背景介绍

opencv的exe安装包提供了配置opencv库所需要的dll和lib文件，但是其提供的动态链接库是将整个opencv打包在一起，体量相对庞大。为了缩减体量，所以对opencv中每个所涉及的项目进行单独编译，生成动态链接库dll以及配合使用的静态链接库lib。

OpenCV的核心组件如下：

core（核心库）. Core functionality imgproc（图像处理）. Image Processing imgcodecs（图像读取和保存）. Image file reading and writing videoio（视频输入输出）. Video I/O highgui（高级GUI界面）. High-level GUI video（视频分析）. Video Analysis calib3d（相机矫正和三维重建）. Camera Calibration and 3D Reconstruction features2d（二维特征框架）. 2D Features Framework objdetect（物体检测）. Object Detection dnn（深度学习）. Deep Neural Network module ml（机器学习）. Machine Learning flann（多维空间聚类和搜索）. Clustering and Search in Multi-Dimensional Spaces photo（摄影成像）. Computational Photography stitching（图像配准）. Images stitching gapi（图形api）. Graph API

基础知识（个人理解）

通常对我们自己编写的一个程序进行编译，是指**将以高级语言编写的程序转换为以二进制代码的可执行性目标程序**，通俗来说就是将所编写的cpp文件h文件打包生成EXE可执行文件的过程（也可以选择将程序打包成静态链接库lib或者动态链接库dll，后面具体说）。编译的具体过程简单来说可以分成4个阶段：预处理—->编译—->汇编—->链接。

简单来说，对一个cpp文件，预处理时会将它对应的h文件编译进来生成一个.i文件。编译这个阶段会进行语法分析检测，无错误后会将代码翻译成汇编语言，编译器(ccl)将文本文件hello.i 翻译成文本文件hello.s, 它包含一个汇编语言程序。汇编阶段会得到机器语言，汇编器as 将hello.s 翻译成机器语言保存在hello.o 中（二进制文本形式）。最终链接阶段会将程序依赖的其他文件并入，得到可执行文件。

静态链接库会在生成EXE时，将整个lib的代码数据都复制到EXE（相当于编译生成exe之后lib就没用了，因为代码都已经copy到exe了，exe不需要再依赖库，可以独立运行），在执行起来会略快一些，但是exe会变得很大，另外如果依赖库版本更新了，那么需要重新编译生成exe。对于动态链接库，是exe在运行时再连接库，到dll中寻找需要调用的代码，因此使用动态链接库的exe需要配合dll一起才能运行，但是更容易升级依赖的库。

OPENCV4.5.5编译流程

OPENCV的编译依赖于CMAKE，关于CMAKE的介绍如下

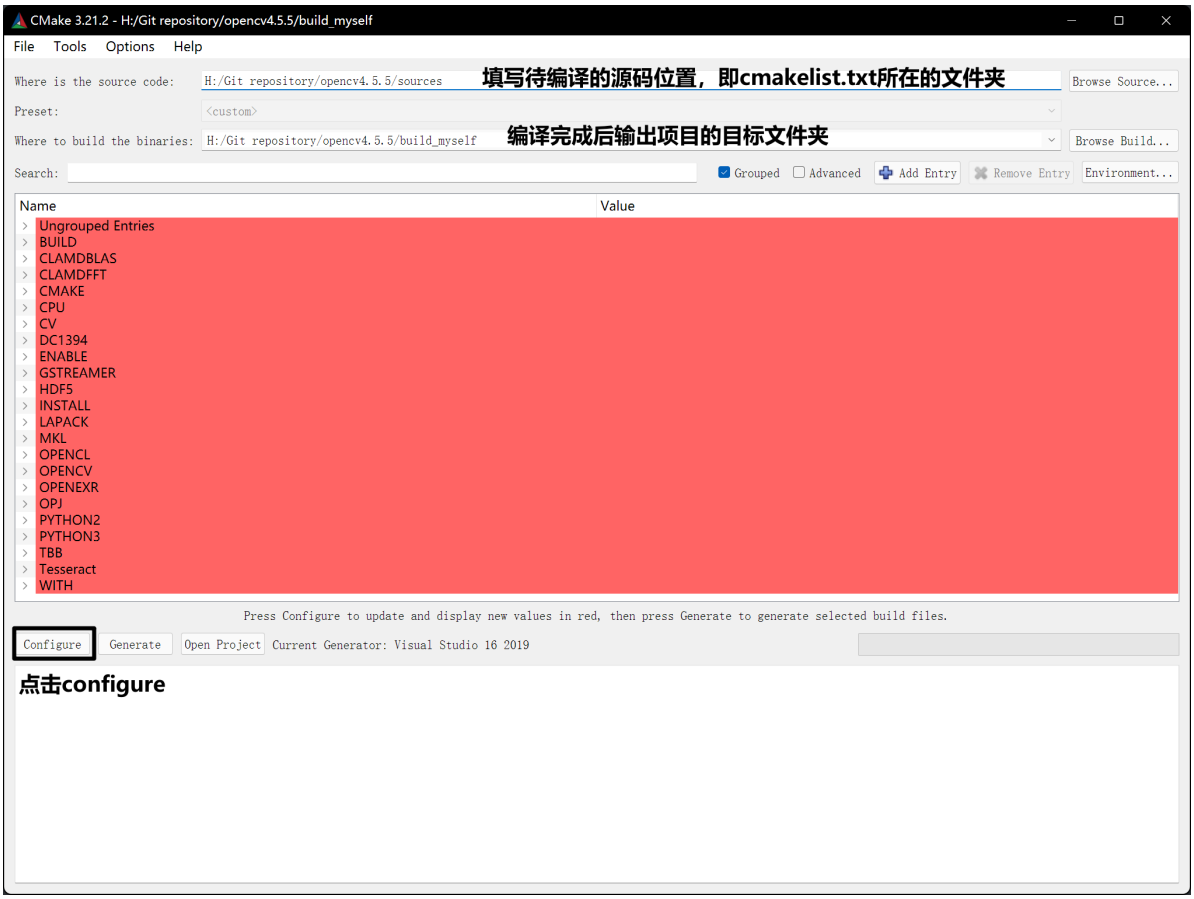
> CMake是个一个开源的跨平台自动化建构系统，用来管理软件建置的程序，并不依赖于某特定编译器，并可支持多层目录、多个应用程序与多个库。 它用配置文件控制建构过程（build process）的方式和Unix的make相似，只是CMake的配置文件取名为CMakeLists.txt。CMake并不直接建构出最终的软件，而是产生标准的建构档（如Unix的Makefile或Windows Visual C++的projects/workspaces），然后再依一般的建构方式使用。

通常在使用cmake的时候，会将配置文件卸载名为cmakelist.txt的文件中，OPENCV的源码压缩包中自带此文件，大部分的配置我们无需更改，只需要进行个别的调整。

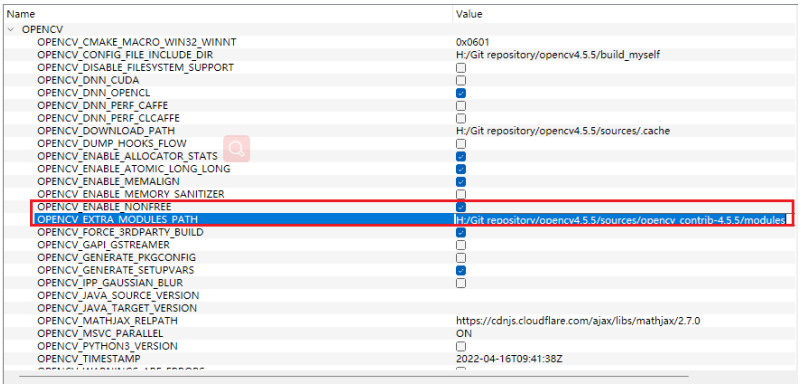
最开始当然是要下载对应的源码，OPENCV的源码分两部分：主库和contrib，后者相当于是主库的扩展包，包含了很多最新的、可能还不完善的算法，在OPENCV3.0.0之后SIFT和SURF等算法也被放入contrib，SIFT在2022年3月6日专利到期被移动到主库当中了。如果不需要使用contrib中的算法可以选择不编译这部分，下述的编译步骤中contrib的编译也被注明了是可选项。

首先推荐使用cmake-gui，当然如果你能熟练使用命令行直接用cmake也行。cmake-gui可以官方下载。

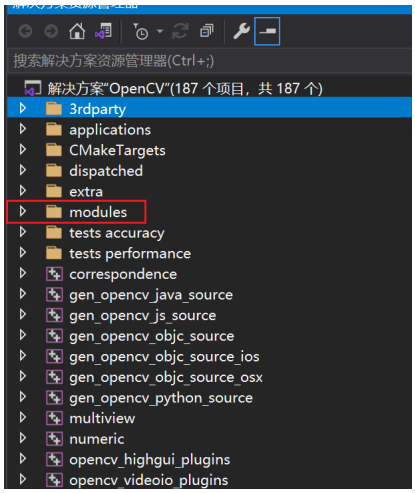
1. 按照下图填写源码位置（就是为了找cmakelist）和项目输出文件夹，点击配置，此时会读取cmakelist中的配置信息，并显示在窗口中，接下来要根据具体需要修改其中的选项。



2. 根据网络上的教程，结合自己的尝试，确定了一套在我的主机（windows11 vs2019）上可以成功编译的配置方法。
    - （可选项）PARALLEL_ENABLE_PLUGINS（这个是并行插件的开关）。OPENCV的并行计算框架按照如下的顺序提供
        - 英特尔线程构建块（第三方库，应显式启用），如TBB(Thread Building Blocks)
        - OpenMP（集成到编译器，应该被显式启用）
        - APPLE GCD（系统范围广，自动使用（仅限APPLE））
        - Windows RT并发（系统范围，自动使用（仅Windows RT））
        - Windows并发（运行时的一部分，自动使用（仅限Windows） - MSVC ++> = 10））
        - Pthreads（如果有的话）
    - （可选项）勾选OPENCV中的OPENCV_ENABLE_NONFREE，勾选后会将contrib中的项目编译进来，比如SURF、brief等模块（SIFT专利到期，已经移植到主库当中）。勾选后要写明额外模块的路径，也就是contrib的cmakelist.txt所在的文件夹。

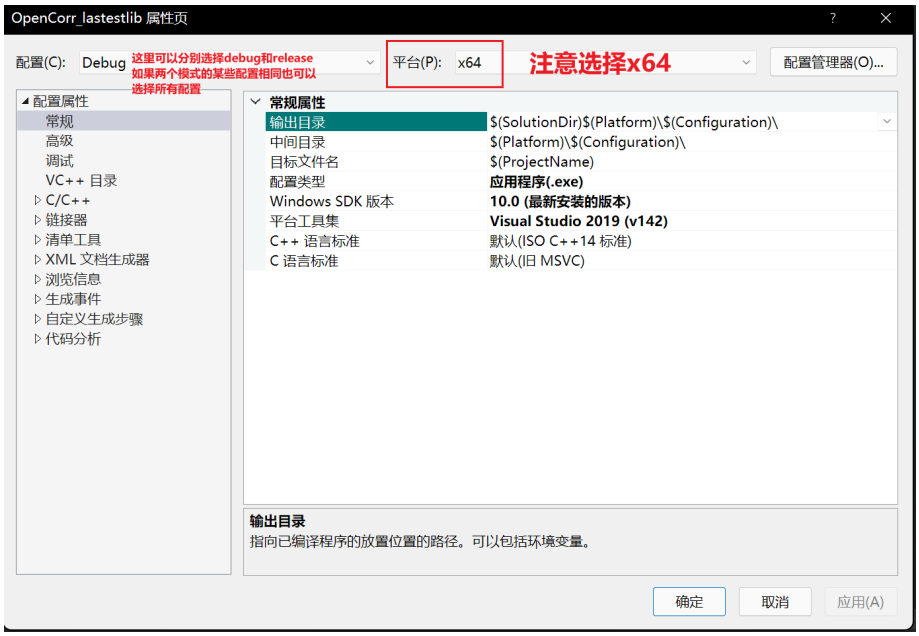


3. 点击Generate生成VS解决方案，生成之后点击Open Project，将生成的解决方案sln打开。以VS2019为例，打开后如下图所示。需要编译的内容是过滤器modules中的项目，contrib中的modules文件夹中的项目也都在这里。右键生成，分别完成debug和release模式下的编译，应该是cmakelist中有过设置，这里会同时生成exe、dll和lib。打开解决方案所在的目录，dll文件在bin文件夹当中，lib就在lib文件夹中，每个项目都对应一个dll和lib，至此OPENCV的编译结束。使用include中的头文件、dll和lib就可以正常配置opencv啦

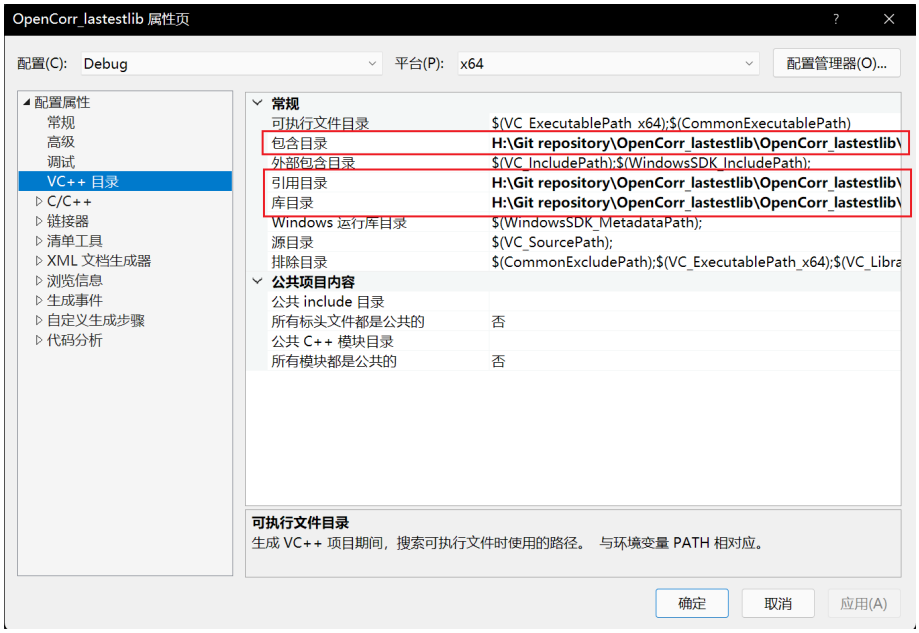


OPENCV的配置

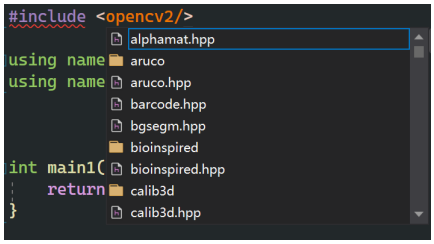
1. 右键需要配置的项目，点击属性。



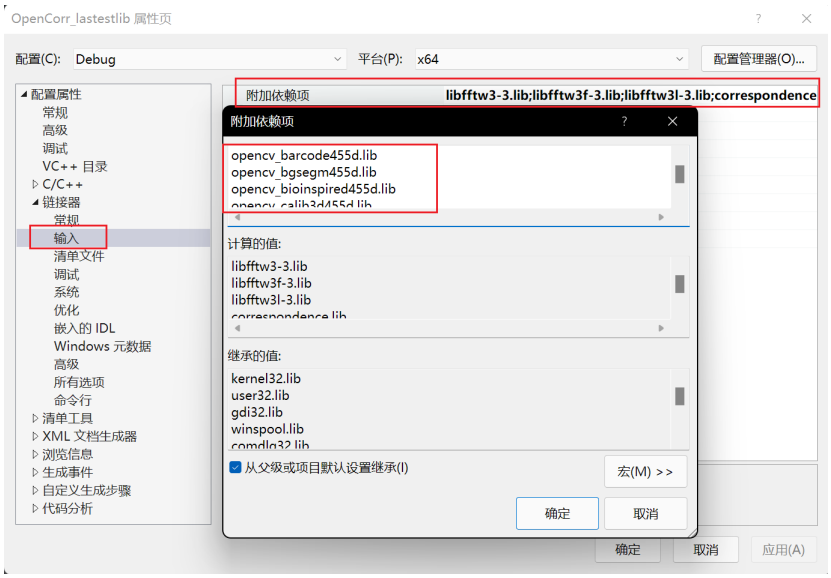
2. 需要在VC++目录和链接器中进行配置。



包含目录代表项目可以寻找头文件的目录，也就是说#include <XXX.h>这个文件所在的目录，对应到opencv中就是opencv中include文件夹（这里有opencv所有的头文件）。引用目录是可以让你在#include的时候自动弹出来那个文件夹里面的文件列表如下图。库目录则是lib文件所在的目录，在编译的时候会将这些lib中用到的代码拷贝到生成的exe当中。



接下来还需要在链接器中添加需要链接的lib名称。最后将与lib对应的dll拷贝到项目的debug或者release文件夹（exe所在的文件夹）中，代码就能正常调用OpenCV库了。



注

在opencv4.5.2之后opencv中加入了并行加速的功能，在debug模式下会有关于查找并行计算库的提示，可以通过在代码中加入下述的内容关闭信息提示。

```
#include <opencv2/core/utils/logger.hpp>
cv::utils::logging::setLogLevel(cv::utils::logging::LOG_LEVEL_SILENT);
```