目录

DFace SDK 安装说明书(v2.0.0)

1.0 Linux amd64 系统安装

1.1 环境变量设置

下载 dface_sdk 并解压(**tar -xvf 命令**)，注意一定要在 Linux 下的 ext 文件系统下解压，保持压缩包里的库软链不受破坏。

把 DFace SDK 目录下的 lib/ 目录 和 system/lib/ 目录 添加到 **LD_LIBRARY_PATH** 环境变量

```
export LD_LIBRARY_PATH={dface_sdk}/lib:{dface_sdk}/system/lib
```

1.2 工具包使用

DFace SDK 在 tools 目录下提供了几个工具可以快速在目标机器展开人脸识别类测试。使用 --help 参数可以查看具体使用说明。

testDetectImage（人脸检测图片工具)

Usage: testDetectImage [model_path] [num_threads] [min_facesize] [img_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[img_path]: 图片路径

testDetectCamera（人脸检测摄像头工具)

Usage: testDetectCamera [model_path] [num_threads] [min_facesize] [video_width] [video_height]
[model_path]:模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[video_width]: 摄像头分辨率宽
[video_height]: 摄像头分辨率高

testDetectVideo（人脸检测视频工具)

Usage: testDetectVideo [model_path] [num_threads] [min_facesize] [video_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)

[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[video_path]: 视频路径

testRecognize（人脸识别工具）

Usage: testRecognize [model_path] [num_threads] [min_facesize] [accuracy_mode] [img1_path] [img2_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[accuracy_mode]: 精度模式 0:普通精度 1:高精度(速度稍慢) 2:实时模式(精度稍低)
[img1_path]: 图片 1 路径
[img2_path]: 图片 2 路径

testMassiveCompare（大规模 1:N 人脸比对工具）

Usage: testMassiveCompare [model_path] [num_threads] [accuracy_mode] [size]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[accuracy_mode]: 精度模式 0:普通精度 1:高精度(速度稍慢) 2:实时模式(精度稍低)
[size]: 1:N 规模,N 值

testPoseImage（人脸 68 关键点和 3D 姿态角估计工具）

Usage: testPoseImage [model_path] [num_threads] [min_facesize] [img_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[img_path]: 图片路径

testInfrared（测试红外活体检测）

Usage: testInfrared [model_path] [img_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[img_path]: 图片路径

testRgbantiCamera(测试单目活体检测)

Usage: tesRgbantiCamera [model_path] [pat_threshold] [pat_count] [video_width] [video_height]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[pat_threshold]: pat 阈值
[pat_count]: pat 数量
[video_width]: 摄像头分辨率宽
[video_height]: 摄像头分辨率高

testTrackCamera（人脸跟踪摄像头工具）

Usage: testTrackCamera [model_path] [max_age] [video_width] [video_height]
[model_path]:模型目录(dface_sdk/model/normal_encrypt)
[max_age]: 被跟踪物体的最大过期次数
[video_width]: 摄像头分辨率宽
[video_height]: 摄像头分辨率高

testDatabase（人脸数据库测试工具）

Usage: testDatabase

1.3 测试源码编译

DFace SDK 利用 CMake 相关文件快速构建测试项目，测试源码位于 example 目录。提供脚本编译和手动编译的方法，建议手动编译。

编译工具清单:

工具	描述
编译器	gcc/g++ 版本号 4.8 以上，支持 c++11 标准。
构建工具	Cmake, 版本 3.6 以上。
Ide	vim clion qtcreator codeblocks
标准库	libstdc++6 libc6
链接器	ld.so.3

手动编译，进入 dface_sdk/example/cpp/demo_cmake_simple 目录下的 demo 工程

1. 创建编译目录

mkdir build & cd build

2. 执行 cmake

cmake ..

3. 构建测试项目

make -j 4

构建完之后会在 build 目录生成几个可执行文件

1.4 QT 编译测试

我们也提供了 QT 的 demo 源码，直接用 QTCreator 打开 **dface_sdk/example/cpp/demo_qt 项目即可编译测试。**

2.0 ARM Linux 系统安装

2.1 环境变量设置

1. 下载 dface_sdk 并解压(**tar -xvf 命令**) ,注意一定要在 Linux 下的 ext 文件系统下解压，保持压缩包里的库软链不受破坏。

以下是 GCC/G++ 4.9 版本的配置，其他版本需要具体分析，目标是把 GCC/G++的系统库目录添加到环境变量。

2. 把{dface_sdk}/lib 目录 和 arm-linux-gnueabihf/lib 目录和{dface_sdk}/system/arm-linux-gnueabihf/libc/lib/arm-linux-gnueabihf 添加到 **LD_LIBRARY_PATH** 环境变量。

```
export LD_LIBRARY_PATH={dface_sdk}/lib:{dface_sdk}/system/arm-linux-gnueabihf/lib:{dface_sdk}/system/arm-linux-gnueabihf/libc/lib/arm-linux-gnueabihf
```

```
/opt/tiny_sdk/lib:/opt/tiny_sdk/system/arm-linux-gnueabihf/lib:/opt/tiny_sdk/system/arm-linux-gnueabihf/libc/lib/arm-linux-gnueabihf/
[root@3352-T tiny_sdk]#
```

这个只针对 GCC/G++ 4.9 的 toolchain ,如果其他版本的 GCC/G++类似 ,只需要把对应的 GCC/G++系统库目录加入即可。如果开发板自带的系统库目录和我们提供的 arm-linux-gnueabihf 系统库目录版本一致或兼容，则可以不加 arm-linux-gnueabihf 的系统库目录。

注意: 该环境变量需要同时对 ARM 终端设备和交叉编译的宿主机设置。

3. 添加 **ld-linux-armhf.so.3** 软链

```
ln -s {dface_sdk}/system/arm-linux-gnueabihf/libc/lib/arm-linux-gnueabihf/ld-2.19.2014.07.s0 /lib/ld-linux-armhf.so.3
```

我们在 SDK 目录下提供了 Linux 开发环境下的 GCC/G++和 toolchain 工具。

2.2 工具包使用

DFace SDK 在 tools 目录下提供了几个工具可以快速在目标机器展开人脸识别类测试。使用 --help 参数可以查看具体使用说明。

testDetectImage（人脸检测图片工具)

Usage: testDetectImage [model_path] [num_threads] [min_facesize] [img_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[img_path]: 图片路径

testDetectCamera（人脸检测摄像头工具）

Usage: testDetectCamera [model_path] [num_threads] [min_facesize] [video_width] [video_height]
[model_path]:模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[video_width]: 摄像头分辨率宽
[video_height]: 摄像头分辨率高

testDetectVideo（人脸检测视频工具）

Usage: testDetectVideo [model_path] [num_threads] [min_facesize] [video_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[video_path]: 视频路径

testRecognize（人脸识别工具）

Usage: testRecognize [model_path] [num_threads] [min_facesize] [accuracy_mode] [img1_path] [img2_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[accuracy_mode]: 精度模式 0:普通精度 1:高精度(速度稍慢) 2:实时模式(精度稍低)
[img1_path]: 图片 1 路径
[img2_path]: 图片 2 路径

testMassiveCompare（大规模 1:N 比对工具）

Usage: testMassiveCompare [model_path] [num_threads] [accuracy_mode] [size]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[accuracy_mode]: 精度模式 0:普通精度 1:高精度(速度稍慢) 2:实时模式(精度稍低)
[size]: 1:N 规模,N 值

testPoseImage（人脸 68 关键点和 3D 姿态角估计工具）

Usage: testPoseImage [model_path] [num_threads] [min_facesize] [img_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)

[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[img_path]: 图片路径

testInfrared（测试红外活体检测）

Usage: testInfrared [model_path] [img_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[img_path]: 图片路径

testRgbantiCamera(测试单目活体检测)

Usage: tesRgbantiCamera [model_path] [pat_threshold] [pat_count] [video_width] [video_height]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[pat_threshold]: pat 阈值
[pat_count]: pat 数量
[video_width]: 摄像头分辨率宽
[video_height]: 摄像头分辨率高

testTrackCamera（人脸跟踪摄像头工具)

Usage: testTrackCamera [model_path] [max_age] [video_width] [video_height]
[model_path]:模型目录(dface_sdk/model/normal_encrypt)
[max_age]: 被跟踪物体的最大过期次数
[video_width]: 摄像头分辨率宽
[video_height]: 摄像头分辨率高

testDatabase（人脸数据库测试工具)

Usage: testDatabase

2.3 测试源码编译

工具	描述
编译器	arm-linux-gnueabihf-g++/aarch64-linux-g++ 版本号 4.9 以上，支持 c++11 标准。
构建工具	cmake, 版本 3.6 以上。
Ide	vim clion qtcreator codeblocks
标准库	libstdc++6 libc6

链接器	ld-linux-armhf.so.3

DFace SDK 利用 CMake 相关文件快速构建测试项目，测试源码位于 example 目录。由于是 ARM 架构，因此一般在宿主机上执行交叉编译，DFace SDK 提供了 arm 交叉编译的 toolchain。

交叉编译

1. 安装交叉编译环境

sudo apt-get install gcc-arm-linux-gnueabihf

sudo apt-get install g++-arm-linux-gnueabihf

如果是 arm 32 位架构(armv7),还需要安装 32 位的库

sudo apt-get install ia32-libs

sudo apt-get install lib32ncurses5 lib32z1

2. 创建编译目录

进入 dface_sdk/example/cpp/demo_cmake_simple 目录下的 demo 工程

mkdir build & cd build

3. 执行 cmake

cmake .. -DCMAKE_TOOLCHAIN_FILE={dface_sdk}/arm-gnueabi.toolchain.cmake

注意: 如果需要 opencv，可以加上-DOpenCV_DIR={dface_sdk}/redistribute/share/OpenCV。

可以增加优化选项 -DUSE_NEON=ON -DCMAKE_CXX_FLAGS=" -mfpu=neon"

4. 构建测试项目

make -j 4

2.4 QT 编译测试

我们也提供了 QT 的 demo 源码，直接用 QTCreator 打开 **dface_sdk/example/cpp/demo_qt 项目即可编译测试。**

ARM 终端设备运行

参考 2.1 设置环境变量，把交叉编译完的可执行文件拷贝到 ARM 终端设备运行。

注意: Linux arm 由于加密狗的原因需要 root 权限执行。设置加密狗参考 2.4。

2.4 Sentinel 硬件加密狗设置

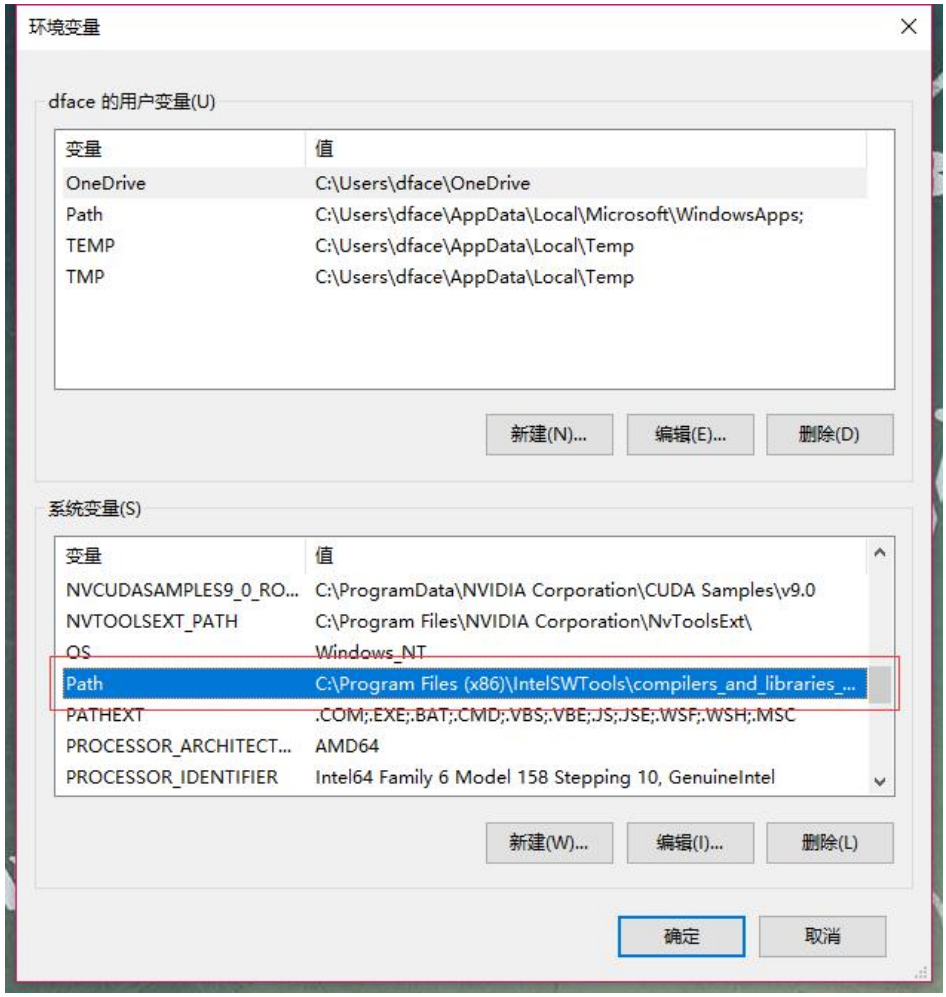
Linux ARM 的 sentinel 的加密狗需要手动被引导，请将 key/Sentinel/driver/linux_arm 目录下的所有文件拷贝至 /etc/udev/rules.d 目录。root 权限运行。

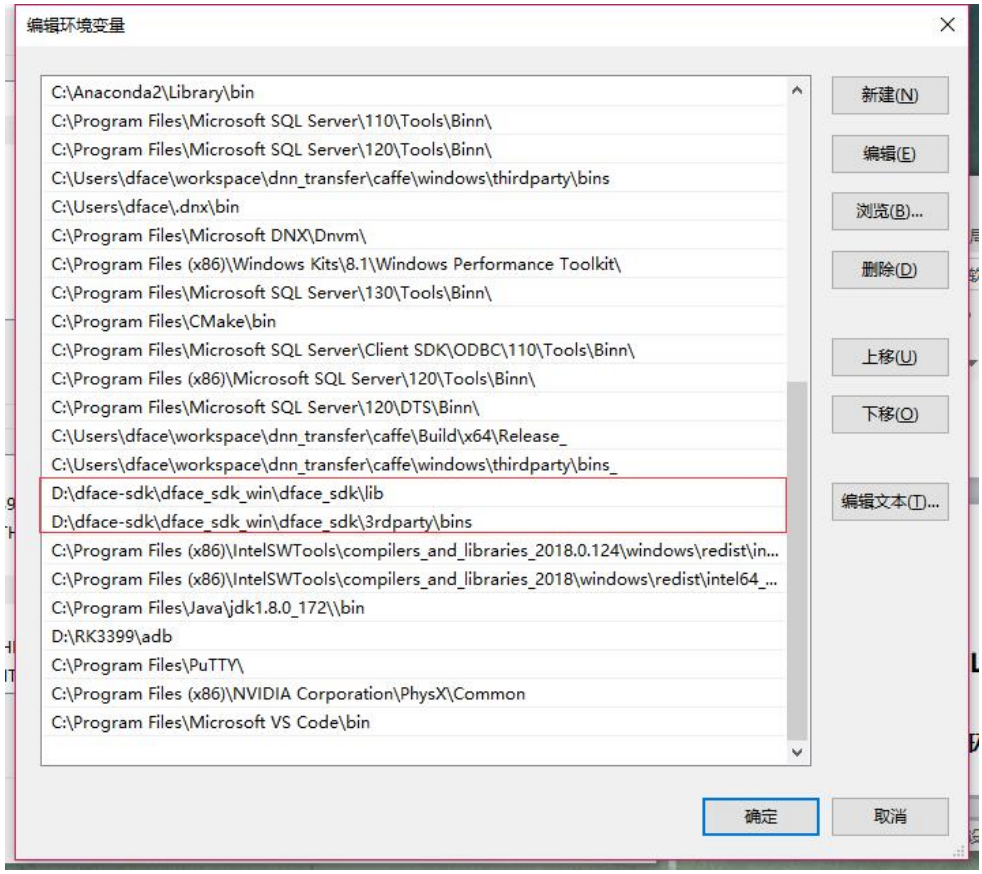
3.0 Windows(64 位)系统安装

3.1 环境变量设置

把 DFace SDK 目录下的 lib 目录 和 3rdparty/bins 目录 添加到 PATH 环境变量

电脑->属性->高级系统设置->环境变量->Path(编辑并添加 dface_sdk 库目录)


环境变量
dface 的用户变量(U)
变量
值
OneDrive
C:\Users\dface\OneDrive
Path
C:\Users\dface\AppData\Local\Microsoft\WindowsApps;
TEMP
C:\Users\dface\AppData\Local\Temp
TMP
C:\Users\dface\AppData\Local\Temp
新建(N)...
编辑(E)...
删除(D)
系统变量(S)
变量
值
NVCUDASAMPLES9_0_RO...
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v9.0
NVTOOLSEXT_PATH
C:\Program Files\NVIDIA Corporation\NvToolsExt\
OS
Windows_NT
Path
C:\Program Files (x86)\IntelSWTools\compilers_and_libraries_...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECT...
AMD64
PROCESSOR_IDENTIFIER
Intel64 Family 6 Model 158 Stepping 10, GenuineIntel
新建(W)...
编辑(I)...
删除(L)
确定
取消



3.2 工具包使用

DFace SDK 在 tools 目录下提供了 6 个工具可以快速在目标机器展开人脸识别类测试。使用 --help 参数可以查看具体使用说明。

testDetectImage.exe（人脸检测图片工具）

Usage: testDetectImage.exe [model_path] [num_threads] [min_facesize] [img_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[img_path]: 图片路径

testDetectCamera.exe（人脸检测摄像头工具）

Usage: testDetectCamera.exe [model_path] [num_threads] [min_facesize] [video_width]

[video_height]
[model_path]:模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[video_width]: 摄像头分辨率宽
[video_height]: 摄像头分辨率高

testDetectVideo.exe（人脸检测视频工具）

Usage: testDetectVideo.exe [model_path] [num_threads] [min_facesize] [video_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[video_path]: 视频路径

testRecognize.exe（人脸识别工具）

Usage: testRecognize.exe [model_path] [num_threads] [min_facesize] [accuracy_mode] [img1_path] [img2_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[accuracy_mode]: 精度模式 0:普通精度 1:高精度(速度稍慢) 2:实时模式(精度稍低)
[img1_path]: 图片 1 路径
[img2_path]: 图片 2 路径

testMassiveCompare.exe（大规模 1:N 比对工具）

Usage: testMassiveCompare.exe [model_path] [num_threads] [accuracy_mode] [size]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[accuracy_mode]: 精度模式 0:普通精度 1:高精度(速度稍慢) 2:实时模式(精度稍低)
[size]: 1:N 规模,N 值

testPoseImage（人脸 68 关键点和 3D 姿态角估计工具）

Usage: testPoseImage.exe [model_path] [num_threads] [min_facesize] [img_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[num_threads]: 并行运算线程数(1,2,4,8)
[min_facesize]: 需要被检测的最小人脸尺寸(40,60,80,120,200...)
[img_path]: 图片路径

testInfrared（测试红外活体检测）

Usage: testInfrared.exe [model_path] [img_path]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[img_path]: 图片路径

testRgbantiCamera.exe(测试单目活体检测)

Usage: tesRgbantiCamera [model_path] [pat_threshold] [pat_count] [video_width] [video_height]
[model_path]: 模型目录(dface_sdk/model/normal_encrypt)
[pat_threshold]: pat 阈值
[pat_count]: pat 数量
[video_width]: 摄像头分辨率宽
[video_height]: 摄像头分辨率高

testTrackCamera.exe（人脸跟踪摄像头工具)

Usage: testTrackCamera [model_path] [max_age] [video_width] [video_height]
[model_path]:模型目录(dface_sdk/model/normal_encrypt)
[max_age]: 被跟踪物体的最大过期次数
[video_width]: 摄像头分辨率宽
[video_height]: 摄像头分辨率高

testDatabase.exe（人脸数据库测试工具)

Usage: testDatabase

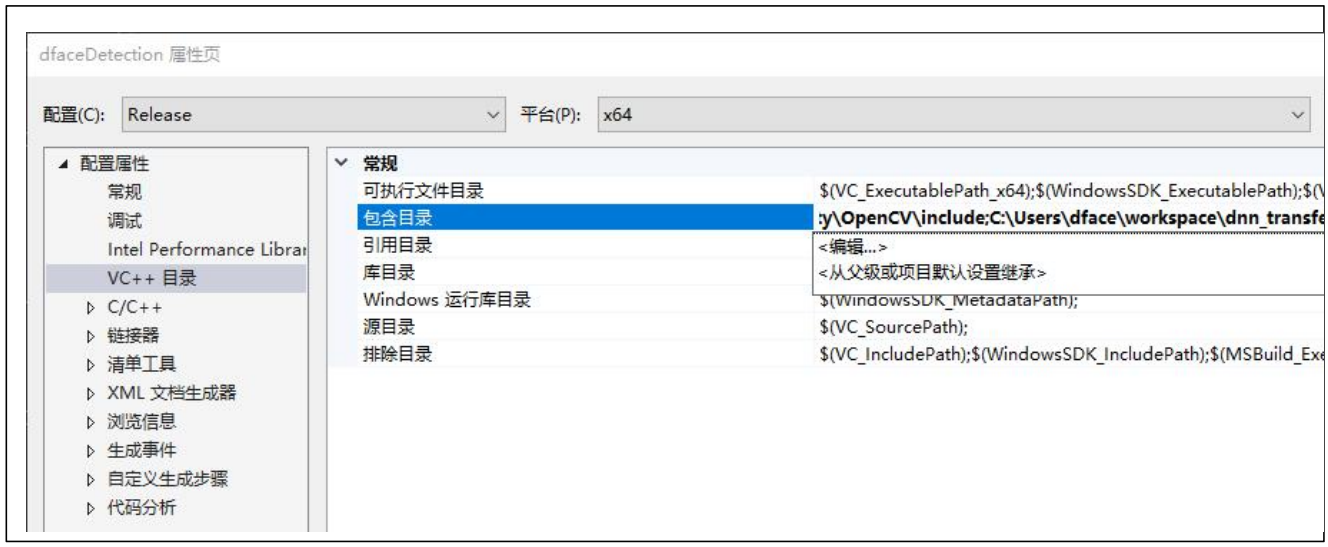
3.3 测试源码编译

工具	描述
编译器	Cl(V140) 版本号 V140 以上，需要支持 c++11 标准。
构建工具	Visual studio 2015 以上版本
Ide	Visual studio 2015 以上版本
标准库	
链接器	

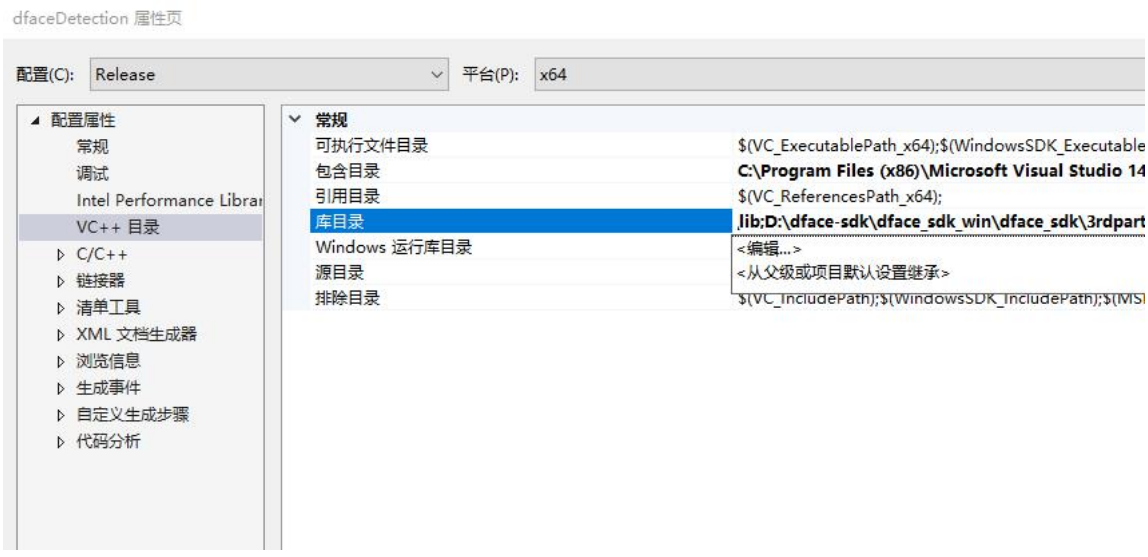
1. 我们直接提供了 visual studio 2015 的测试工程(**dface_sdk\example\cpp\demo_vs2015**)，用户打开工程即可。

2. **(非必须)**打开 visual studio 项目属性 ,VC++目录->包含目录->编辑 添加 DFace SDK

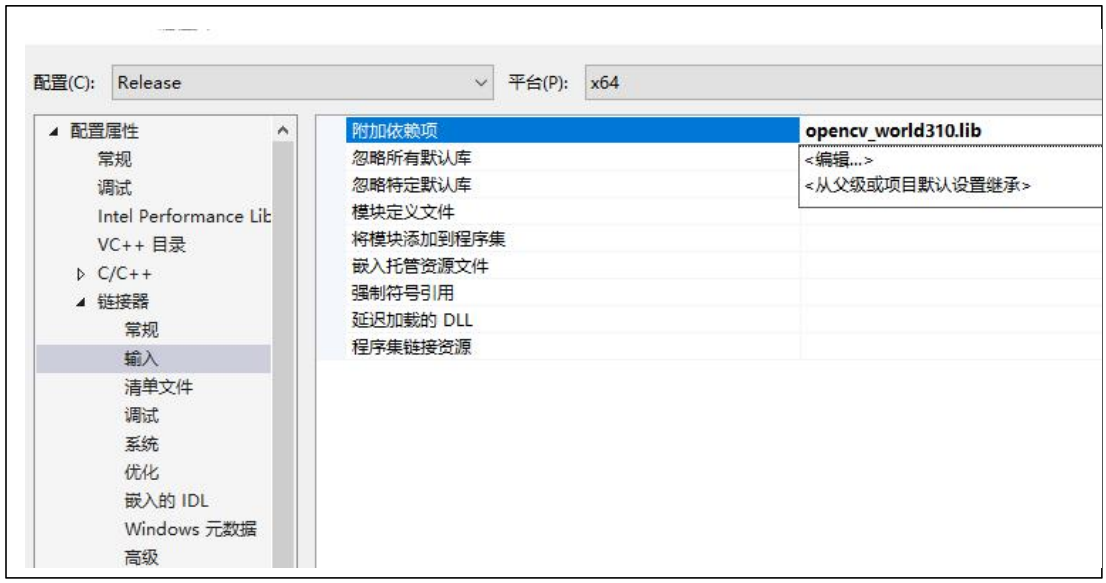
目录下的 include 目录和 3rdparty\OpenCV\include 目录



3. **(非必须)**打开 visual studio 项目属性 ,VC++目录->库目录->编辑 把 Dface SDK 目录下的 3rdparty\OpenCV\x64\vc14\lib 目录添加到库目录



4. (非必须)打开 visual studio 项目属性，连接器->输入->附加依赖项->编辑 添加 opencv_world310.lib


配置(C): Release
平台(P): x64
配置属性
常规
调试
Intel Performance Lib
VC++ 目录
C/C++
链接器
常规
输入
清单文件
调试
系统
优化
嵌入的 IDL
Windows 元数据
高级
附加依赖项
opencv_world310.lib
忽略所有默认库
<编辑...>
忽略特定默认库
<从父级或项目默认设置继承>
模块定义文件
将模块添加到程序集
嵌入托管资源文件
强制符号引用
延迟加载的 DLL
程序集链接资源

4.0 Android 系统安装

4.1 环境变量设置

DFace SDK 安卓版本是静态库生成的，无需设置复杂的环境变量。

4.2 工具包使用

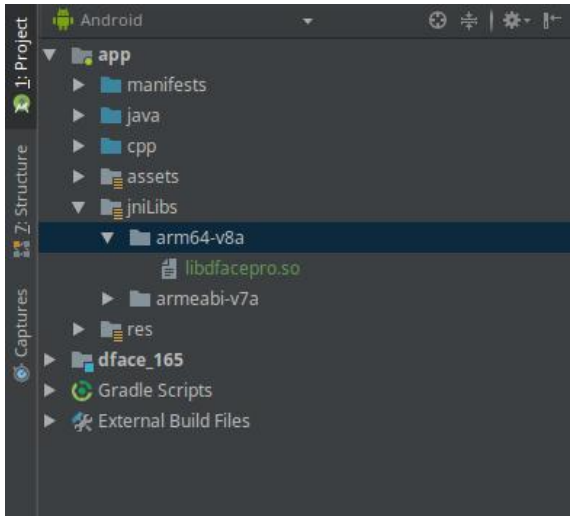
请安装我们的 APK，查看演示。

4.3 编译测试

工具	描述
编译器	Clang，最好选择 3.3 以上的版本，支持 c++11 标准。
构建工具	cmake 3.6 以上版本
Ide	Android studio/visual studio
标准库	libc++(clang 的标准库)
链接器	LLVM，最好选择 3.3 以上的版本，支持 c++11 标准。
NDK	我们的 NDK 选用 r16b 的版本

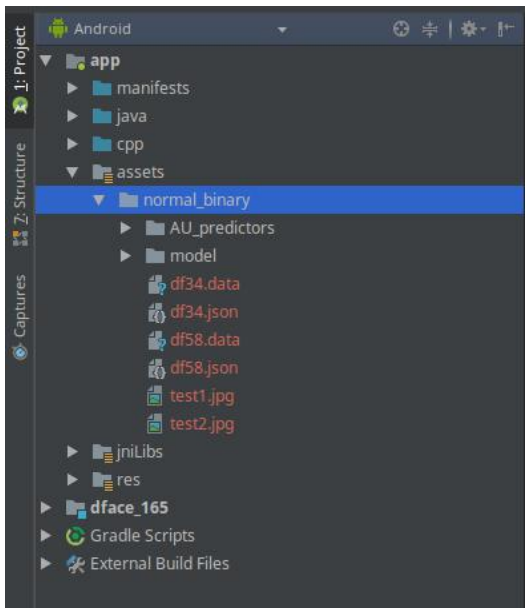
1. 我们在 example/android_studio 目录下直接提供了 Android Studio 的 Demo 工程项目。并且在 example/apk 目录下提供了测试安装包。

2. 把 dface_sdk 目录下的 lib/libdfacepro.so 库文件拷贝到 Android studio 项目的 jniLibs 目录下，例如 64 位 sdk 放在 jniLibs/arm64-v8a，32 位 sdk 放在 jniLibs/armeabi-v7a 等。



3. 导入 Jar 包，把 dface_sdk 目录下的 android/jar/dface.jar 导入 android studio。也可以将 dface_sdk 目录下的 example/java/src/main/java 源码拷贝到自己的项目中。

4. 把 dface_sdk 目录下的 model/normal_binary 目录拷贝至 android studio 目录下的 assets 目录。运行时请自行将该 assets 目录下的模型目录拷贝至设备的 SD 内存卡中。后续的通道初始化过程依赖于该模型目录。



5. 调用 SDK 之前需要动态加载 libdfacepro.so libdfacecommon.so 库文件。

例如:

```
static {
        System.loadLibrary("dfacepro");
        System.loadLibrary("dfacecommon")
    }
```

6. 具体调用 SDK 请参考开发文档, dface_sdk/doc 目录。

5.0 ROS(机器人操作)系统安装

持续更新中