CONFIDENTIAL

Sony Depthsensing Solutions

Modern C++ API project Skelton

SUMMARY

- **Project Structure**
- **Get Started**
- **Create Packages**
- **Design of API project**

Folder Structure

```
C:.
    cmakelists.txt
    open_source_software_licenses.txt

---cmake
        Findvitalslib.cmake

---config
        vitalslib.config.xml

---deps
    ---iu456
    ---mlpack
    ---opencv
---samples
    ---vitalslib_sample
            CMakeLists.txt
            main.cpp
            vitalslib_sample.cpp
            vitalslib_sample.h

---scripts
        ALL_BUILD.bat
        build.py

---tests
        catch.hpp
        CMakeLists.txt
        main.cpp
        test_example.cpp

---vitalslib_lib
    |   CMakeLists.txt
    |   version.h
    |   vitalslib_api.cpp
    |   vitalslib_api.h
    |   vitalslib_platform.h
    |   vitalslib_types.h
    |
    ---core
            pipeline.h
```

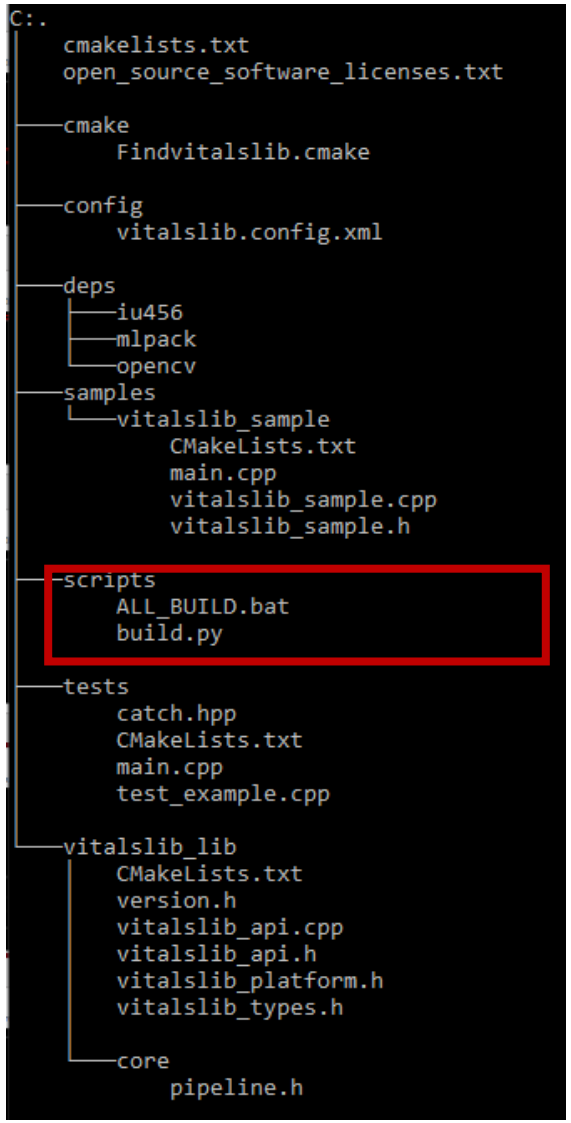
Intro

This project give an example of modern C++ API project Cmake based

For the example this project is call *VITALSLIB*.

Some empty folder dependency have been added (mlpack, opencv, …) in order to get the logics of the structure.

Get Started (1/3)



Prerequire

- Cmake 3.3 or more
- Python 3
- Visual Studio 2015 or more

Build solution

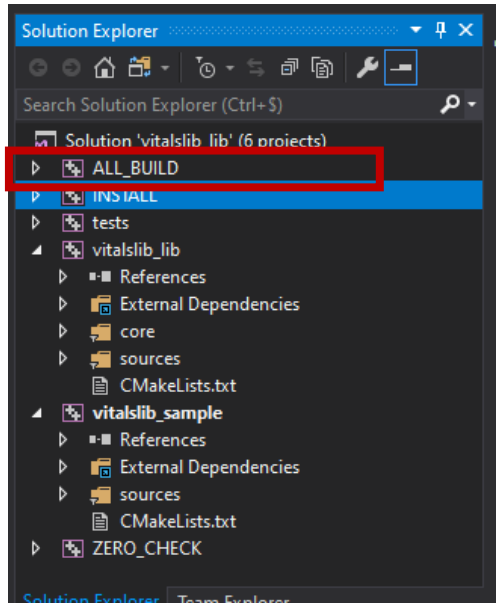
-Double click on scripts/ALL_BUILD.bat. A folder _build is created and contain the VS solution of the project

Get Started (2/3)

```
C:.
    cmakelists.txt
    open_source_software_licenses.txt

├───cmake
        Findvitalslib.cmake

├───config
        vitalslib.config.xml

├───deps
    ├───iu456
    ├───mlpack
    └───opencv
├───samples
    └───vitalslib_sample
            CMakeLists.txt
            main.cpp
            vitalslib_sample.cpp
            vitalslib_sample.h

├───scripts
        ALL_BUILD.bat
        build.py

├───tests
        catch.hpp
        CMakeLists.txt
        main.cpp
        test_example.cpp

└───vitalslib_lib
    │   CMakeLists.txt
    │   version.h
    │   vitalslib_api.cpp
    │   vitalslib_api.h
    │   vitalslib_platform.h
    │   vitalslib_types.h
    │
    └───core
            pipeline.h
```

Compile Solution

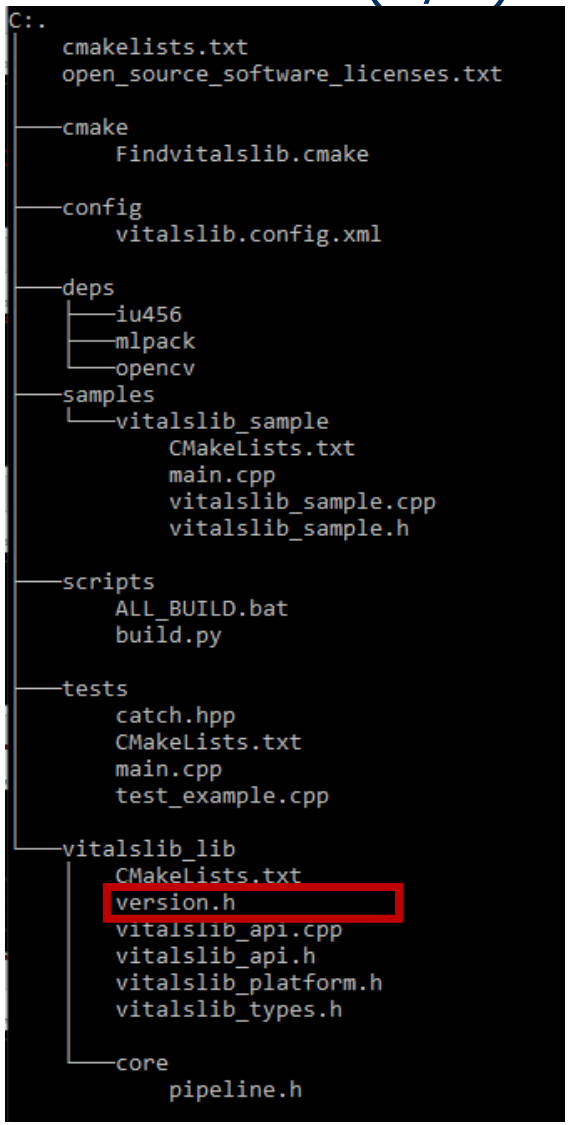
- In _build open open vitalslib_lib.sln. The solution is splited in 3 VS project:
  - vitalslib_lib : The dummy API project that generate a .dll
  - vitalslib_sample : A dummy app project that will generate an .exe using vitalslib_lib.dll
  - tests : A project in order to test vitalslib_lib pipline units generating a .exe
- Execute ALL_BUILD to compile all the solution (Debug and/or Release). If the compilation is success full a folder _output generated



NB:
This structure of project is made by cmakelists.txt in the root of the project:

```
#-------------------------------------------------------------------------------
#-Subprojects
#-------------------------------------------------------------------------------
add_subdirectory(vitalslib_lib)
add_subdirectory(samples/vitalslib_sample)
add_subdirectory(tests)
```

Get Started (3/3)



Execute Sample App

- In _output/Windows/<Debug/Release>/ open a cmd
- Execute vitalslib_sampled.exe OR vitalslib_sample.exe as:

```
Windows\Release>vitalslib_sample.exe "HELLO WORLD!"
Argument input : HELLO WORLD!
vitalslib version : 0.1.0.0
Result output : 0
```

Observation

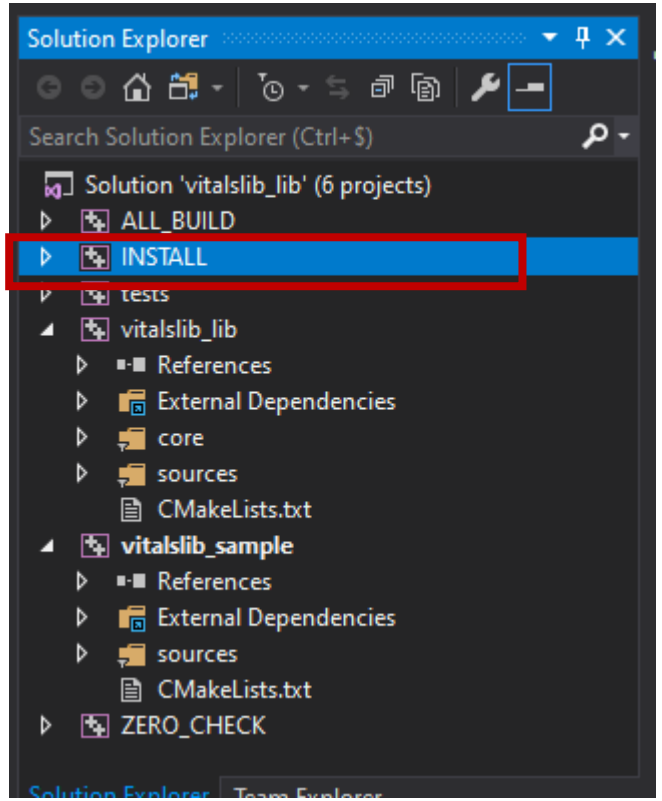
- Version of the dll is encapsulated into it, and accessible from an API function. The version is the same in version.h from vitalslib_lib folder.

Create Packages

```
C:.
├───vitalslib_lib
│   ├───bin
│   │   └───Windows
│   │           vitalslib_lib.dll
│   │
│   ├───cmake
│   │       Findvitalslib.cmake
│   │
│   ├───config
│   │       vitalslib.config.xml
│   │
│   ├───include
│   │       vitalslib_api.h
│   │       vitalslib_platform.h
│   │       vitalslib_types.h
│   │
│   └───lib
│       └───Windows
│               vitalslib_lib.lib
│
└───vitalslib_sample
    ├───deps
    │   └───vitalslib_lib
    │       ├───bin
    │       │   └───Windows
    │       │           vitalslib_lib.dll
    │       │
    │       ├───cmake
    │       │       Findvitalslib.cmake
    │       │
    │       ├───config
    │       │       vitalslib.config.xml
    │       │
    │       ├───include
    │       │       vitalslib_api.h
    │       │       vitalslib_platform.h
    │       │       vitalslib_types.h
    │       │
    │       └───lib
    │           └───Windows
    │                   vitalslib_lib.lib
    │
    └───src
            main.cpp
            vitalslib_sample.cpp
            vitalslib_sample.h
```

INSTALL API

- In Visual Studio build INSTALL. A new folder call dist is generated.
- As we can observe, 2 packages (delivrable) have been created:
  - The first one is the dummy library project (vitalslib_lib), that a client can use and access with the API, the .lib and the .dll
  - The second one the dummy sample app (vitalslib_sample), that use as dependency the dummy API (Includes, lib and dll).



Design of API project

```
C:.
    cmakelists.txt
    open_source_software_licenses.txt

---cmake
        Findvitalslib.cmake

---config
        vitalslib.config.xml

---deps
    ---iu456
    ---mlpack
    ---opencv
---samples
    ---vitalslib_sample
            CMakeLists.txt
            main.cpp
            vitalslib_sample.cpp
            vitalslib_sample.h

---scripts
        ALL_BUILD.bat
        build.py

---tests
        catch.hpp
        CMakeLists.txt
        main.cpp
        test_example.cpp

---vitalslib_lib
    |   CMakeLists.txt
    |   version.h
    |   vitalslib_api.cpp
    |   vitalslib_api.h
    |   vitalslib_platform.h
    |   vitalslib_types.h
    |
    ---core
            pipeline.h
```

CMakeLists.txt

In term of design, your API project is splited into 3 different type of source file:

- The Public Source Files: the once that will be part of the future include of the package
- The Core files: the once where the processing, the logics and the algorithm will be contained.
- The Private Source Files: the once that are not processing and glue the logics towards the Core files

```
#.-------------------------------------------------------------------------------
#.Sources
#.-------------------------------------------------------------------------------

set(vitalslib_PUBLIC_SRCFILES.
--->vitalslib_api.h
--->vitalslib_platform.h
--->vitalslib_types.h
)

set(vitalslib_PRIVATE_SRCFILES.
--->vitalslib_api.cpp
--->version.h)
--->
set(vitalslib_CORE.
--->core/pipeline.h)
```

CONFIDENTIAL

Sony Depthsensing Solutions