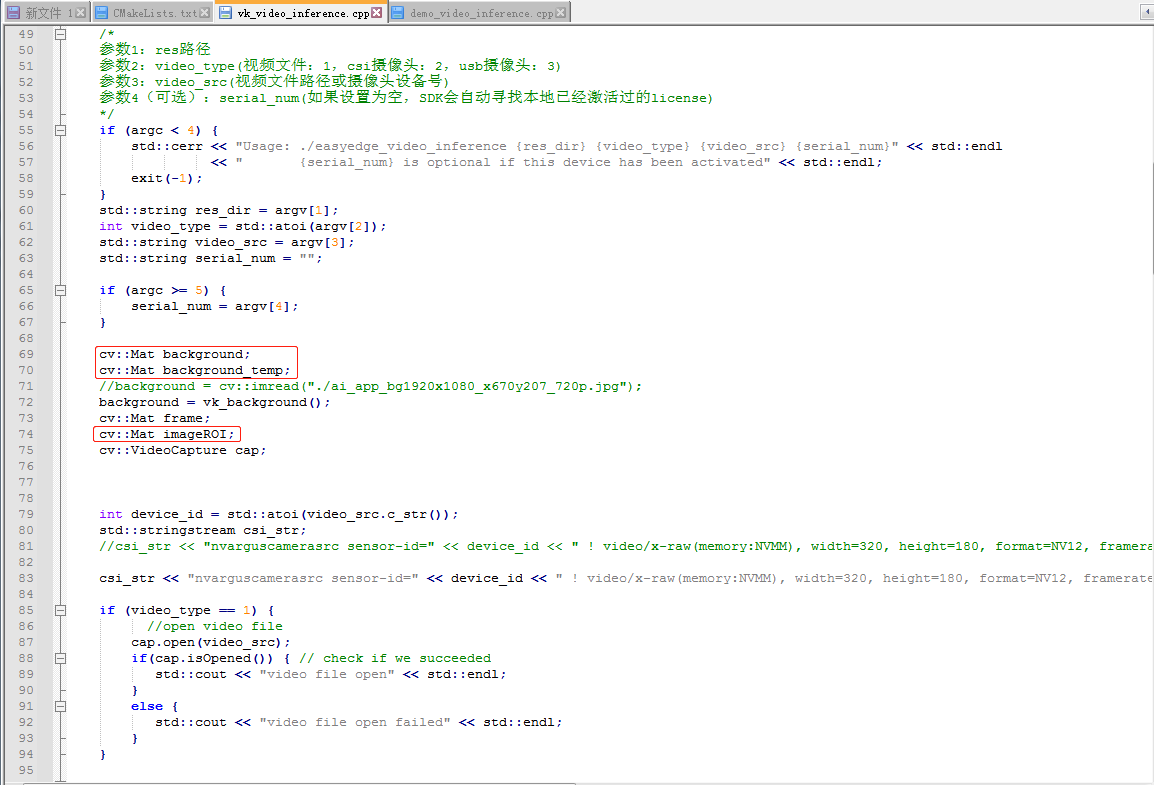
增加背景，将每一帧画面复制到背景特定的ROI中（注意背景的刷新）

背景图：右侧框显示摄像头内容，左侧框显示识别结果

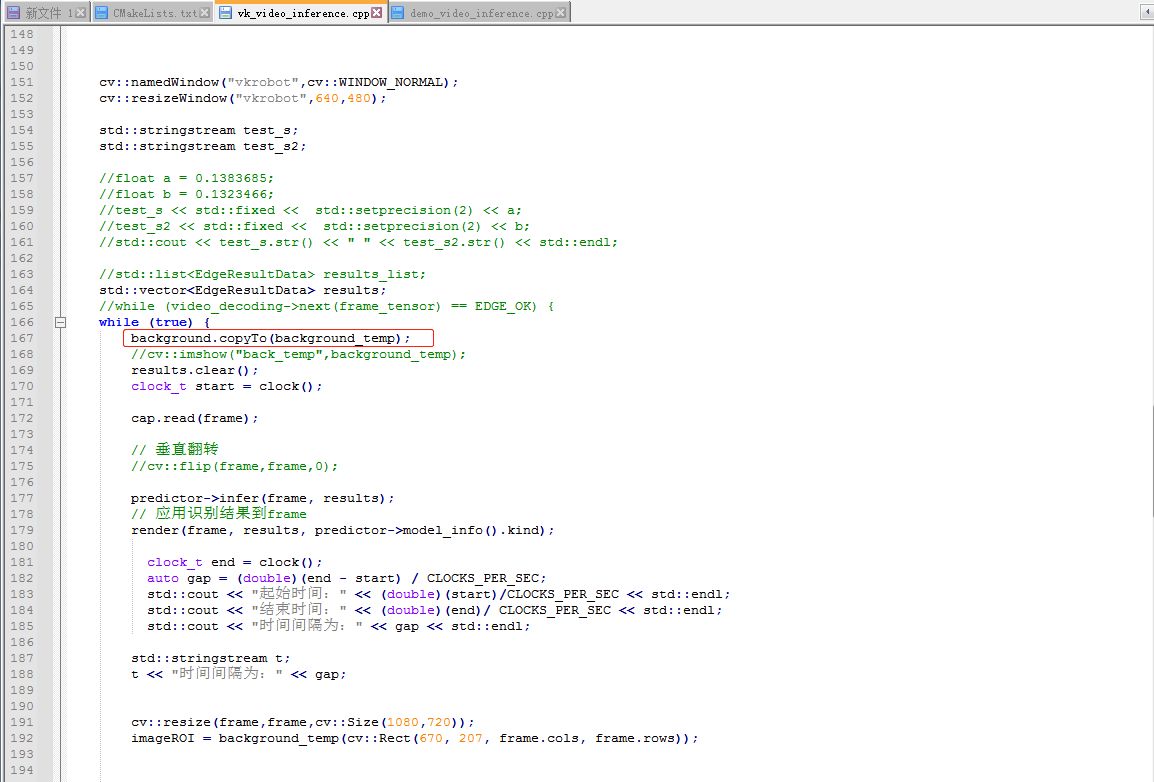


修改逻辑：

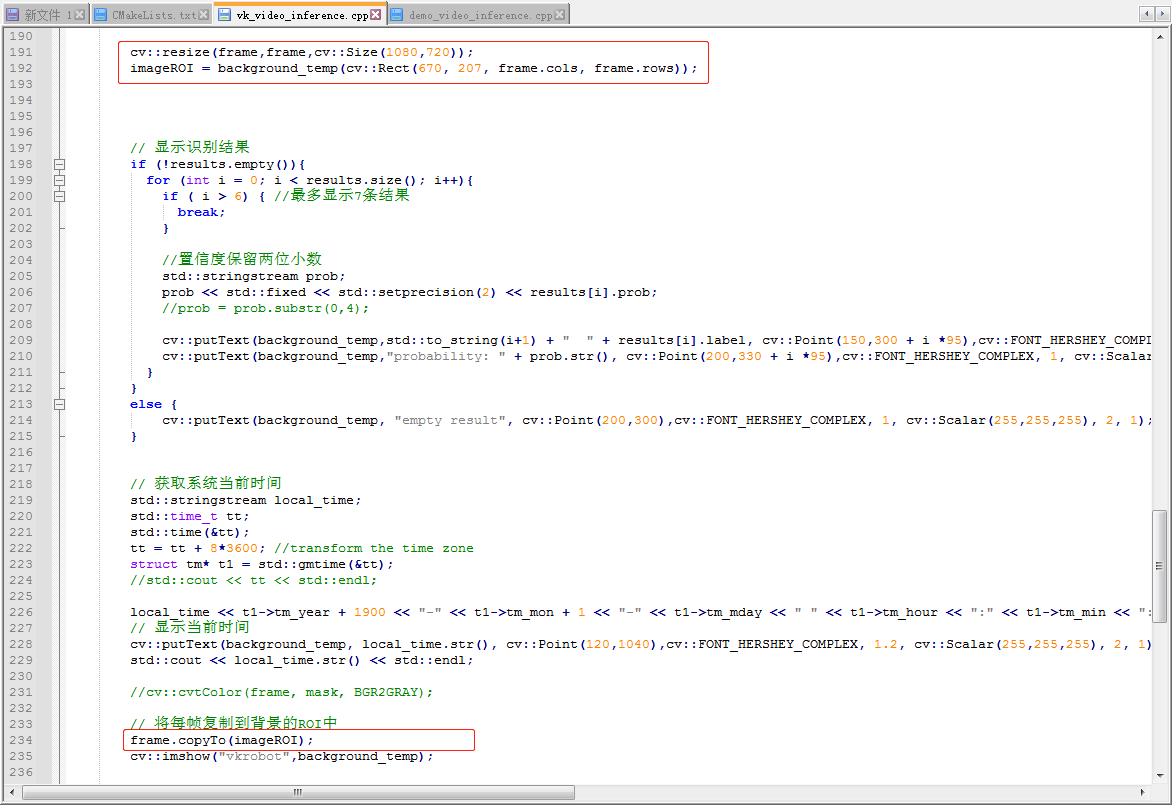
1.定义三个cv::Mat，一个是background，一个是中间变量background\_temp，一个是imageROI



2.每读完一帧，都需要将background\_temp更新为原背景background，而接下来都在background\_temp上操作，更新背景的原因是要确保读每一帧都是在原有的背景上操作的



3.用cv::resize()改变每一帧的大小为1080x720，从背景background\_temp的(670,207)位置选择感兴趣区域ROI，再将每一帧复制到选定的ROI中

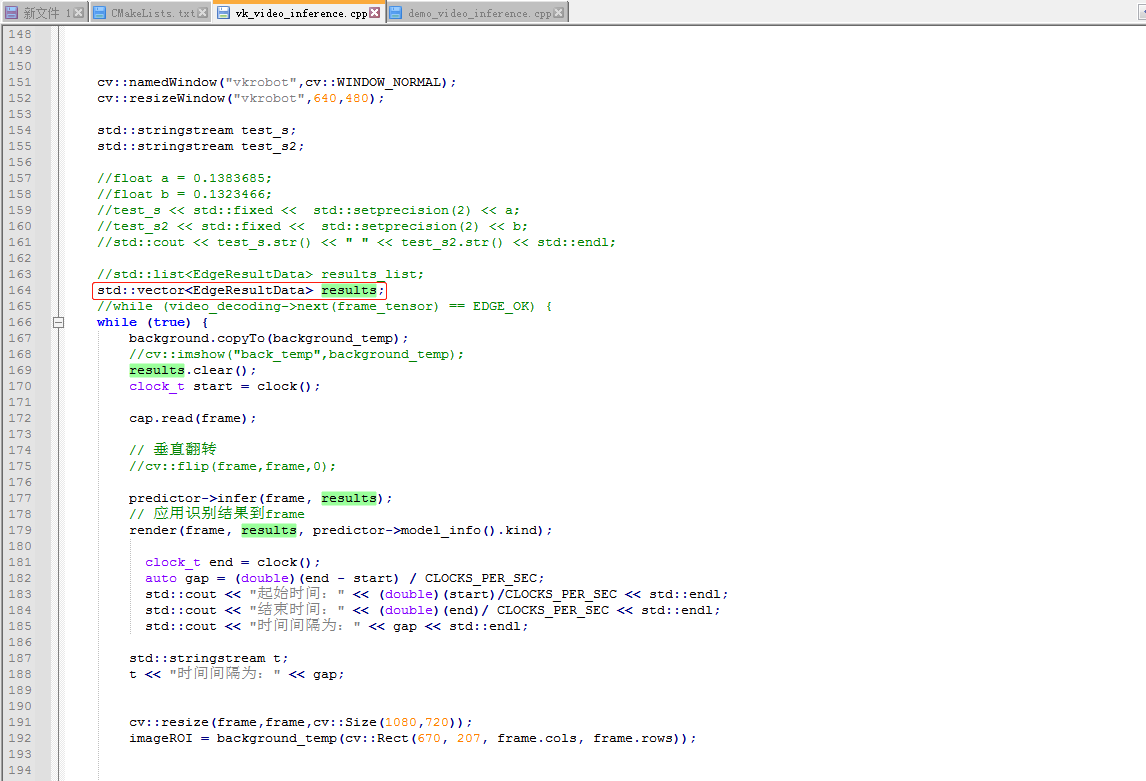


至此，可以将画面叠加到背景（效果图可看第4点）

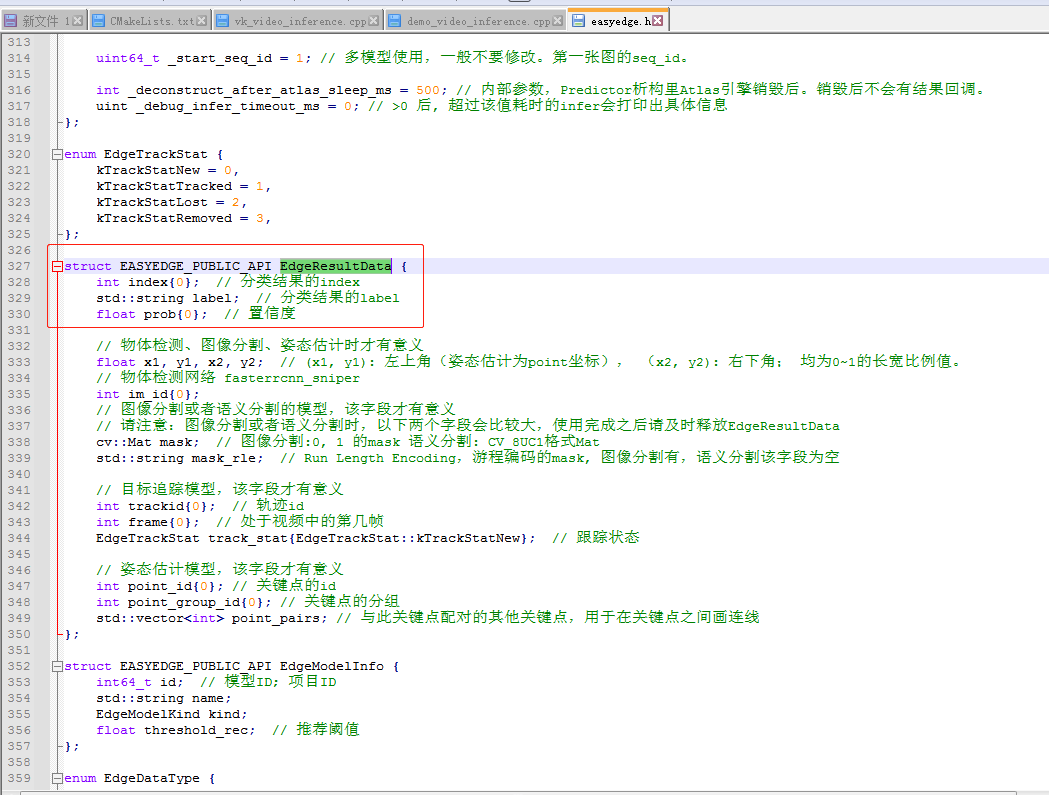
## 显示识别结果和当前时间

在背景中将识别结果和当前时间显示出来，并调整好字体的大小格式和行距等

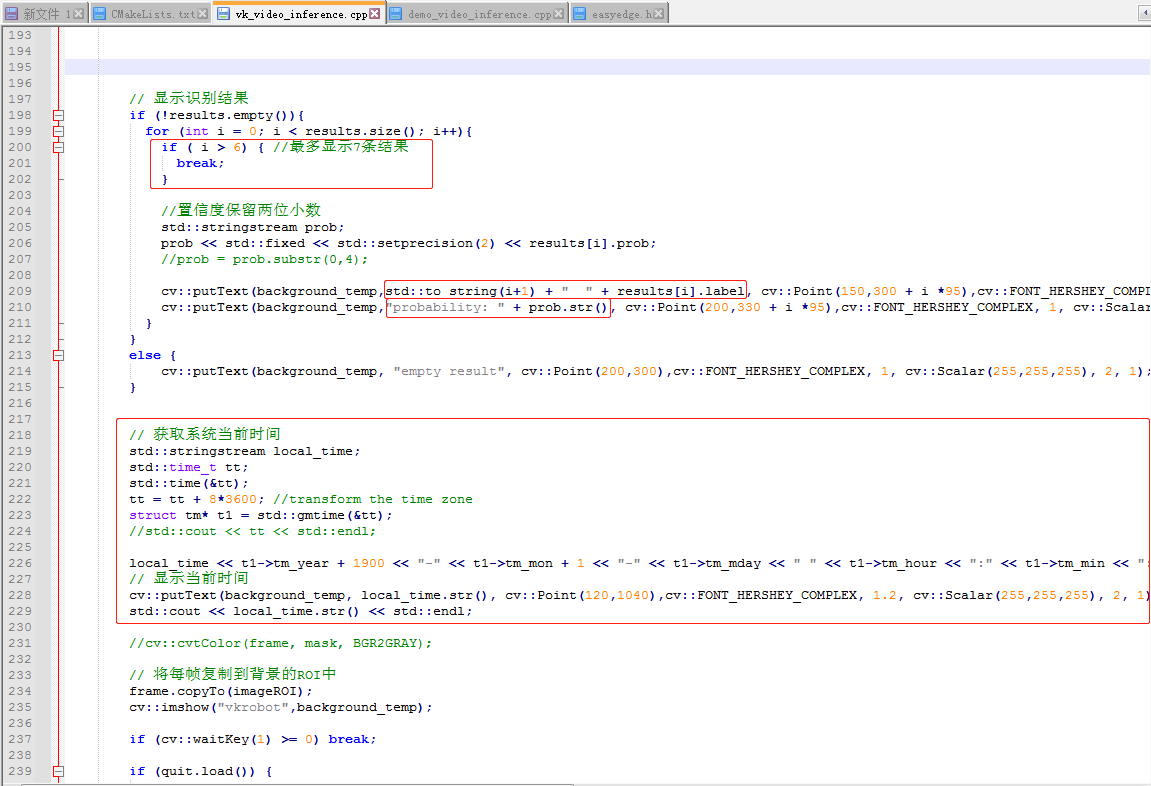
识别结果是一个EdgeResultData类型的vector，我们需要遍历vector将识别结果的label和prob（置信度）显示出来，并显示当前时间，调整字体大小格式与行距（**注意限制结果的显示数量**）



include/easyedge/easyedge.h:



vk\_video\_inference.cpp:



已将识别结果和当前时间显示出来，如下图：



## 6.将背景图以base64方式储存

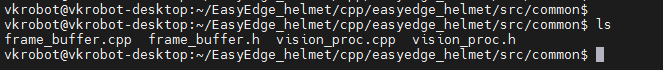
将背景图编码成base64储存在一个新建cpp文件的变量内，再进行解码，最后转化成cv::Mat类型

1. **用网上在线编码工具将背景图编码成base64，并新建vkbg.cpp,将该base64编码定义成字符串（std::string）**

并编写vk\_init()函数

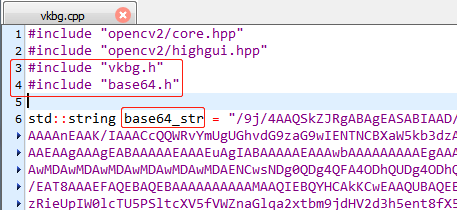
步骤分解：新建vkbg.cpp文件->定义变量存储base64码->定义函数->解码转化成cv::Mat类型,并返回cv::Mat类型图片

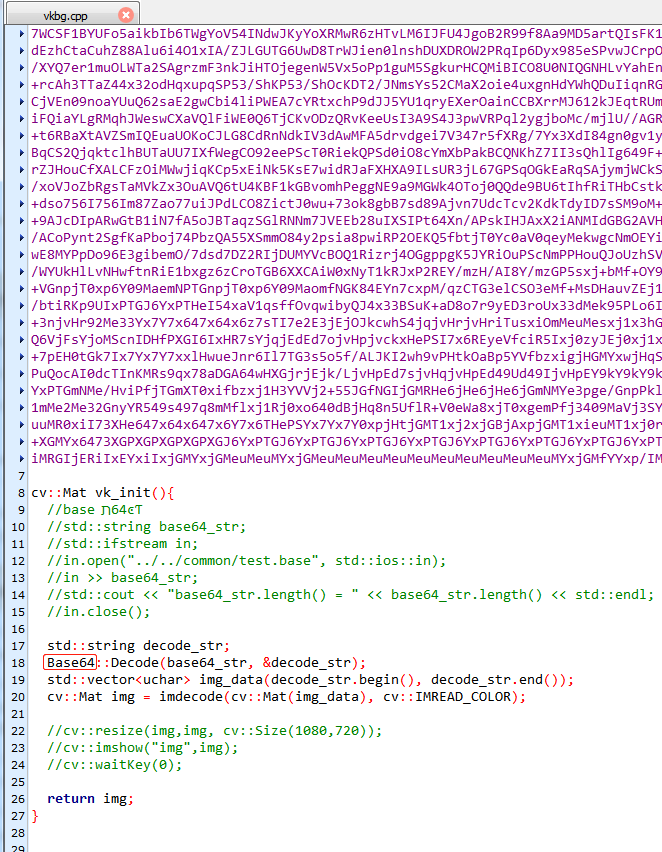
vkbg.cpp可以新建在 ~/EasyEdge\_helmet/cpp/easyedge\_helmet/src/common路径下：



参考：<https://blog.csdn.net/guo_lei_lamant/article/details/80592120>

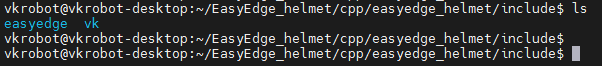
vkbg.cpp 需要导入opencv头文件外，还需要导入vkbg.h和base64.h，其中vkbg.h是下一步新建的，Base64类是在base64.h定义的（**base64.h可以在上面链接中拷贝，放到与vkbg.h同一目录下**）





1. **新建vkbg.h，并声明vk\_init()函数：**

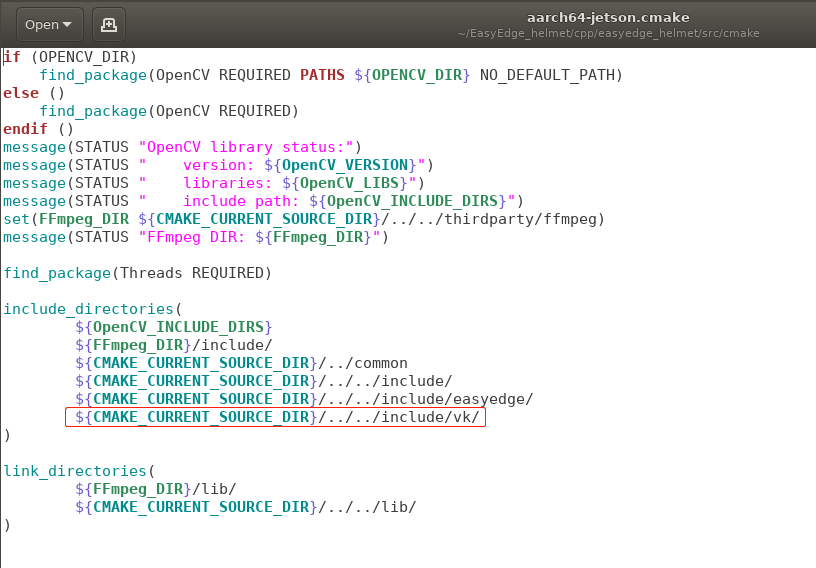
可以将vkbg.h新建在 ~/EasyEdge\_helmet/cpp/easyedge\_helmet/include路径下



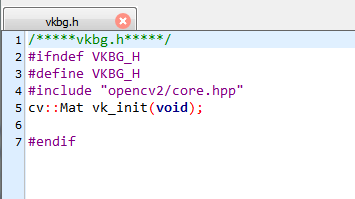
注：若想在该路径新建文件夹存放自己写的头文件，则需要修改

~/EasyEdge\_helmet/cpp/easyedge\_helmet/src/cmake路径下的aarch64-jetson.cmake文件：

将新建文件夹的路径添加到include\_directories中



vkbg.h中的内容：



在写头文件时，记得在头部要写：

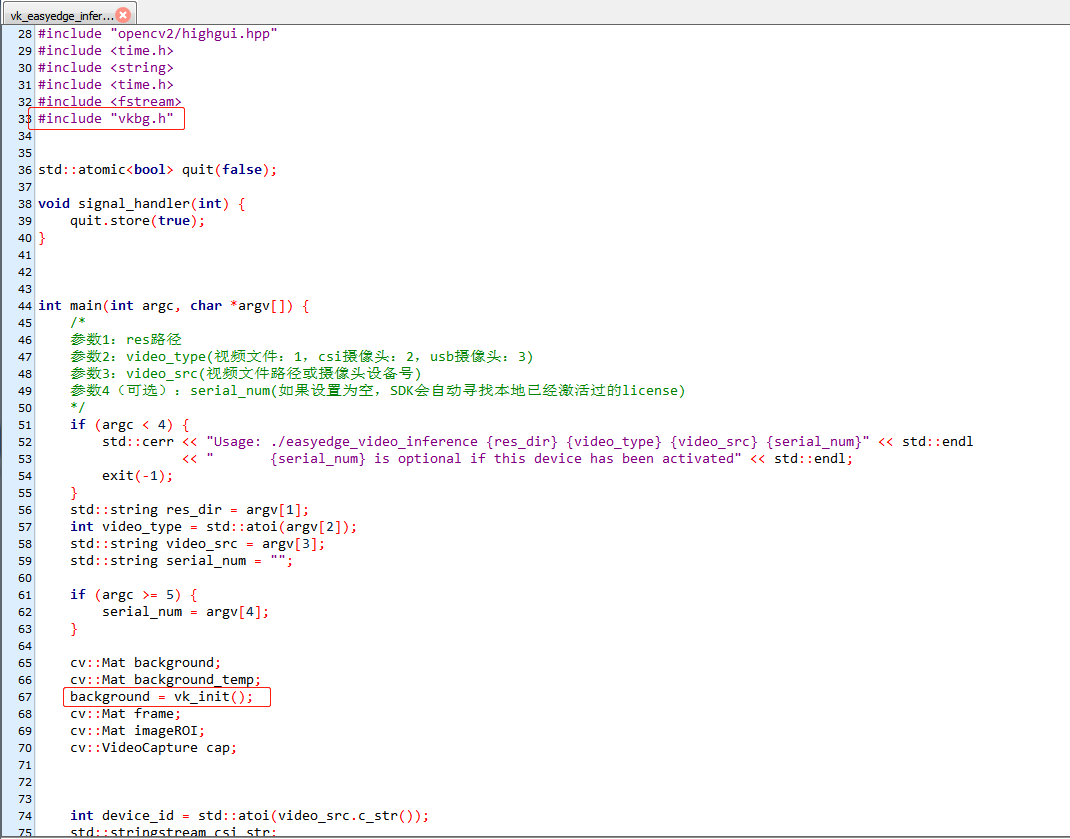
#ifndef VKBG\_H

#define VKBG\_H

在尾部要写：

#endif

1. **在vk\_video\_inference.cpp中导入vkbg.h头文件，通过vk\_init()函数得到cv::Mat类型的背景图**



修改完成后，可以开始编译，最好是删掉build目录再新建，再编译

编译成功后，vk\_video\_inference导入vkbg.h头文件，即可通过vk\_init()函数得到cv::Mat类型的背景图了

## 7.将vkbg.cpp编译成动态链接库libvkbg.so

参考：

C/C++ 动态库so的生成与调用：<https://blog.csdn.net/dzning123/article/details/80572303>

指定g++ 头文件路径的方法：<https://www.cnblogs.com/JiangLe/p/5855915.html>

cd去到vkbg.cpp所在路径：~/EasyEdge\_helmet/cpp/easyedge\_helmet/src/common

运行以下命令生成动态链接库libvkbg.so

|  |
| --- |
| $g++ vkbg.cpp -fPIC -shared -o libvkbg.so -I /usr/include/opencv4/ -I ~/EasyEdge\_helmet/cpp/easyedge\_helmet/include/vk |

bc9859f724c00f5cad8c6a3502d4730

由于vkbg.cpp中用到了opencv头文件，以及我们自己新建的vkbg.h和base64.h

所以在将vkbg.cpp编译成.so动态链接库时，需指定头文件路径： 加参数（大写的i）

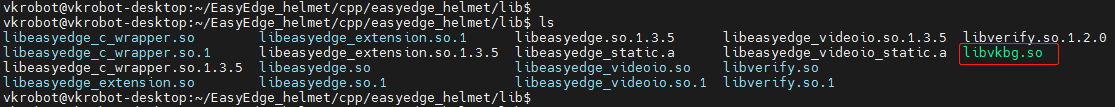
**-I 头文件路径**

-I /usr/include/opencv4/

-I ~/EasyEdge-Linux-m52045-b165981-Jetson\(1\)/cpp/easyedge\_helmet/include/vk

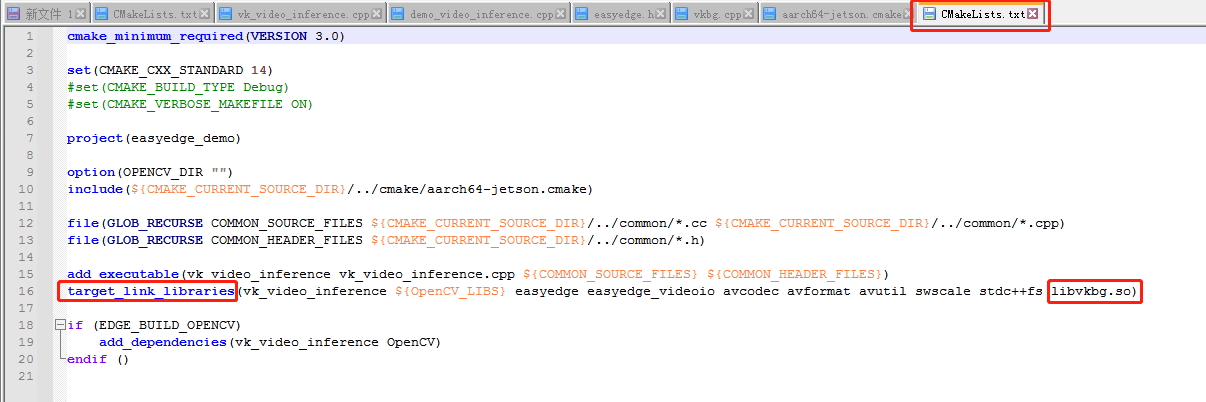
之后将会得到libvkbg.so文件，将其移动到工程的lib目录

（~/EasyEdge\_helmet/cpp/easyedge\_helmet/lib）：

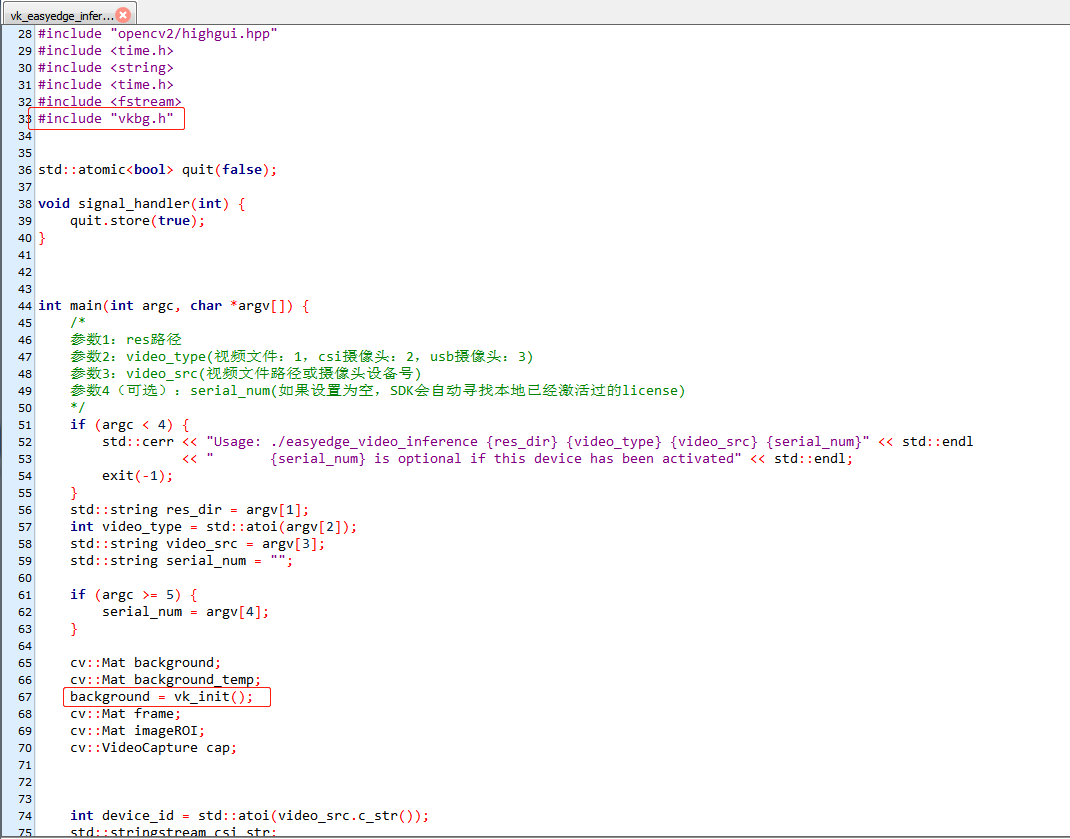


然后需要到~/EasyEdge\_helmet/cpp/easyedge\_helmet/src/vk\_video\_inference路径下修改CMakeLists.txt文件：将刚刚编译生成的libvkbg.so加到target\_link\_libraries中

截图.png



然后在vk\_video\_inference.cpp中导入头文件#include “vkbg.h” 就能使用vkbg.cpp里的函数或变量了，可以把vkbg.cpp删掉也可以了



修改完成后，可以开始编译，将src目录下的build先删掉，再新建build，并在build目录中编译：



至此，已将vkbg.cpp编译成动态链接库libvkbg.so，只需在vk\_video\_inference.cpp文件中导入vkbg.h即可使用vk\_init()函数获得背景图，不再需要vkbg.cpp与jpg格式的背景图

# 备注

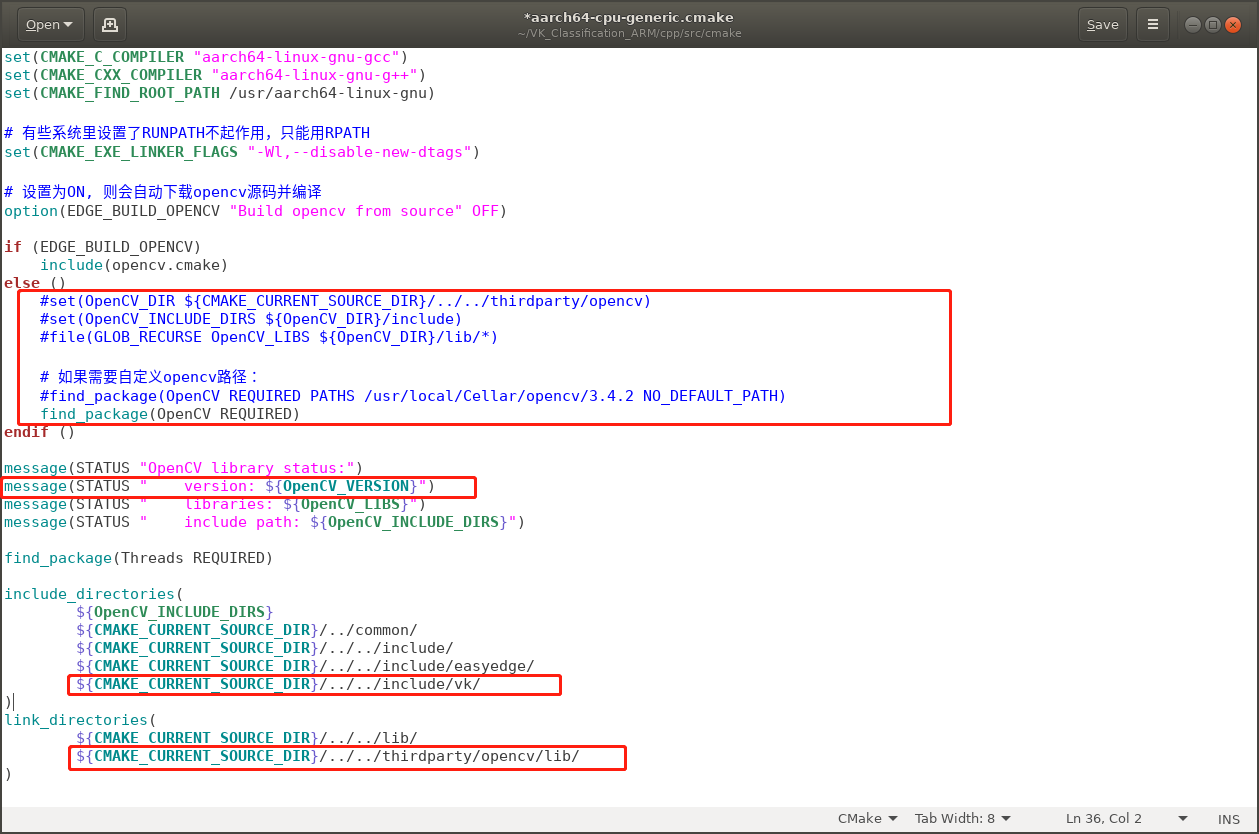
以上是改造百度EasyDL中部署方式为专项适配硬件Jetson(Nano/TX2/Xavier)-Linux的SDK的过程，当改造部署方式为通用小型设备（通用ARM）的SDK时应注意：

1. ~~保留两位小数时：std::setprecision(2)，需加头文件#include <iomanip>~~
2. ~~将vkbg.cpp编译成动态链接库时，使用的opencv头文件是SDK内~~

~~./cpp/thirdparty/opencv/include路径下的opencv2头文件(opencv3.4)~~

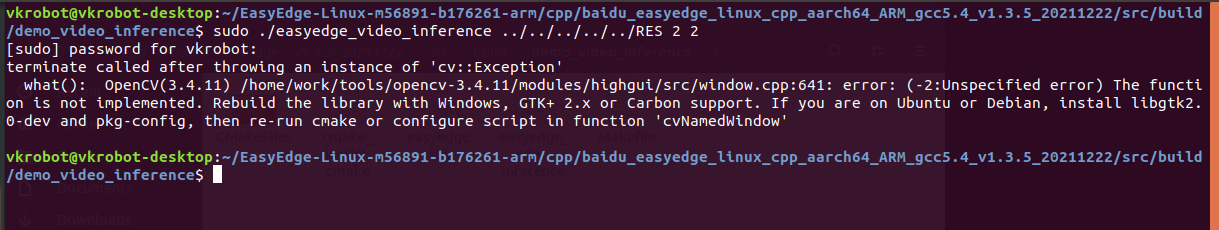
1. 在~/VK\_Classification\_ARM/cpp/src/cmake路径下修改aarch64-cpu-generic.cmake文件

将原有的opencv路径注释掉，改为自定义路径find\_package(OpenCV REQUIRED)



若不修改（使用sdk中的opencv头文件和lib）：

在代码中用opencv： cv::namedWindow(“vkrobot”,cv::WINDOW\_NORMAL);或 cv::imshow(“vkrobot”,frame); 时会报错：



即GTK2.x 要优先于opencv安装

安装虚拟键盘florence：

|  |
| --- |
| $ sudo apt install florence |