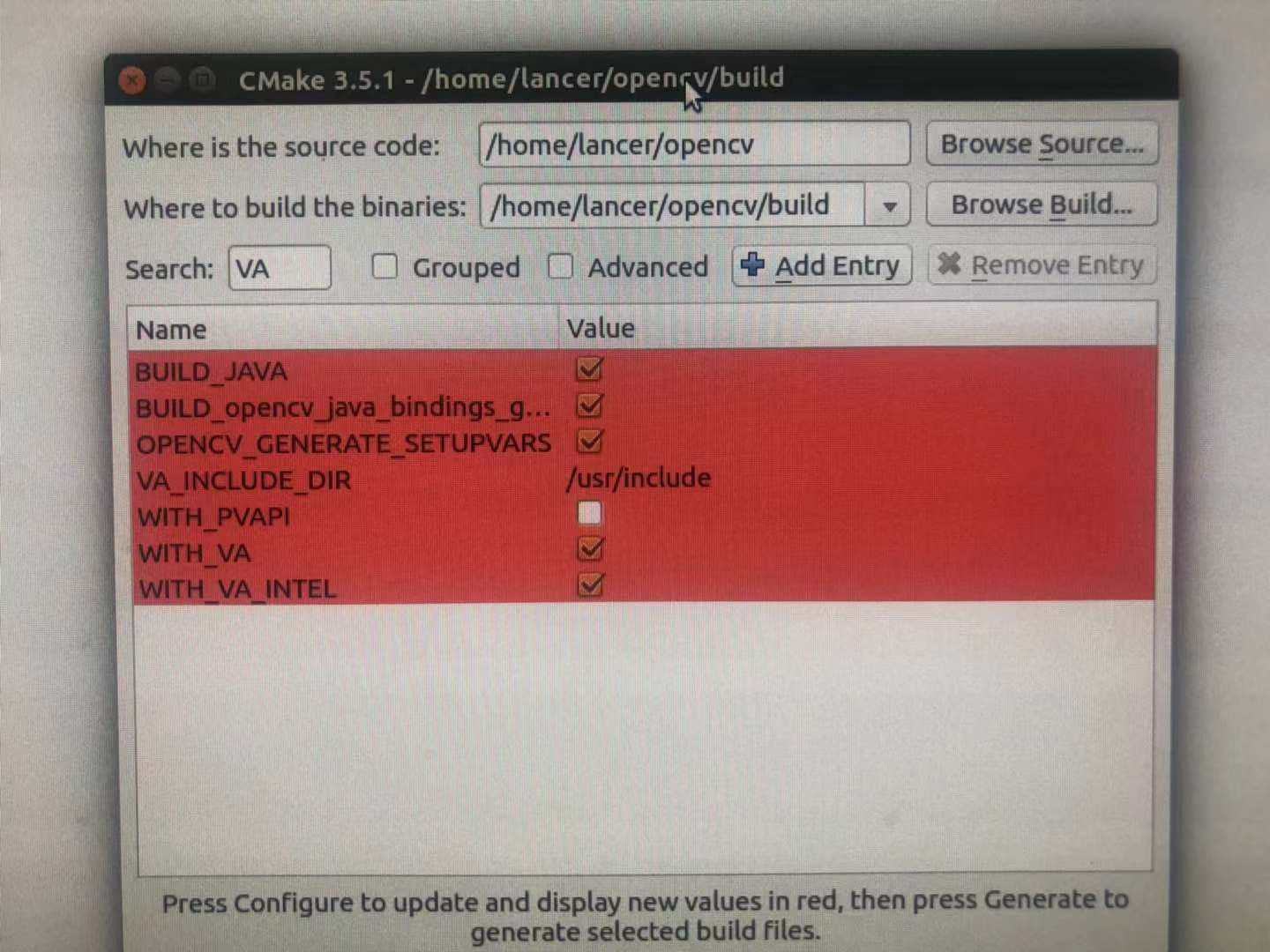
1. 安装swig

sudo apt-get install swig

1. 下载opencv，重新编译支持VA，如下图：



之后修改相应opencv的环境变量，如果安装了openvino需要注意下是否会默认搜索openvino的路径。

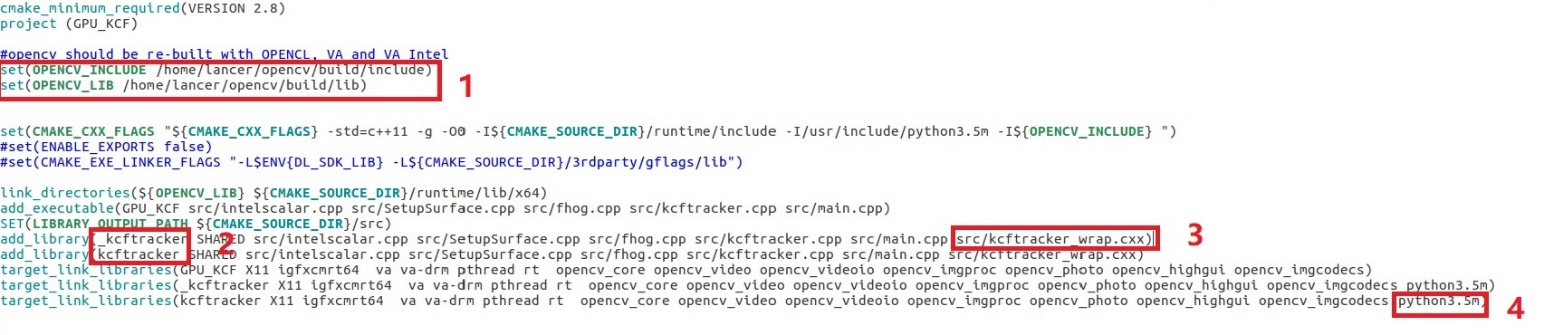
1. 修改CMakeLists相应的路径，将此路径修改为上一步骤编译好的opencv下：



1. swig -I/usr/local/include -c++ -python kcftracker.i

生成kcftracker\_wrap.cxx

1. 上步骤所生成的wrap以用来生成相应的动态库。下图中2即为动态库名称，3即为所需的wrap。最后注意动态库所需的python库如4。



1. 在build目录下make以后在src下生产lib\_kcftracker.so 手动删除此文件lib三个字母以供python调用。
2. 下载opencv-swig（本步骤可选，防止原始生成的动态库更换环境后报错，\_mat.so错误）

<https://github.com/renatoGarcia/opencv-swig.git>

进入test目录下，新建build后cmake一下。最后将生成的\_mat.so以及相应的.py拷贝致项目的src下。

1. 进入项目src下（本步骤可选，防止原始生成的动态库更换环境后报错，\_my\_rect.so错误）

swig -I/usr/local/include -c++ -python my\_rect.i

g++ -shared -fpic my\_rect\_wrap.cxx $(pkg-config --cflags --libs python3) $(pkg-config --libs opencv) -o \_my\_rect.so

1. 最后运行kcf\_test.py脚本测试即可。

## Tips

1. 为方便python端转换对应的数据类型 本代码中kcftracker里面的几个cv::umat类型替换为cv::mat 类型转换在函数内实现.

创建.i文件

例如本项目下的kcftracker.i：

%module kcftracker //名称

%include "opencv.i" //需要include的其他.i

%{

#include "tracker.h" //直接在C++代码中原样写入

#include "kcftracker.hpp"

%}

%include "tracker.h"

%include "kcftracker.hpp"

之后运行：

swig -I/usr/local/include -c++ -python kcftracker.i

本项目的rect数据类型也为自己实现的转换类型，自己写头文件，然后进行转换，可做参考：

swig -I/usr/local/include -c++ -python my\_rect.i

g++ -shared -fpic my\_rect\_wrap.cxx $(pkg-config --cflags --libs python3) $(pkg-config --libs opencv) -o \_my\_rect.so

最后可以在.py中实现import

## Requirements

项目所需：

Intel Media Driver - <https://github.com/intel/media-driver.git>

OpenCV with Intel\_VA Enable

Intel C-for-Media(MDF) - <https://01.org/c-for-media-development-package/downloads>

最后运行的python脚本所需：

pip3 install matplotlib

sudo apt-get install python3-tk

pip3 install scikit-image

sudo apt install swig python3-dev