# SeetaFace开源人脸识别引擎安装流程

开源的SeetaFace人脸识别引擎是由中科院计算所山世光研究员带领的人脸识别研究组研发。代码基于C++实现，且不依赖于任何第三方的库函数，开源协议为BSD-2，可供学术界和工业界免费使用。

具体介绍请参考《深度学习大讲堂知乎专栏》

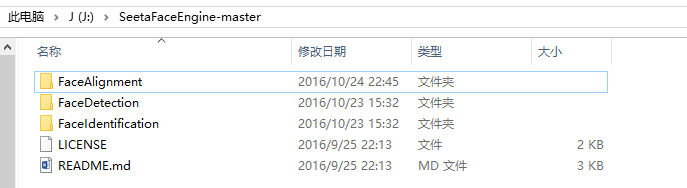
<https://zhuanlan.zhihu.com/p/22451474>

Github开源项目: <https://github.com/seetaface/SeetaFaceEngine>

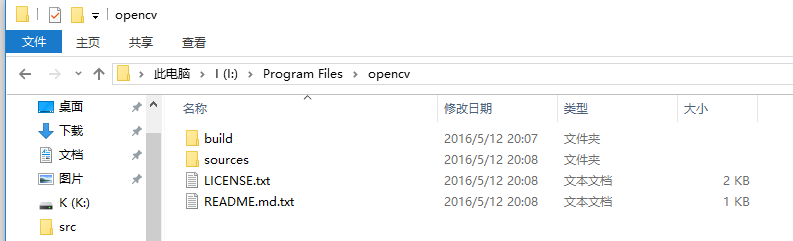
SeetaFace人脸识别引擎包括了搭建一套全自动人脸识别系统所需的三个核心模块，即：人脸检测模块（SeetaFace Detection）、面部特征点定位模块（SeetaFace Alignment）以及人脸特征提取与比对模块（SeetaFace Identification）。  
 SeetaFace项目网址为[https://github.com/seetaface/S](https://github.com/seetaface/SeetaFaceEngine)[eetaFaceEngine](https://github.com/seetaface/SeetaFaceEngine)

**附．文件的存放位置**

我的项目文件存放的文件位置：J:\SeetaFaceEngine-master



I:\Program Files\opencv



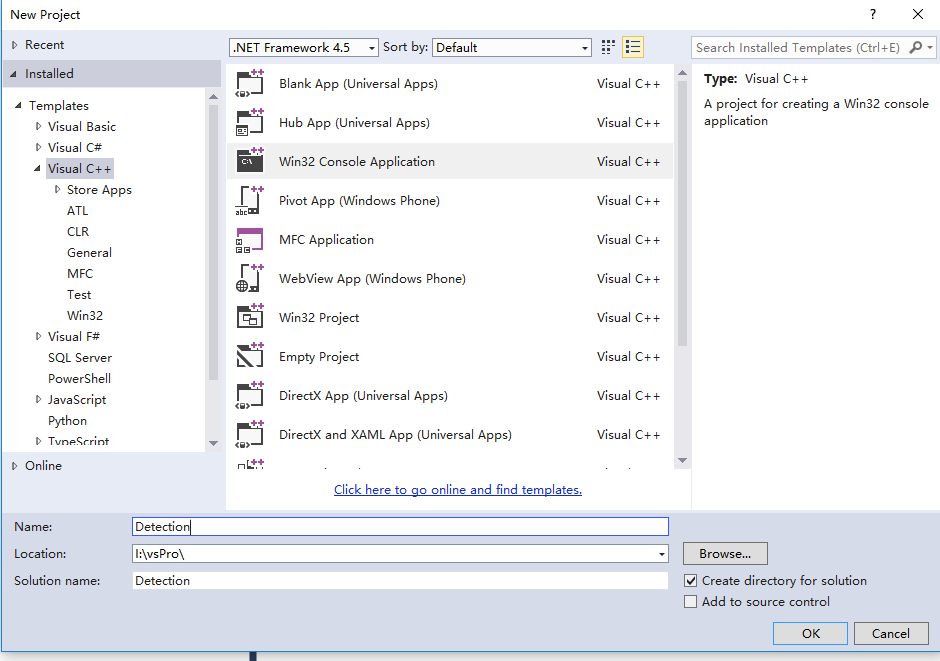
## 一、人脸检测模块SeetaFace Detection

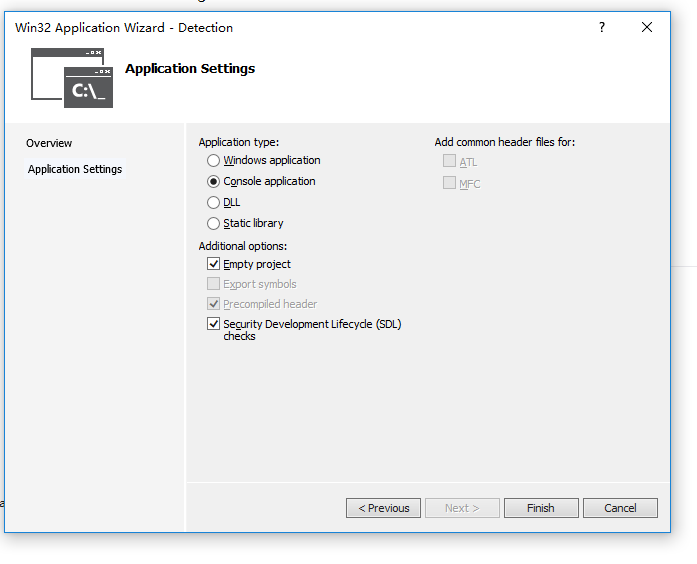
本流程配置的64位版本，32位版本的方法差不多，如果配置32位版本的话，需要在涉及设置属性配置器和平台选择时，注意选择32位版本。

这一部分的配置，总共需要添加两个New Project，一个DLL（在以下过程中命名为FaceDetection）用于载入SeetaFace Detection的SDK，一个Console application（在以下过程中命名为Detection）用于调用SeetaFace Detection的SDK。

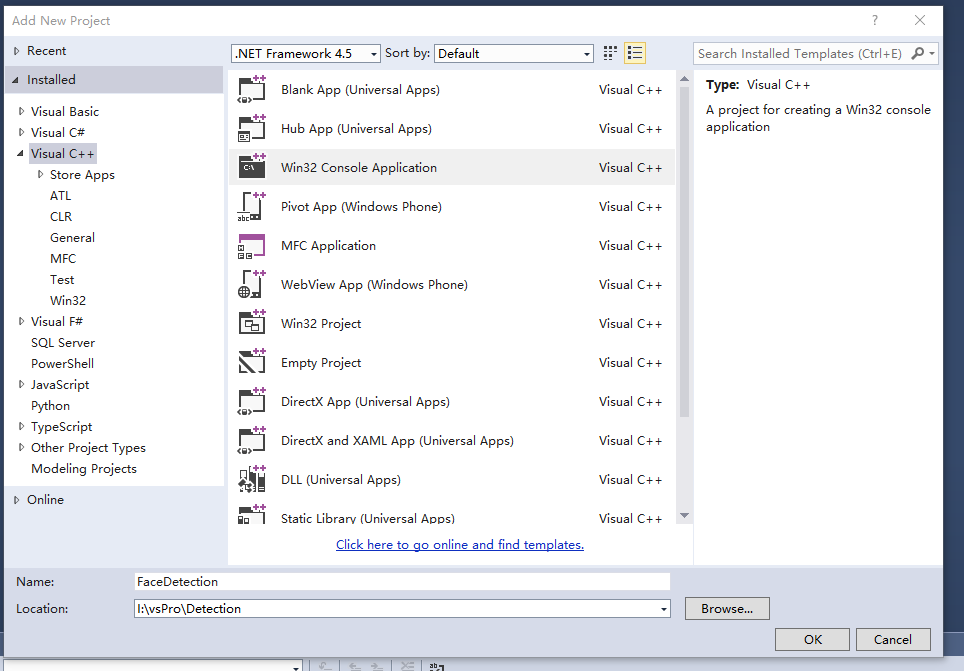
### 1新建项目工程

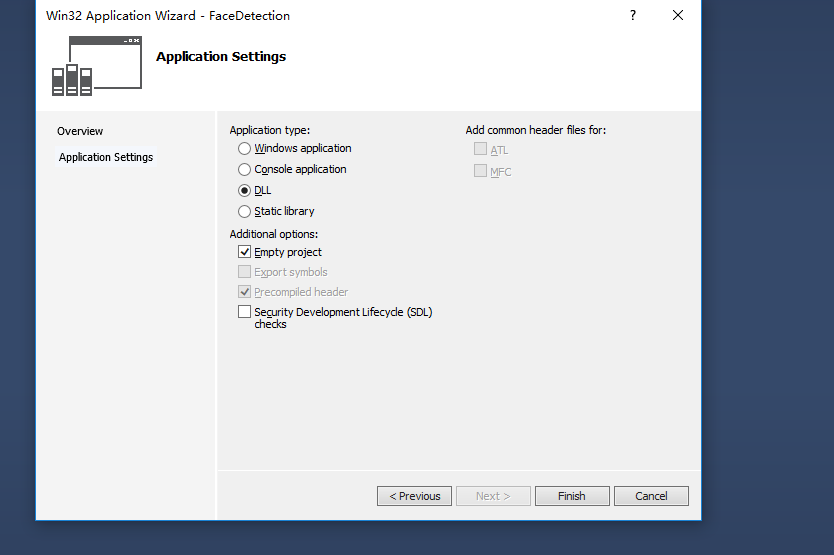
（1）新建项目，创建一个Console application项目。选择win32控制台应用程序。重命名Detection。选择存放位置。确定后选择下一步，选择Console application，空项目。





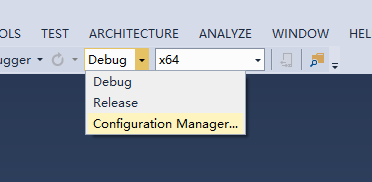
（2）创建一个DLL项目。选择win32控制台应用程序。重命名FaceDetection。选择DLL存放位置。确定后选择下一步，选择DLL，空项目。

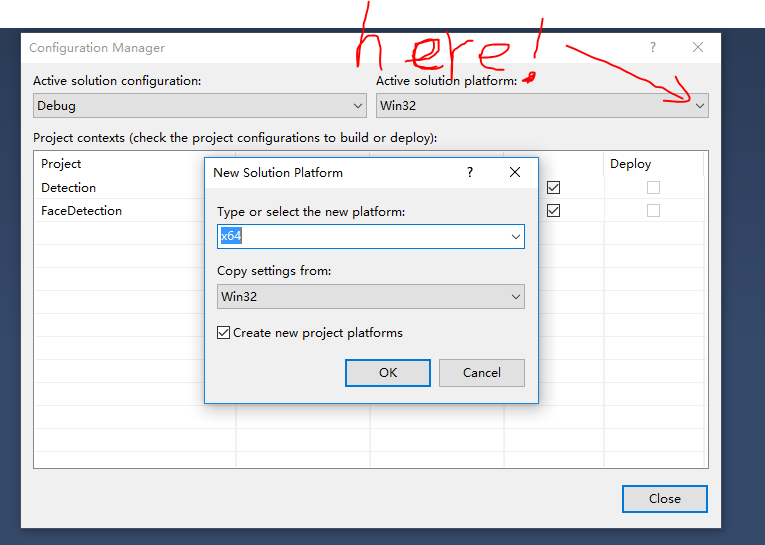




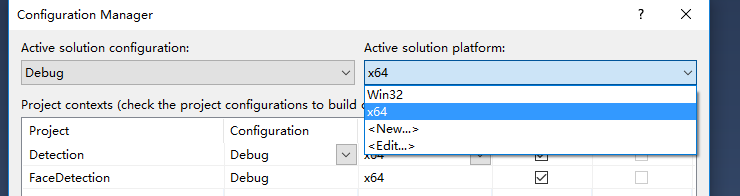
### 2平台配置（这里配的都是X64的）

（1）进入配置管理器，“平台”位置新建，新建平台选择x64。活动解决方案平台选择x64.

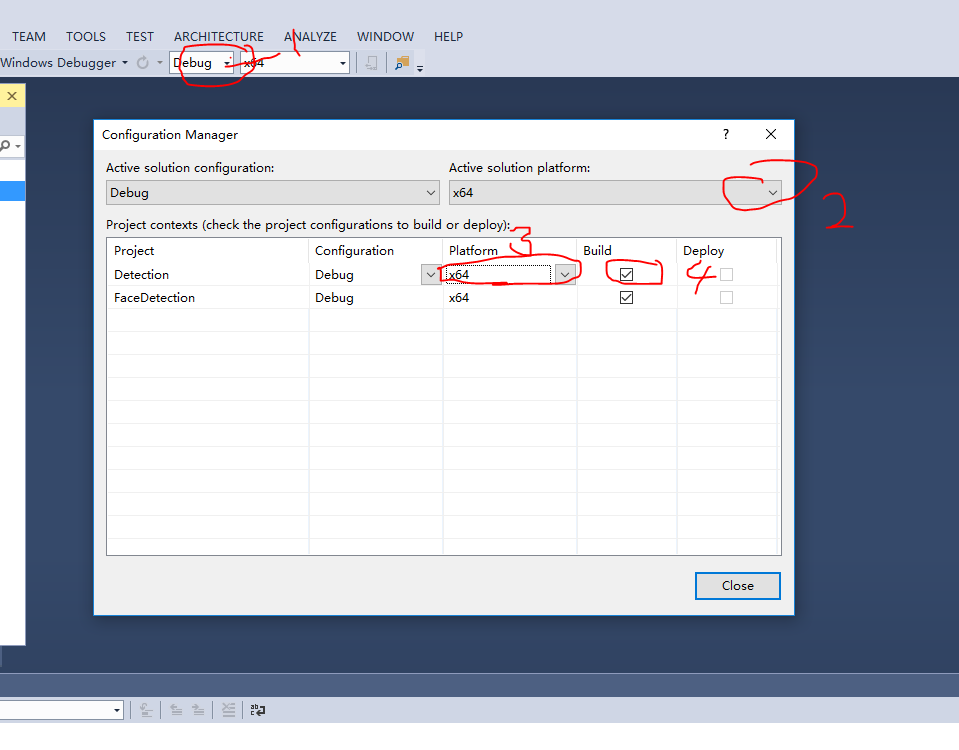




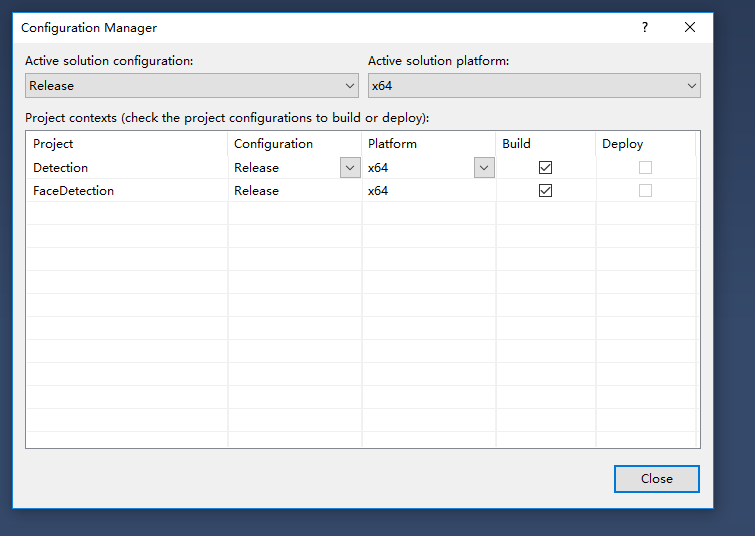
没有X64的话选择new创建一个X64



配置结果如图所示：



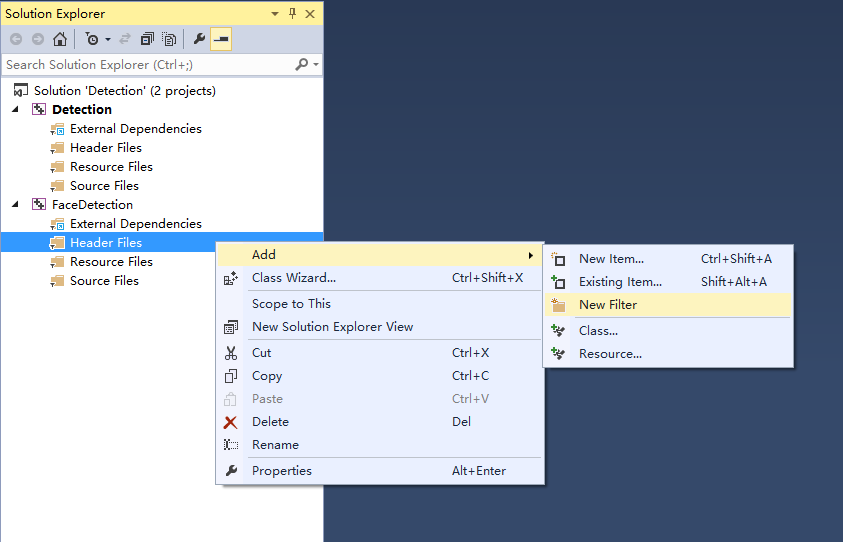
Release也要是这样的呦

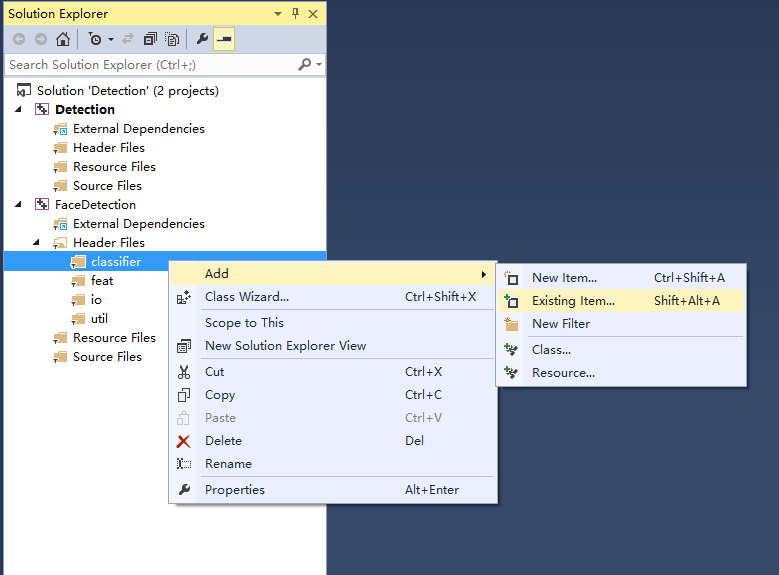


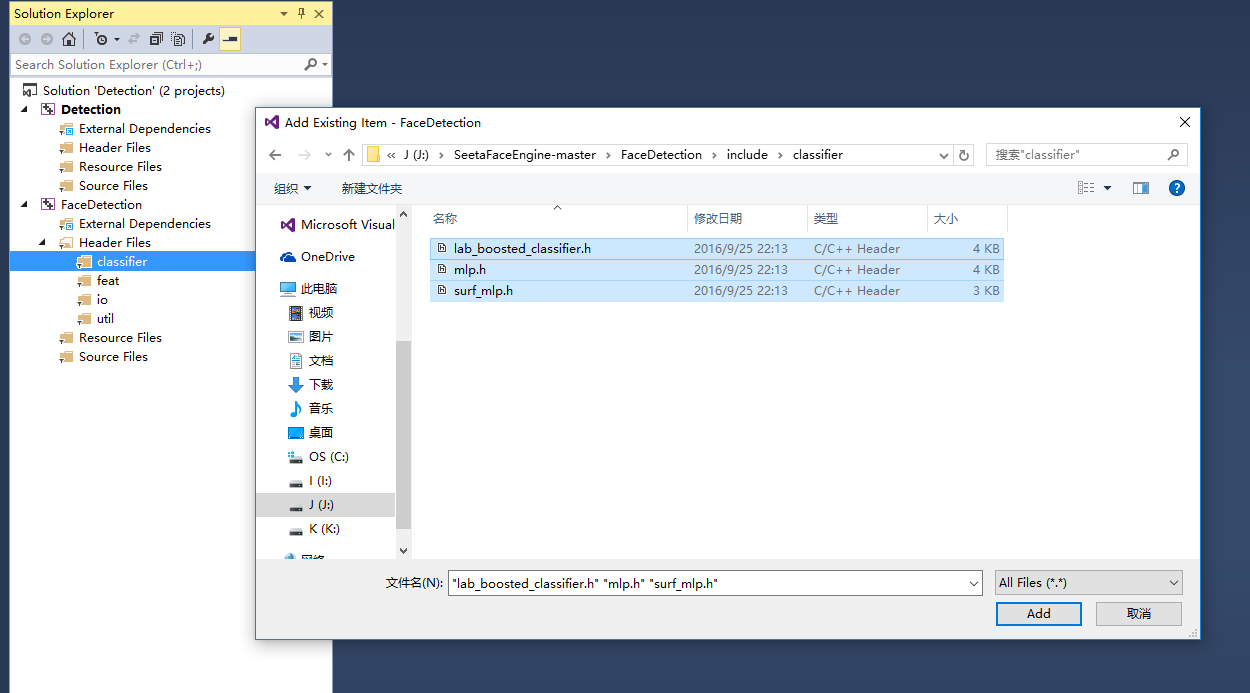
### 3 DLL工程的配置（就是FaceDetection工程文件的配置）

#### (1)添加文件

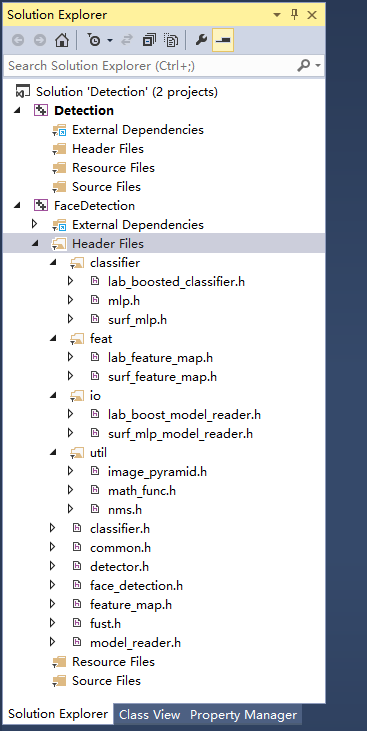
在解决方案资源管理器的头文件处右键添加“新建筛选器”，然后参看SeetaFaceEngine中的FaceDetection文件夹中的include文件夹，“新建筛选器”和include一样，有几个文件夹就建立几个“筛选器”，筛选器命名跟include文件夹下的文件夹相同，在相应的“筛选器”中添加相同名字文件夹中的.h文件（方法：右键筛选器，添加现有项，添加对应的.h文件），如下图，而include文件夹中的.h直接加到头文件下。



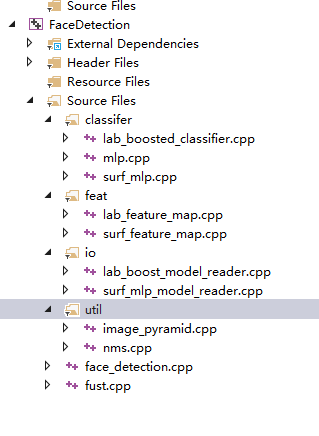




加完之后：

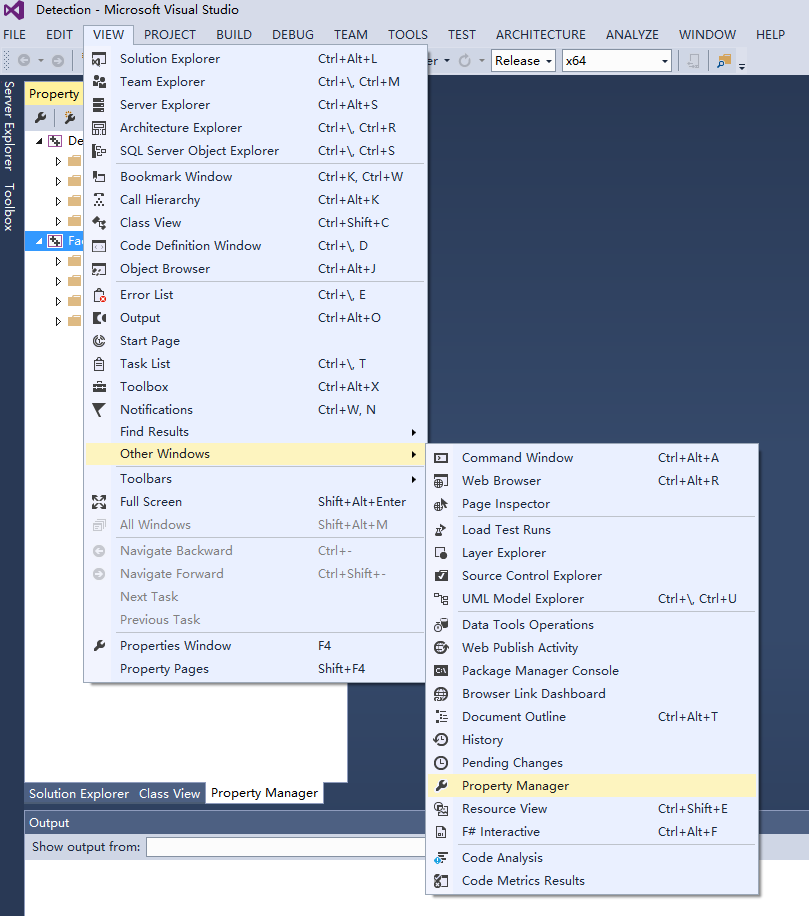


和头文件一样，在源文件处新建筛选器，FaceDetection/src下文件夹除了test全加，然后将FaceDetection/src下各个文件夹**（除了test文件夹）**中的.cpp文件加入到相应的筛选器中。结果如下：



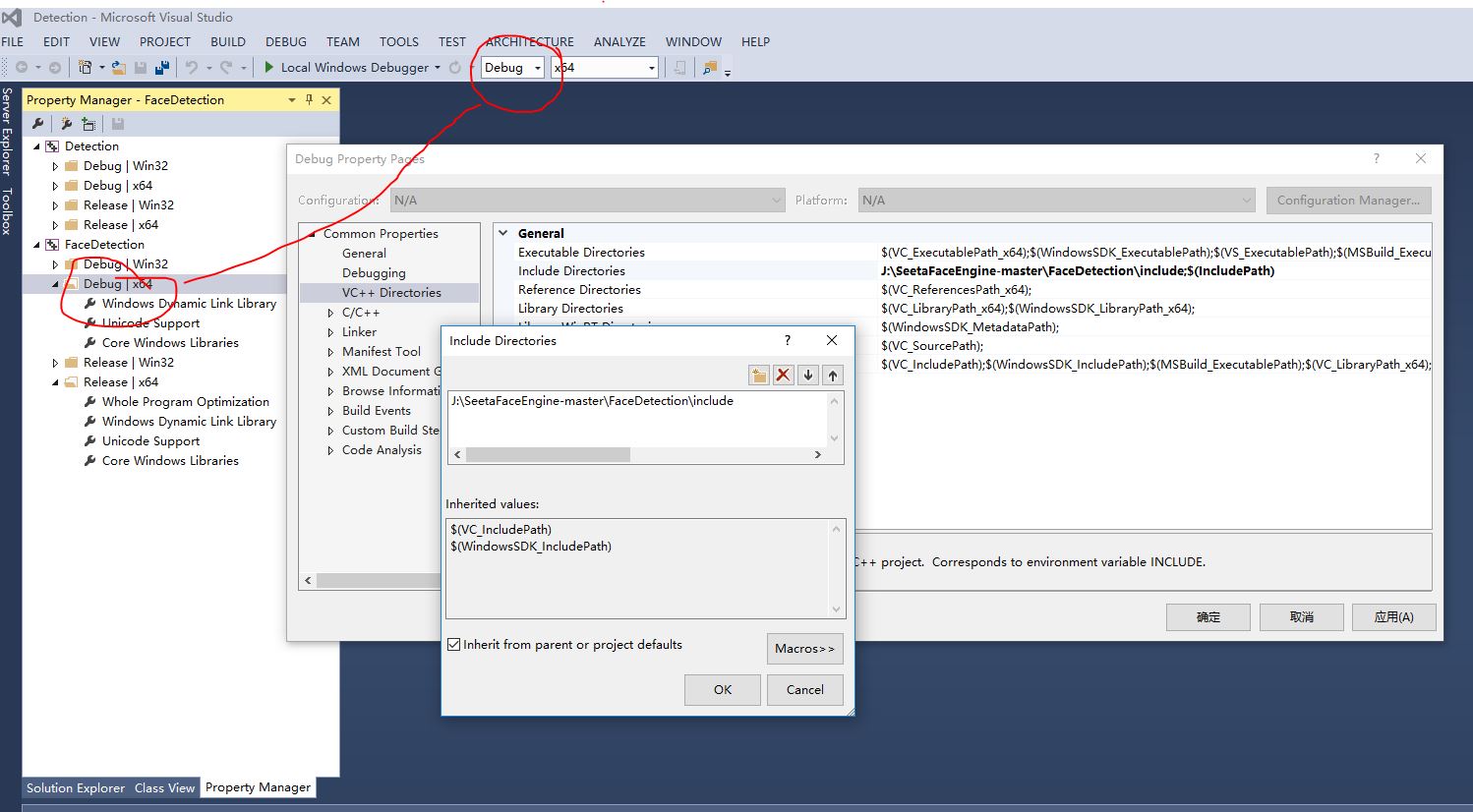
#### （2）属性管理器

点击视图，“其他窗口”中的“属性管理器”。



在Debug配置下，在属性管理器中展开FaceDetection。找到相应的64位，Debug属性文件夹（Debug|x64），双击进入属性，在VC++目录中的包含目录中，添加SeetaFaceEngine中FaceDetection的include路径。如下图。

点生成，重新生成解决方案。



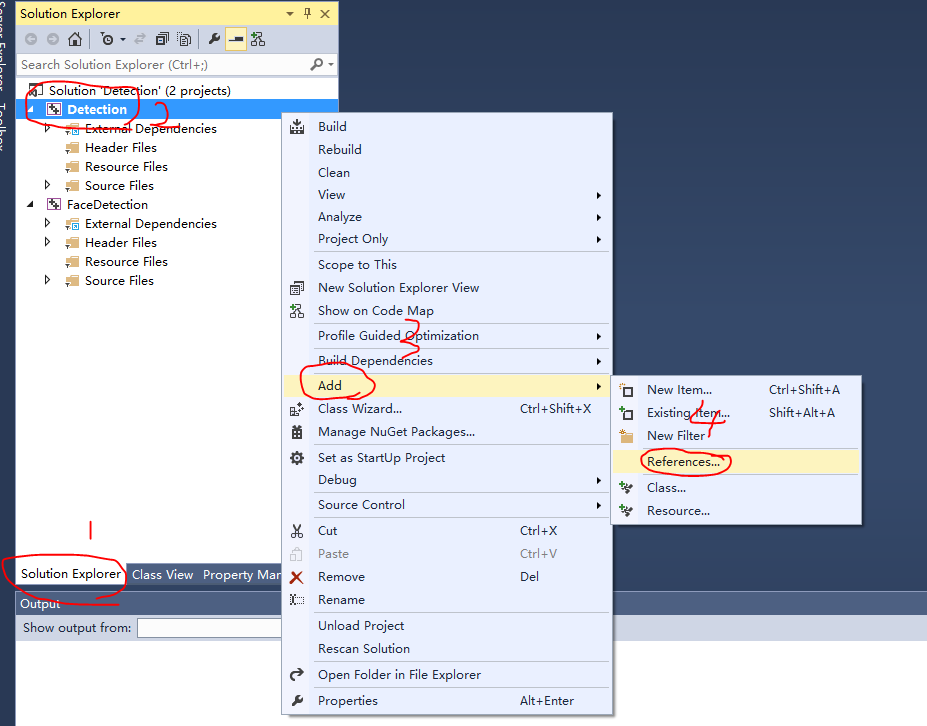
将活动解决方案配置改为Release，然后双击Release|x64属性文件夹，将include地址放到包含目录中。

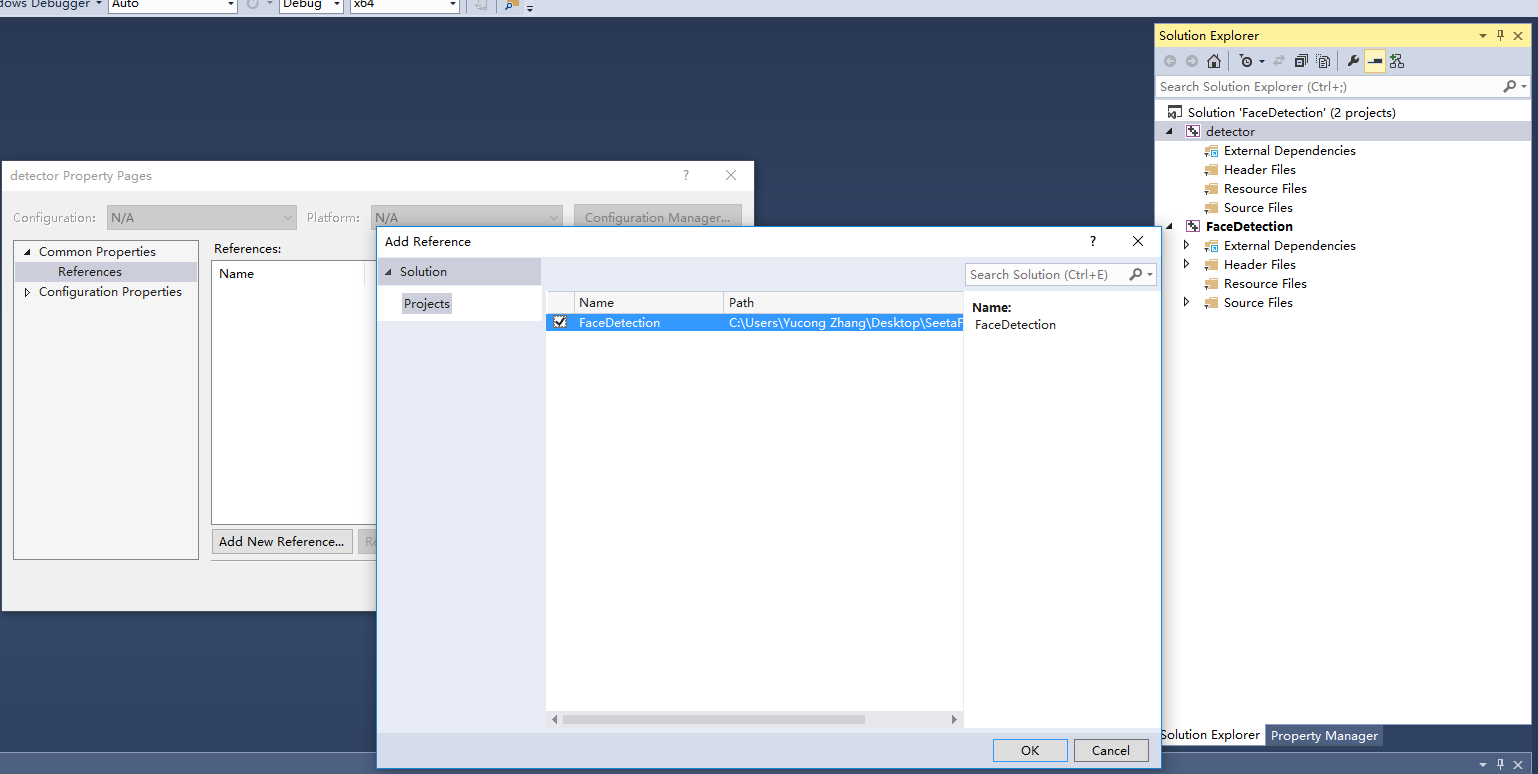
点生成，重新生成解决方案。

### 4 Console application（Detection）工程文件配置

#### （1）引用seetafacedetection

在新产生的Detection右键，添加引用，添加新引用。勾选FaceDetection，确定。

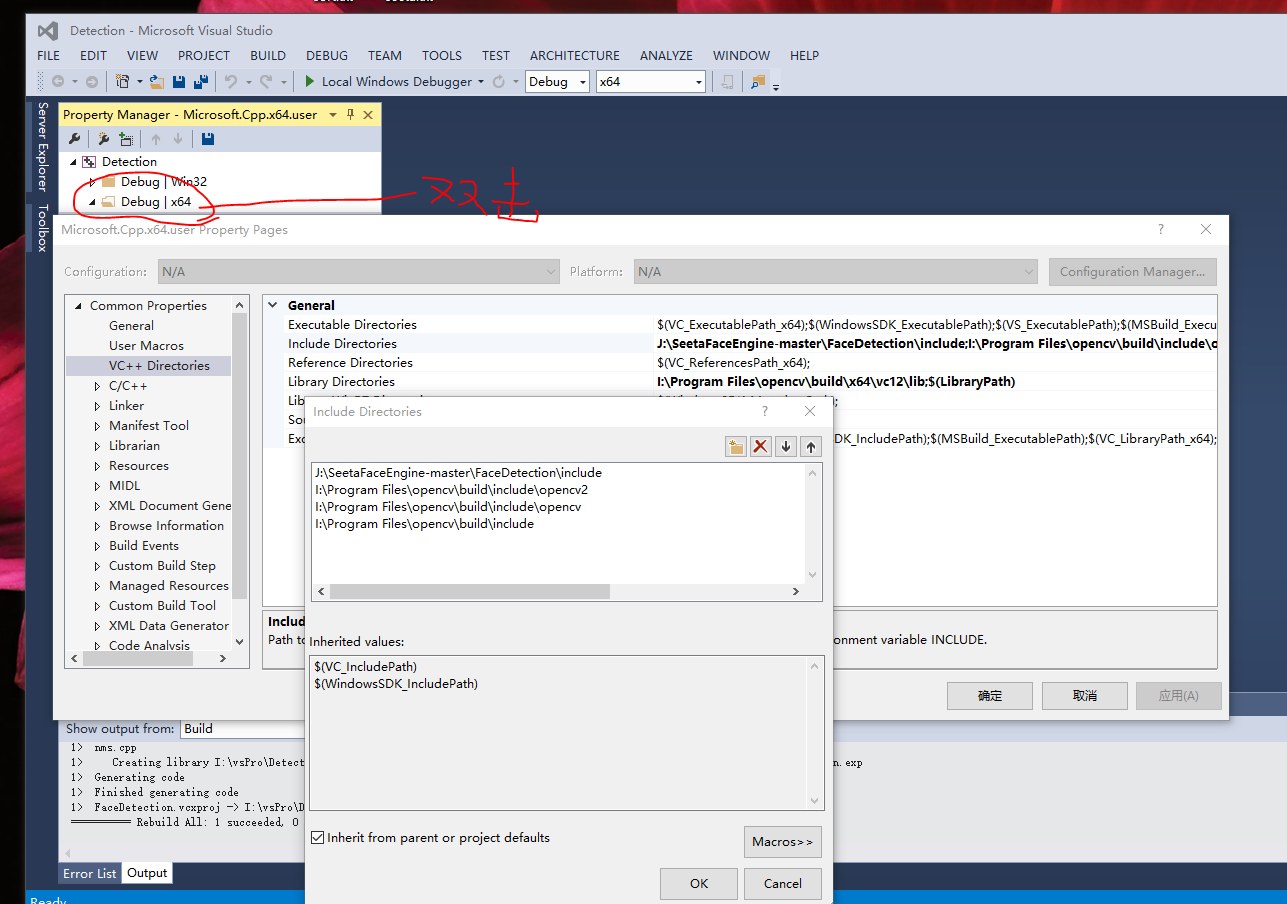




在属性管理器中找到Detection，展开有四个属性文件夹，找到和活动解决方案配置相同的x64文件夹（在Debug配置下找到Debug|x64，Release配置下找到Release|x64），在相应的Release和Debug的在“VC++目录”的包含目录中添加包含目录中加入seeta的include路径（SeetaFaceEngine-master\FaceDetection\include）

#### (2)配置opencv

在属性管理器中找到Detection，展开有四个属性文件夹，找到和活动解决方案配置相同的x64文件夹（在Debug配置下找到Debug|x64，Release配置下找到Release|x64），然后在“VC++目录”的包含目录中添加3个路径，分别为OpenCV的build\include、 include\opencv和include\opencv2的路径。如下图。



分别在Debug和Release的配置下，在 Detection库目录中加入I:\Program Files\opencv\build\x64\vc12\lib。（其中VS2013是vc12，VS2012是vc11。VS2010是vc10。根据自己情况进行选择，x64是64位版本）

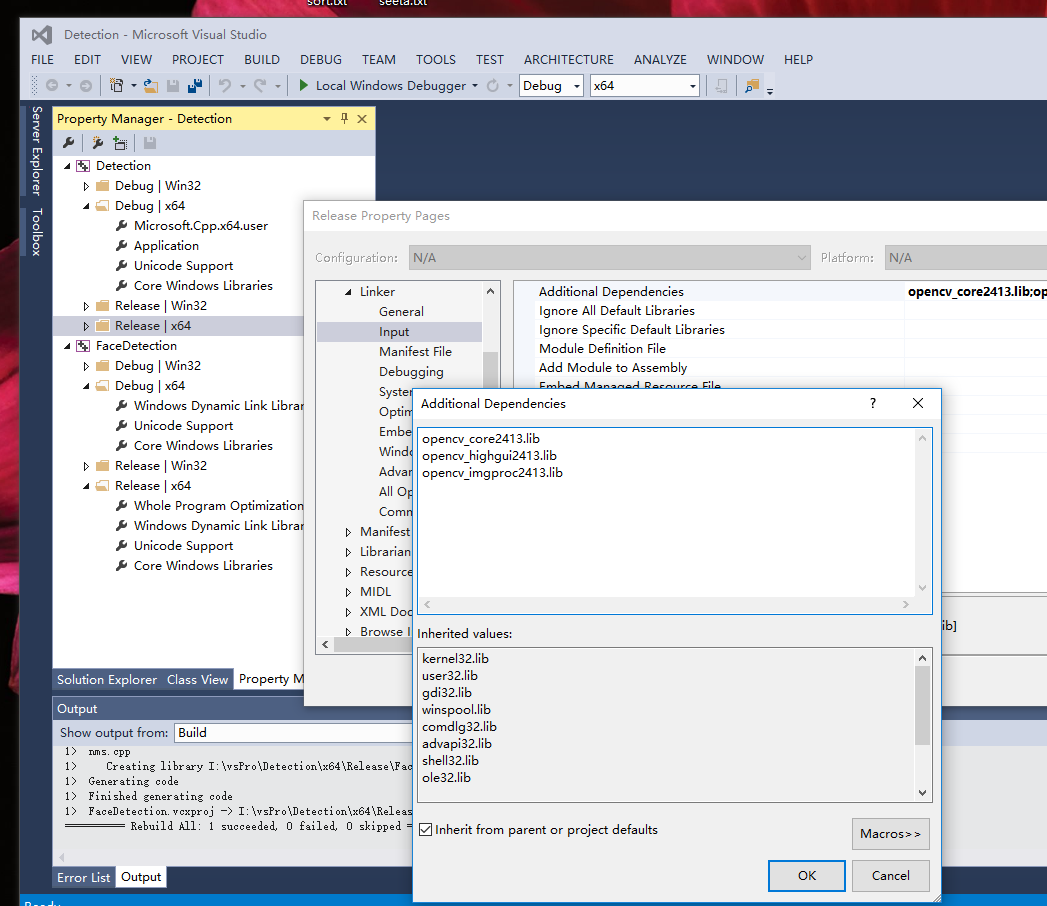
在属性管理器的Detection里有两个Release，一个32位的，一个64位的，在Release|x64属性中的链接器中输入附加依赖项，如下图的3个.lib文件。（其中的2413是OpenCV的版本，自己根据自己电脑的OpenCV版本修改）。

三个lib文件为：

opencv\_core2413.lib

opencv\_highgui2413.lib

opencv\_imgproc2413.lib



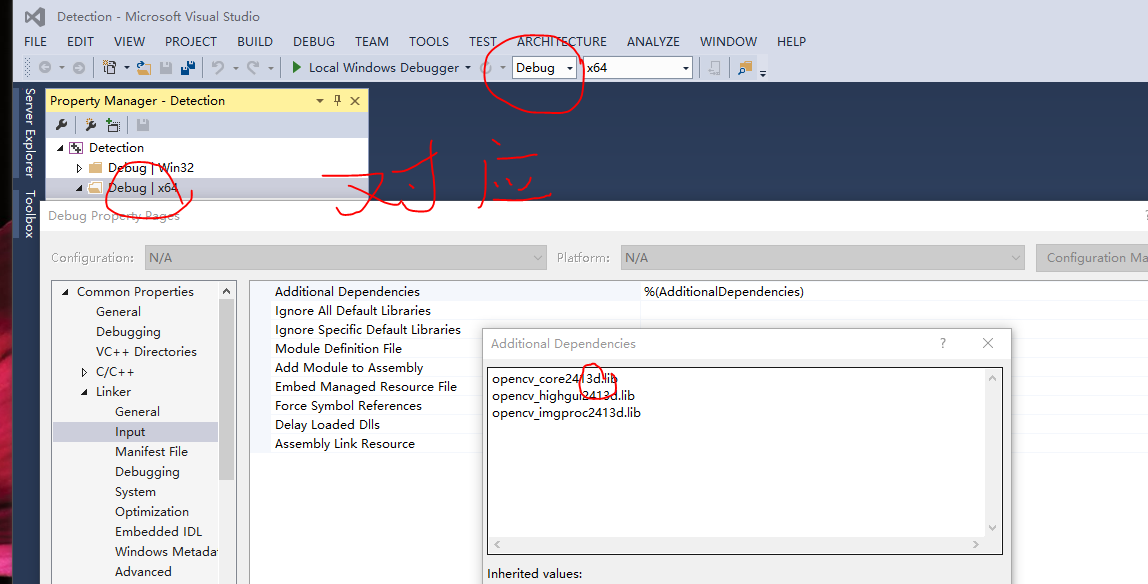
Debug配置下，在Debug|x64中添加如下3个.lib库，记得.lib之前有个d，和Release版不同。

Debug配置下，三个lib文件为：

opencv\_core2413d.lib

opencv\_highgui2413d.lib

opencv\_imgproc2413d.lib



#### （3）例子

在解决方案资源管理器窗口中，在Detection的源文件中添加现有项，把FaceDetection/src/test的test.cpp文件添加进去。

修改实际情况模型地址和图片地址

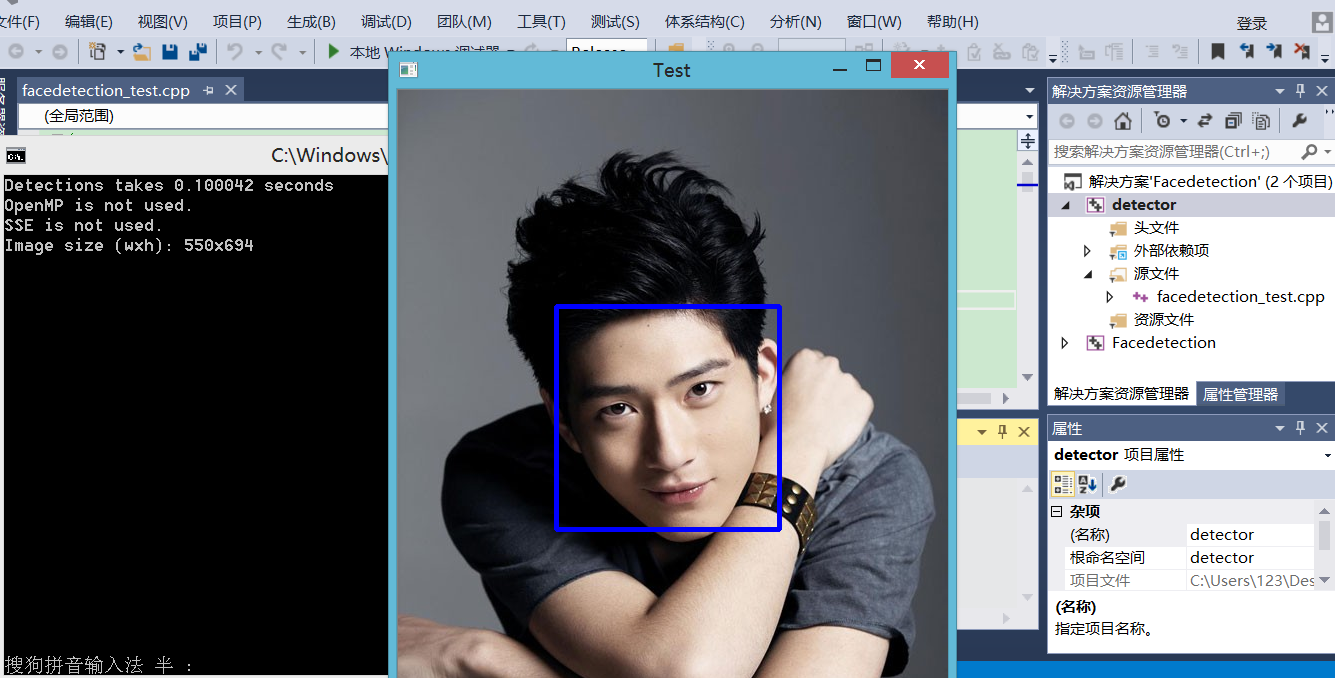


在解决方案资源管理器中的FaceDetection上右键，重新生成，在detector右键，重新生成

点击解决方案资源管理器。在Detection上右键，设为启动项目。

然后点调试，开始执行（不调试）。

成功，产生结果



## 二、面部特征点定位模块SeetaFace Alignment

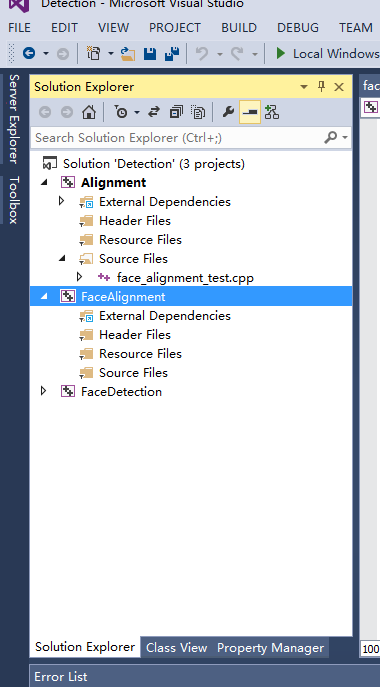
这一部分在第一步的基础上，新建两个New Project，一个DLL（在以下过程中命名为SeetaFace Alignment），一个Console application（在以下过程中命名为Alignment）用于调用SeetaFace Alignment（校准阶段需要建立在检测人脸的基础上，所以需要引入SeetaFaceDetection）的SDK。

### 1新建项目工程

（1）新建项目，创建一个Console application项目。选择win32控制台应用程序。重命名Alignment。选择存放位置。确定后选择下一步，选择Console application，空项目。

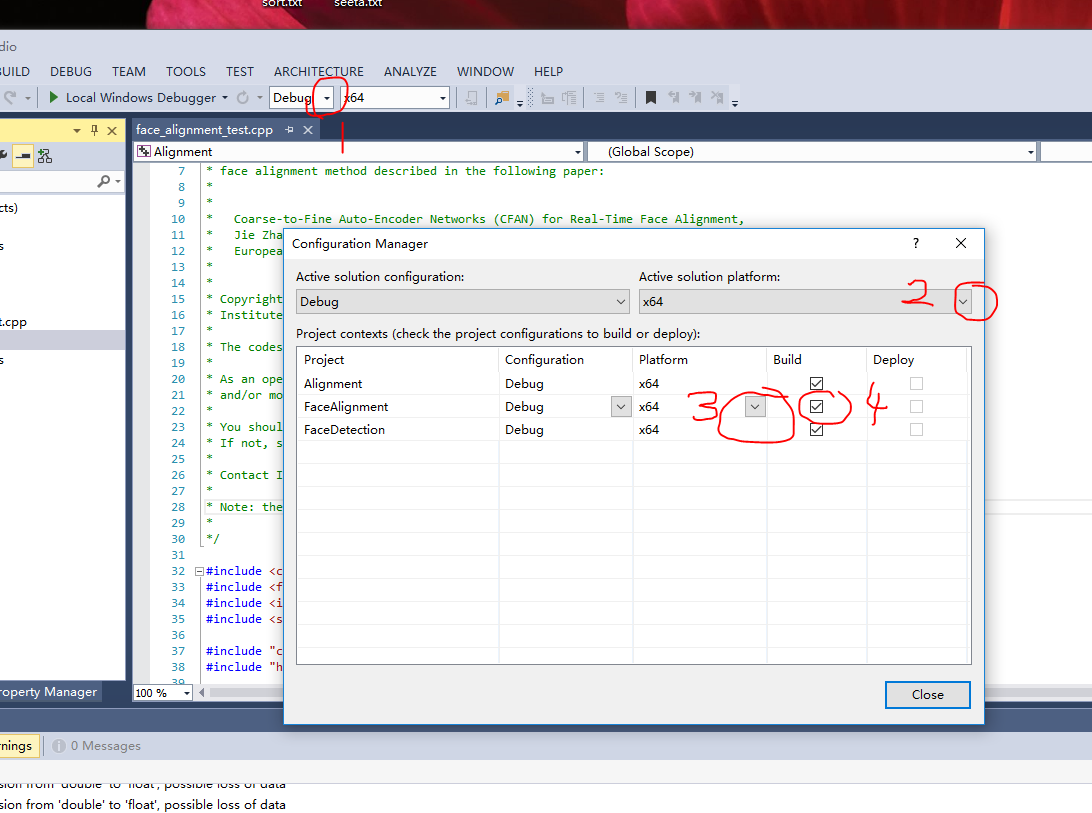
（2）创建DLL项目。选择win32控制台应用程序。分别重命名FaceAlignment。选择DLL存放位置。确定后选择下一步，选择DLL，空项目。

新建好后如图所示：

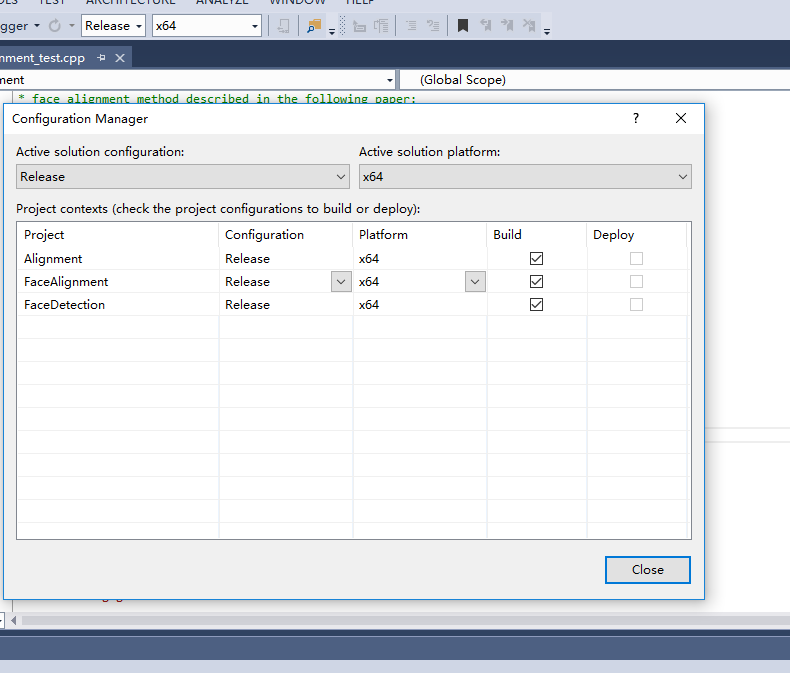


### 2平台配置（这里配的都是X64的）

（1）进入配置管理器，“平台”位置新建，新建平台选择x64。活动解决方案平台选择x64，注意每一个项目的平台都应该与上方的“活动解决平台”一致（这里是X64），找不到选项位置的话可以参考Detection的配置图解。过程结果如图所示：



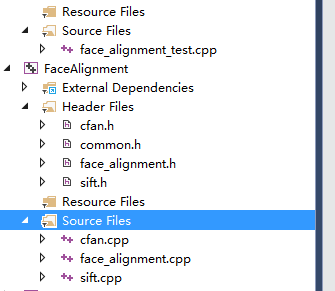
Release也要是这样的呦



### 3 DLL工程的配置（就是FaceAlignment工程文件的配置）

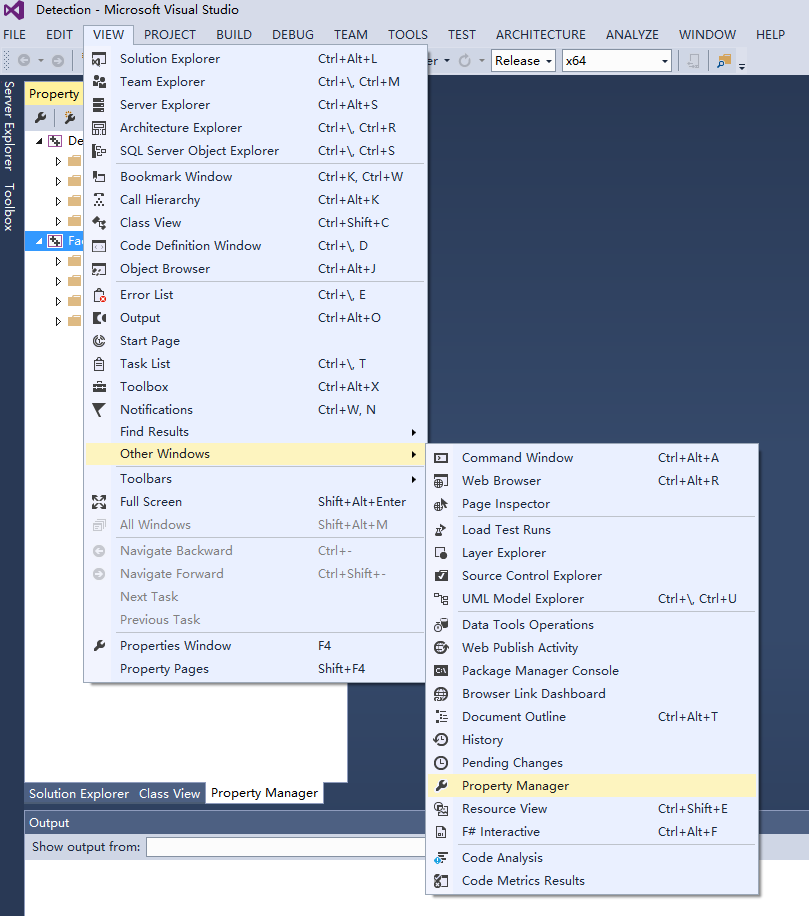
#### (1)添加文件

在解决方案资源管理器的FaceAlignment头文件处右键添加现有项，将SeetaFaceEngine中FaceAlignment的include中的几个.h文件加进去。在源文件处右键添加现有项，将src中的几个.cpp文件加进去。

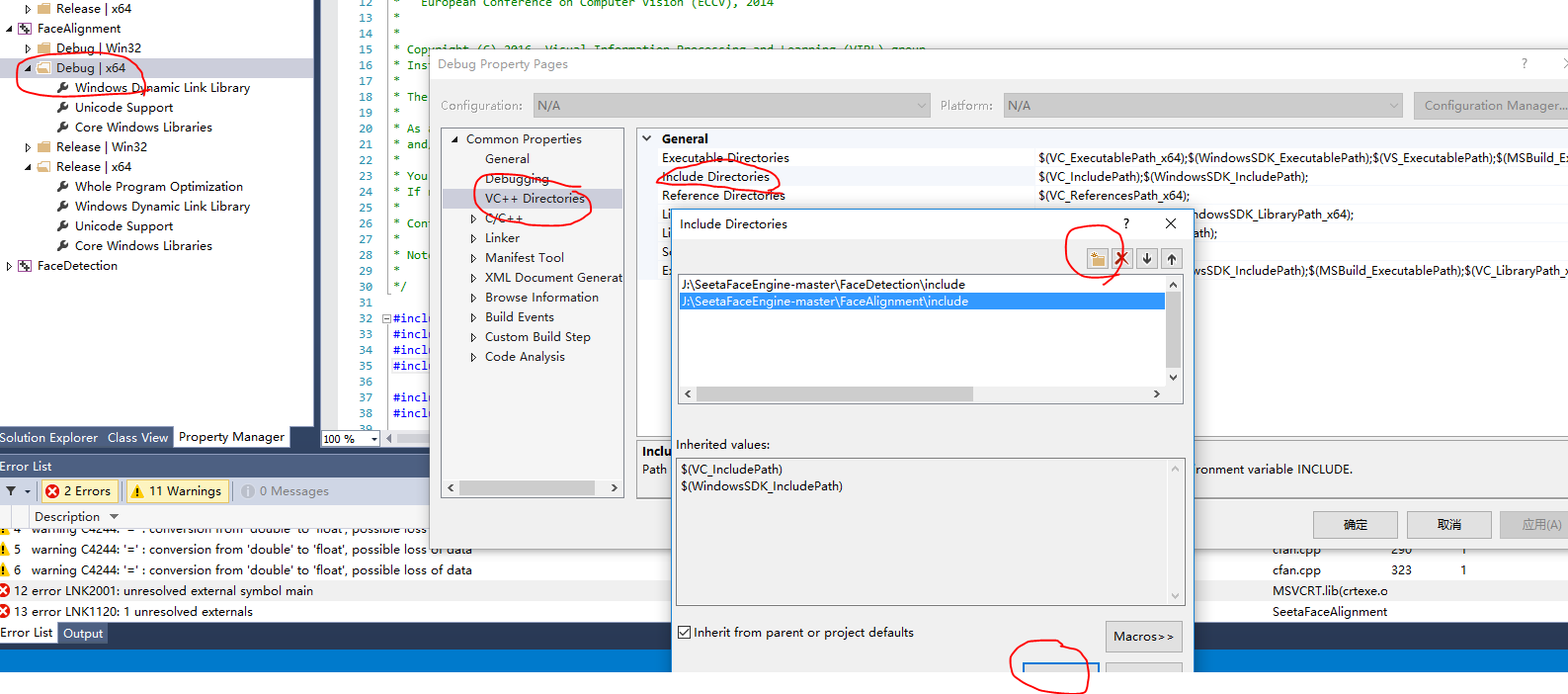


#### （2）属性管理器

点击视图，“其他窗口”中的“属性管理器”。



在Release配置下，在属性管理器中双击FaceAlignment。找到相应的64位，Release属性文件夹（Release|x64），双击进入属性，在VC++目录中的包含目录中，添加FaceAlignmet和FaceDetection中的include路径。点生成，重新生成解决方案。

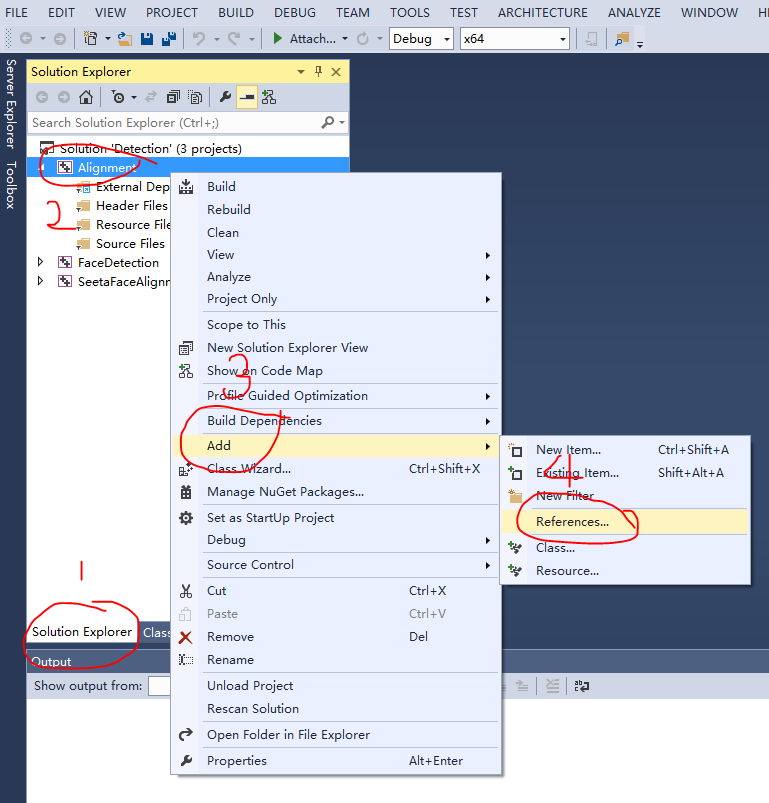
将活动解决方案配置改为Debug，然后双击Debug|x64属性文件夹，将include地址（两个）放到包含目录中。如下图。

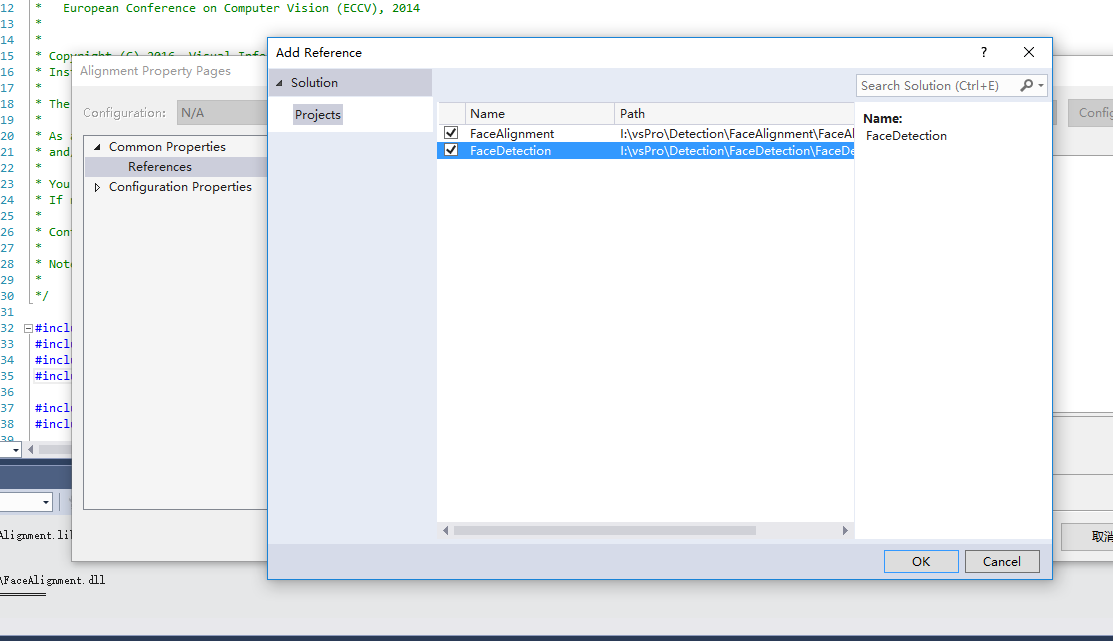
点生成，重新生成解决方案。

### 4 Console application（Alignment）工程文件配置

#### （1）引用seetafacedetection、seetafacealignment

在**Alignment**右键，添加引用，添加新引用。勾选FaceDetection，FaceAligment确定。





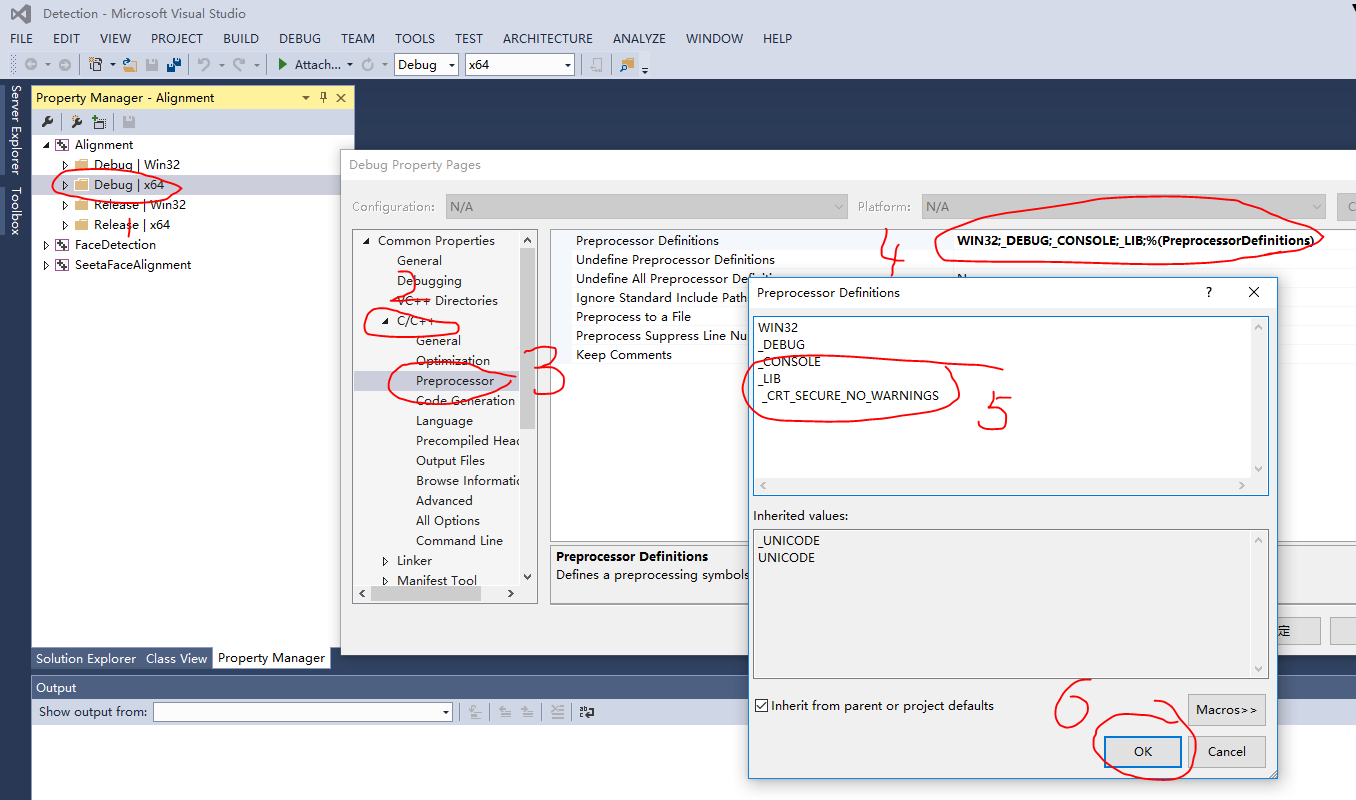
在属性管理器中找到**Alignment**，展开有四个属性文件夹，找到和活动解决方案配置相同的x64文件夹（在Debug配置下找到Debug|x64，Release配置下找到Release|x64），**分别**在相应的Release和Debug的在“VC++目录”的包含目录中添加包含目录中加入两个seeta的include路径（SeetaFaceEngine-master\FaceDetection\include和SeetaFaceEngine-master\FaceAlignment\include）

#### (2)配置opencv

在属性管理器中找到**Alignment，**对其进行opencv的相关配置，可参考上一步对Detection中opencv的配置[(2)配置opencv](#_(2)配置opencv)。

#### (3）配置预处理器

在属性管理器中找到**Alignment，**分别调到Release与Debug配置下，进入Realease|x64和Debug|x64的“属性>“C/C++”>预处理器”，预处理器定义中加入 \_CRT\_SECURE\_NO\_WARNINGS



#### （4）例子

在解决方案启动项上，将Alignment（右击）设为启动项。

在Alignment的资源文件中加入SeetaFace\SeetaFaceEngine-master\FaceAlignment\src\test下的face\_Alignment\_test.cpp文件。

将代码中DATA\_DIR和MODEL\_DIR改为相应的地址，如下图所示。



重新生成Alignment，然后调试，启动调试，成功。在自己建的项目FaceDetection/Alignment路径下输出图片，可以修改test中的图片地址，试试自己的图片（注意图片扩展名）



## 一、人脸识别模块SeetaFaceIdentification

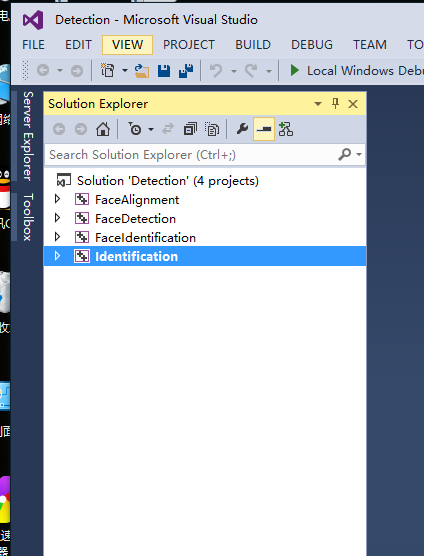
这一部分的配置，在前两步的基础上，添加两个New Project，一个DLL（在以下过程中命名为FaceIdentification），一个Console application（在以下过程中命名为**Identification**）用于调用SeetaFace的SDK。

### 1新建项目工程

（1）新建项目，创建一个Console application项目。选择win32控制台应用程序。重命名Identification。选择存放位置。确定后选择下一步，选择Console application，空项目。

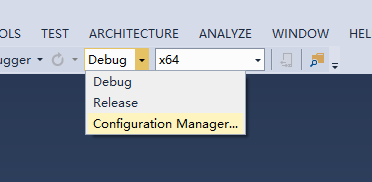
（2）创建DLL项目。选择win32控制台应用程序。命名为FaceIdentification，选择DLL存放位置。确定后选择下一步，选择DLL，空项目。

添加完成后

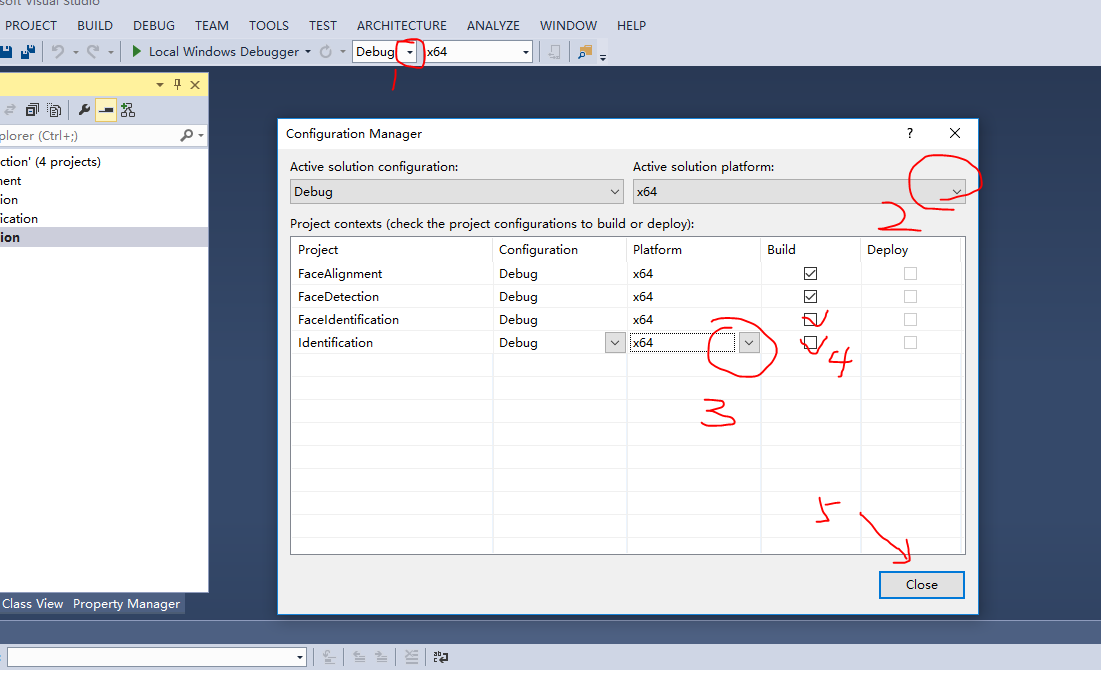


### 2平台配置（这里配的都是X64的）

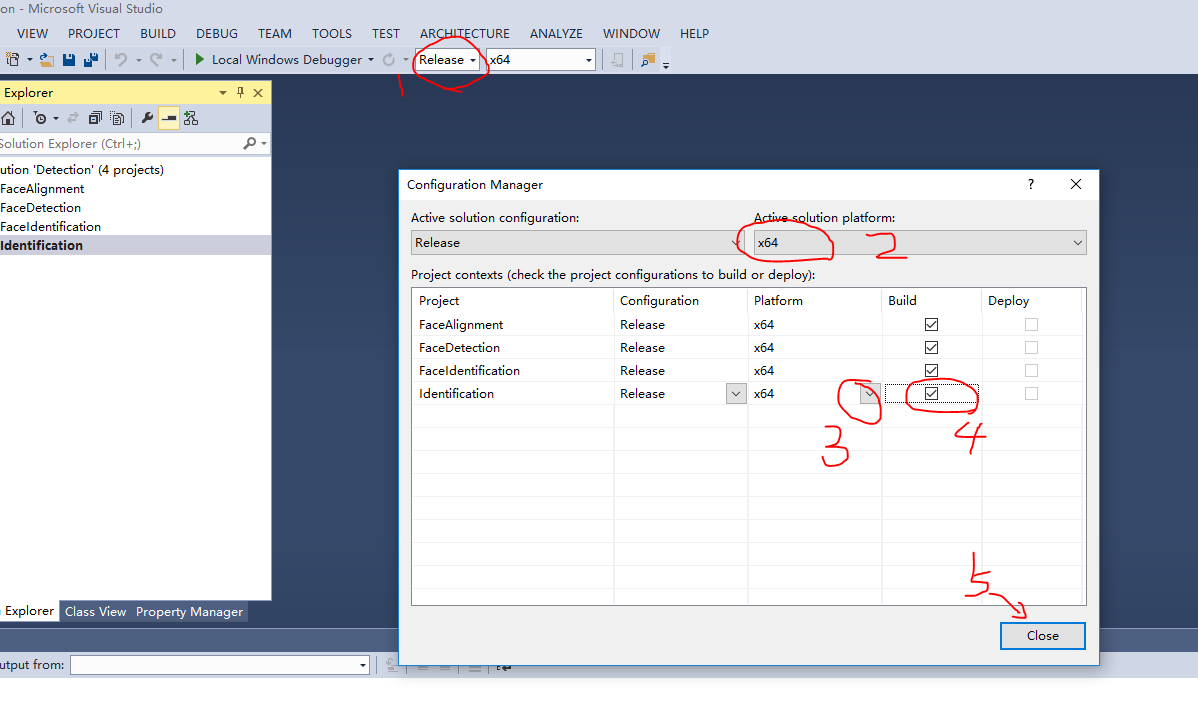
（1）进入配置管理器，“平台”位置新建，新建平台选择x64。活动解决方案平台选择x64.



配置结果如图所示：



Release也要是这样的呦

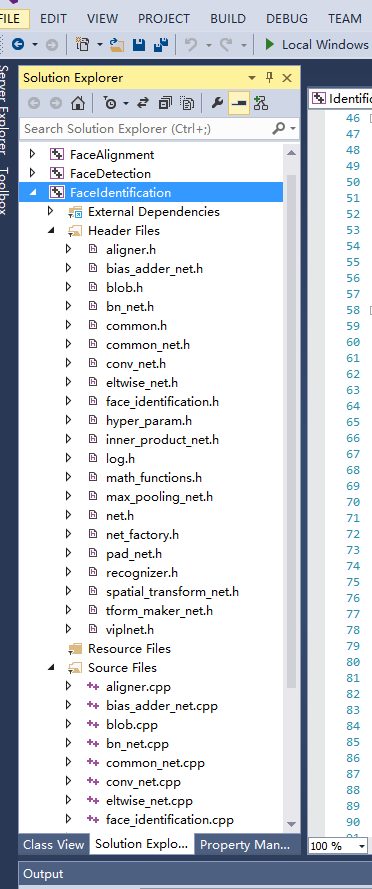


### 3 DLL工程的配置（FaceIdentification工程文件的配置）

#### (1)添加文件

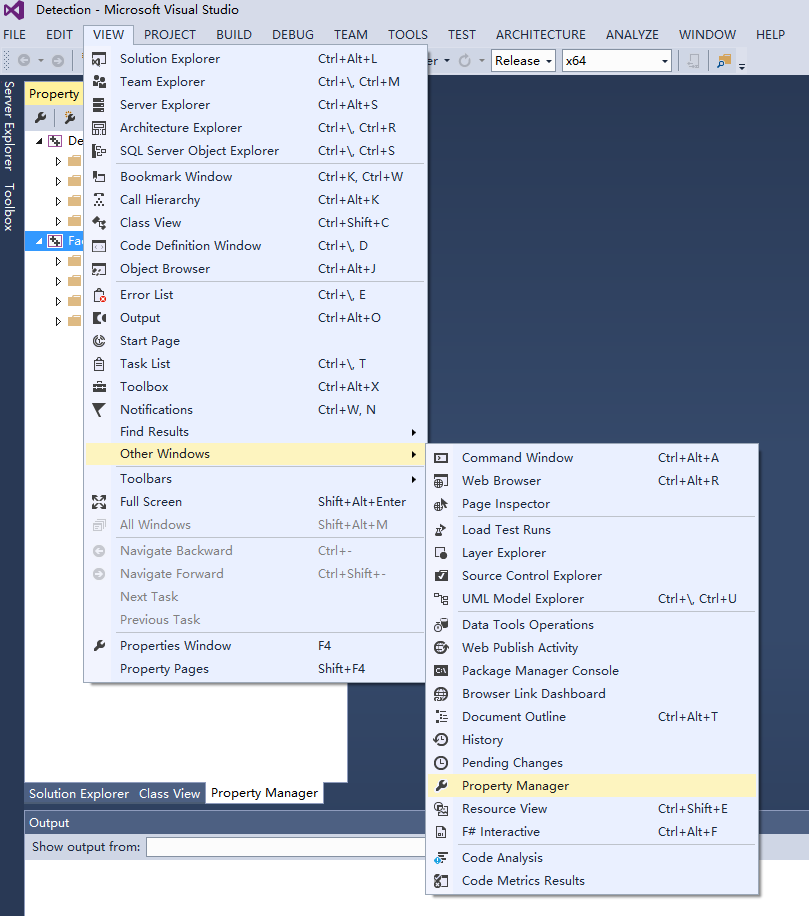
在解决方案资源管理器的FaceIdentification头文件处右键添加现有项，将SeetaFaceEngine中FaceIdentification的include中的几个.h文件加进去。在源文件处右键添加现有项，将src中的几个.cpp文件加进去，**将tools文件夹中的两个.cpp也添加到源文件中。**

结果如图所示：



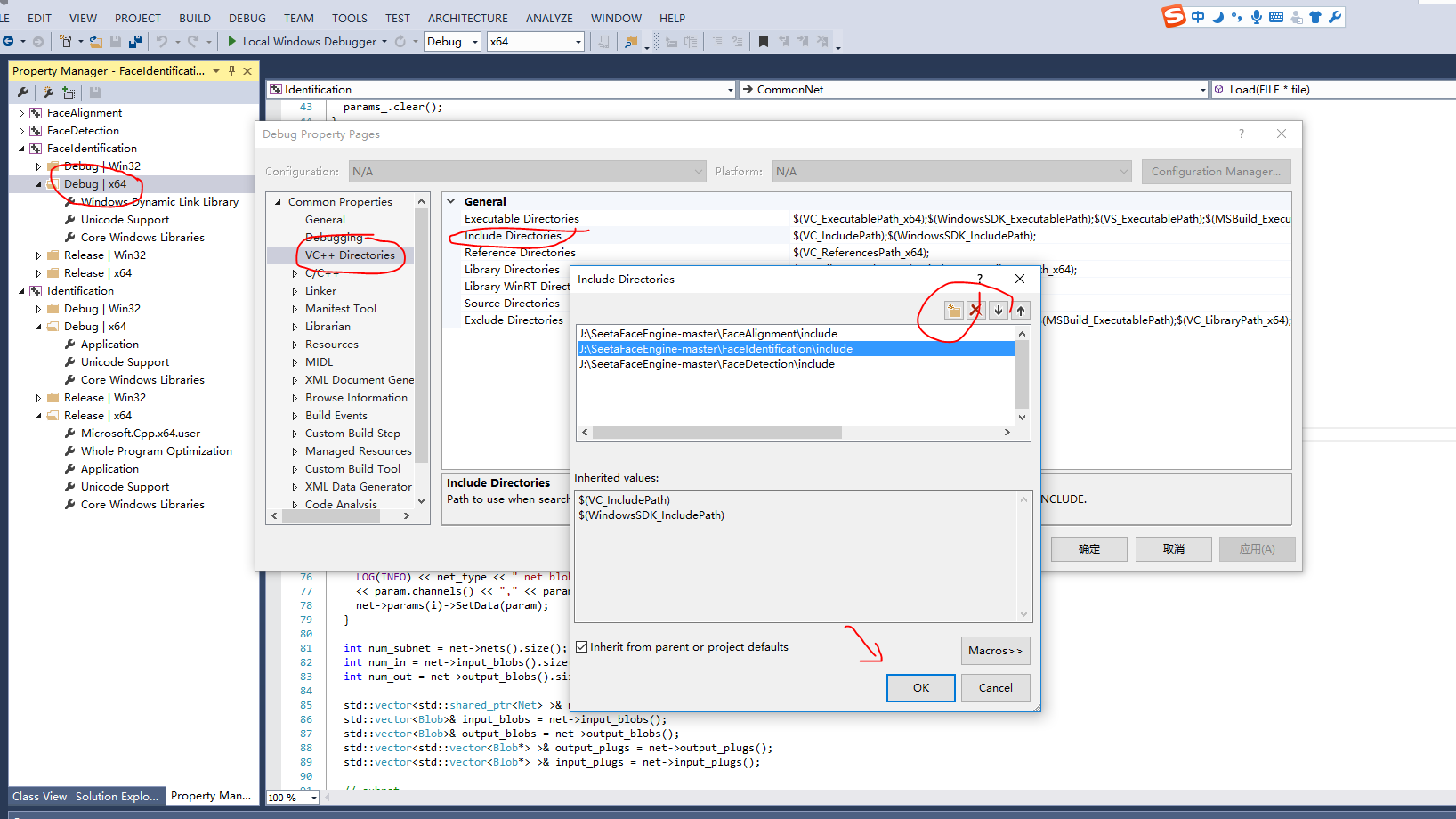
#### （2）属性管理器

点击视图，“其他窗口”中的“属性管理器”。



在Debug配置下，在属性管理器中双击FaceIdentification。找到相应的64位，Debug属性文件夹（Debug|x64），双击进入属性，在VC++目录中的包含目录中，添加SeetaFaceEngine中的include路径（3个）。如下图。

右键点生成，重新生成解决方案。



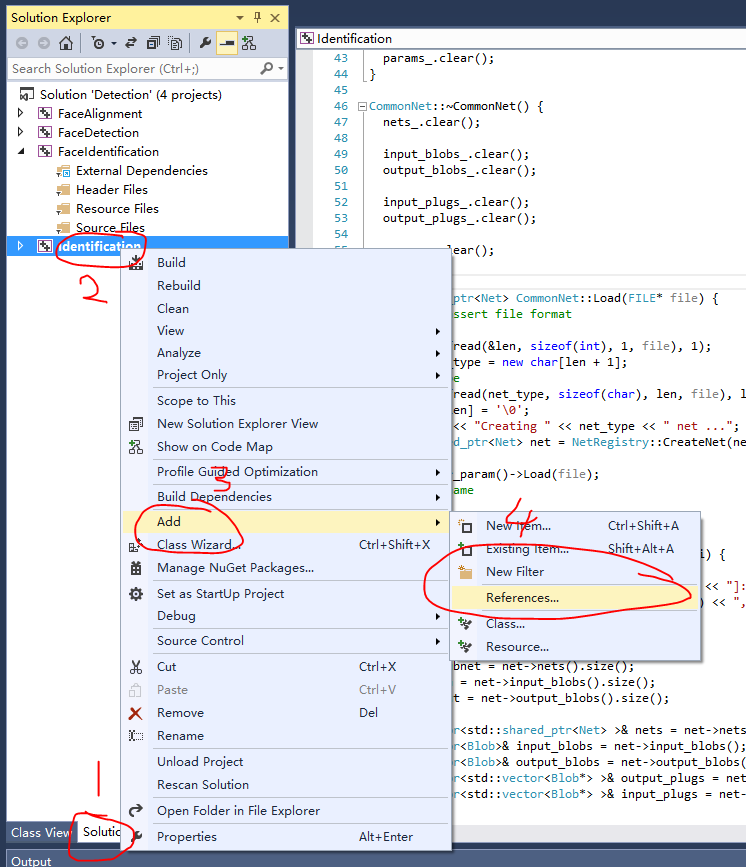
将活动解决方案配置改为Release，然后双击Release|x64属性文件夹，将3个include地址放到包含目录中。

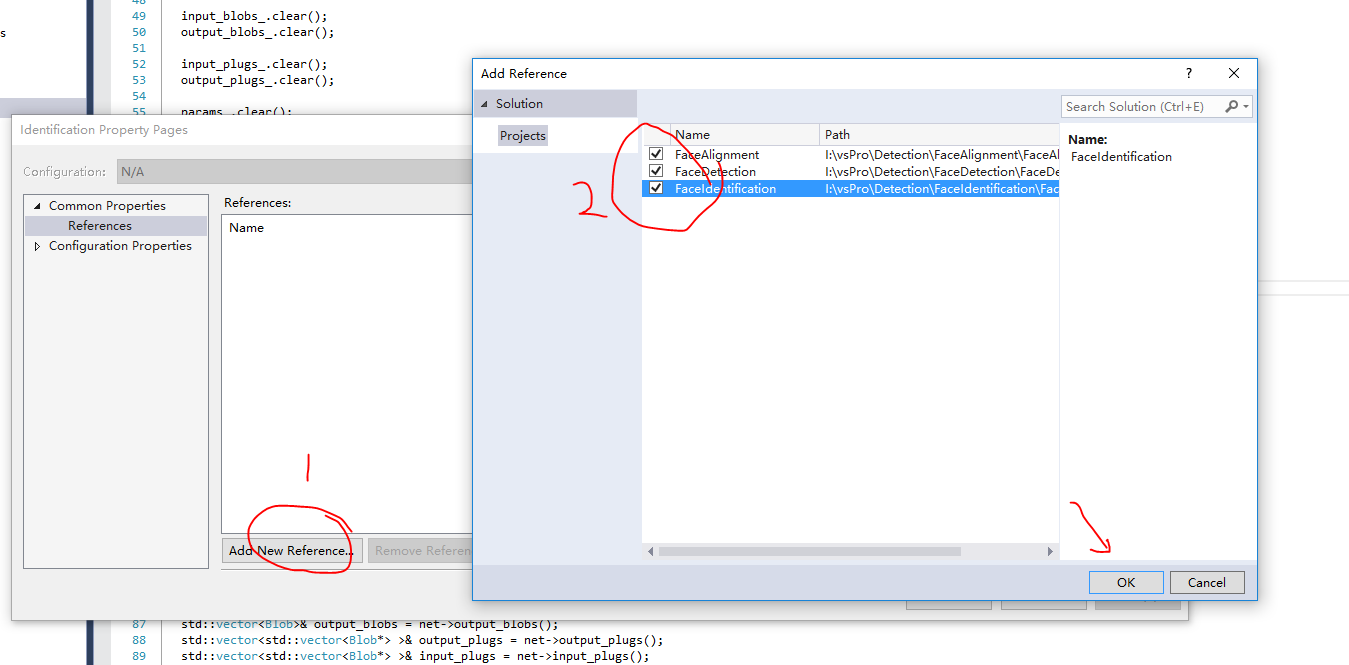
右键点生成，重新生成解决方案。

### 4 Console application（Identification）工程文件配置

#### （1）引用Facedetection、Facealignment、FaceIdentification

在**Identification**右键，添加引用，添加新引用。勾选FaceDetection、FaceAlignment、FaceIdentification确定。



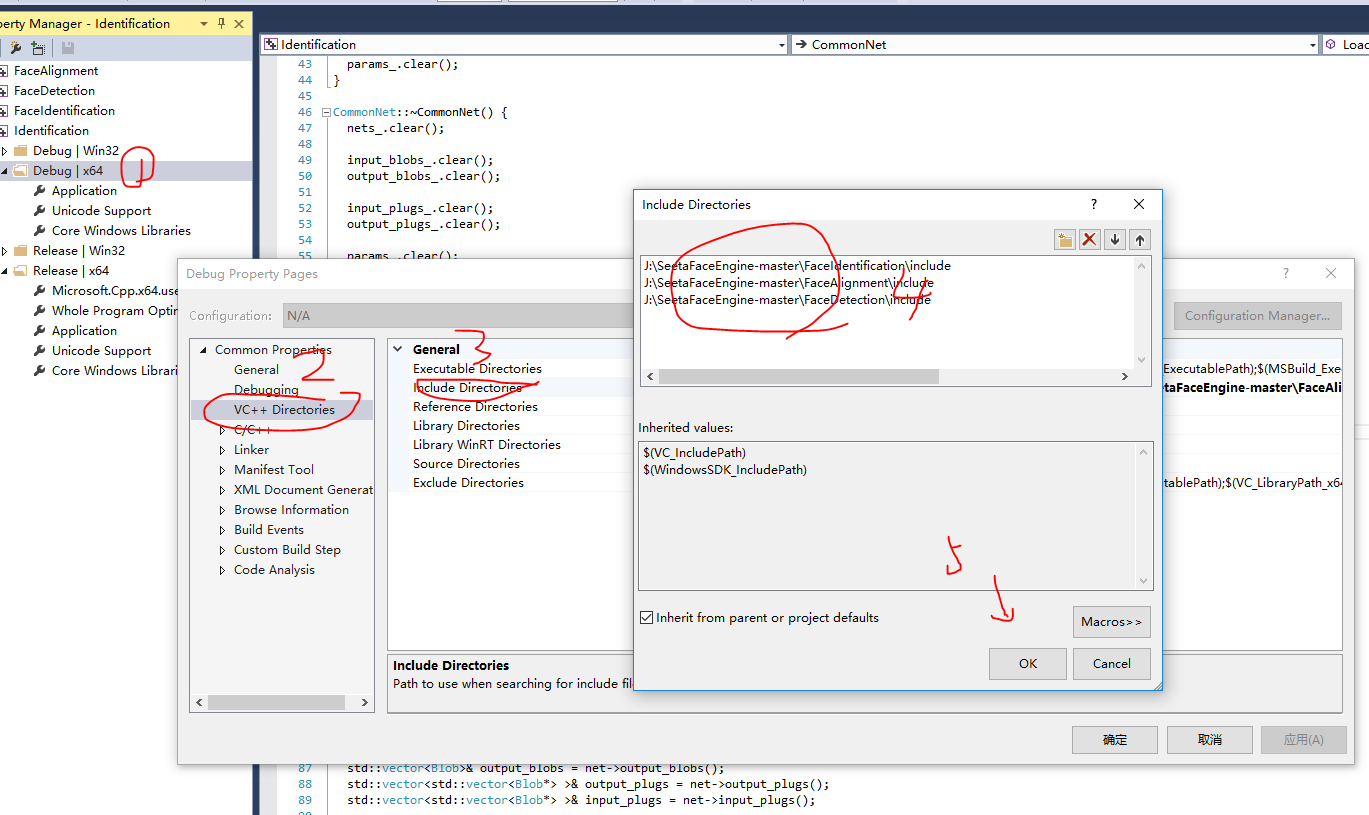


#### （2）添加include路径

在属性管理器中找到**Identification**，展开有四个属性文件夹，找到和活动解决方案配置相同的x64文件夹

（在Debug配置下找到Debug|x64，Release配置下找到Release|x64）

在相应的Release和Debug的在“VC++目录”的包含目录中添加包含目录中加入三个seeta的include路径（SeetaFaceEngine-master\FaceDetection\include和SeetaFaceEngine-master\FaceAlignment\include以及SeetaFaceEngine-master\FaceIdentification\include）

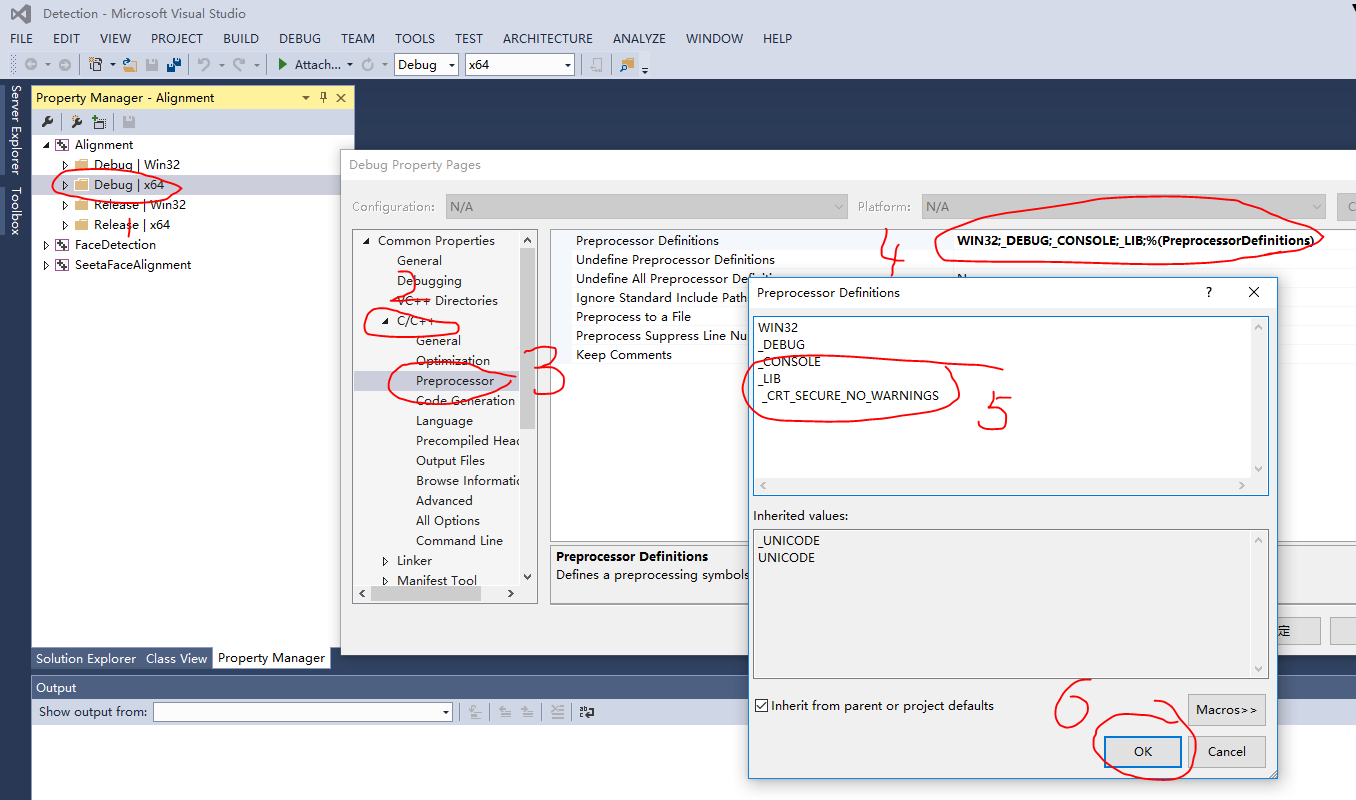


#### (2)配置opencv

在属性管理器中找到**Identification，**对其进行opencv的相关配置，可参考上一步对Detection中opencv的配置[(2)配置opencv](#_(2)配置opencv)。

#### (3）配置预处理器

在属性管理器中找到**Identification，**分别调到Release与Debug配置下，进入Realease|x64和Debug|x64的“属性>“C/C++”>预处理器”，预处理器定义中加入 \_CRT\_SECURE\_NO\_WARNINGS。



#### （4）例子

（1）在解决方案资源管理器窗口中，在**Identification**的源文件中添加现有项，把SeetaFaceEngine-master\FaceIdentification\src\test的test\_face\_recognizer.cpp文件添加进去。

修改实际情况模型地址和图片地址

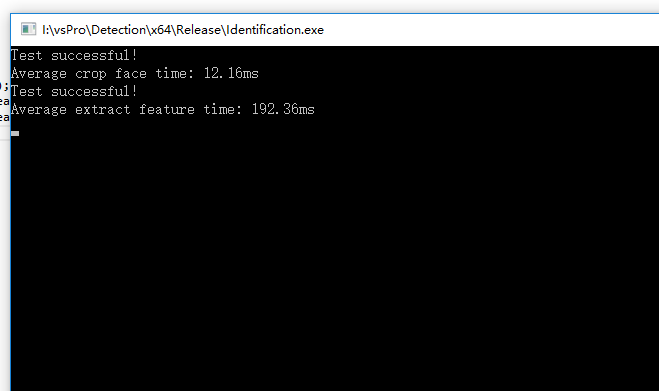


在解决方案资源管理器中在Identification右键，重新生成。

点击解决方案资源管理器。Identification上右键，设为启动项目。

然后点调试，开始执行（不调试）注意编译平台应该是X64。

成功，产生结果



（2）在解决方案资源管理器窗口中，在**Identification**的源文件中添加现有项，把SeetaFaceEngine-master\FaceIdentification\src\test的test\_face\_verification.cpp文件添加进去。

修改实际情况模型地址和图片地址



在解决方案资源管理器中在Identification右键，重新生成。

点击解决方案资源管理器。Identification上右键，设为启动项目。

然后点调试，开始执行（不调试）注意编译平台应该是X64。

成功，产生结果

