MYNT EYE D SDK Documentation

Release 1.7.3

MYNTAI

Mar 25, 2019

Contents

1 MYNT® EYE PRODUCT INTRODUCTION **1**
- 1.1 Product Description . . . 1
- 1.2 Product Surface . . . 2
- 1.3 D1000-120/Color specification . . . 4
- 1.4 D1000-50/Color specification . . . 5
- 1.5 Support Resolutions . . . 6
- 1.6 IMU Coordinata System . . . 6

2 MYNT® EYE SDK INSTALLATION **9**
- 2.1 Change log . . . 9
- 2.2 Supported Platforms . . . 9
- 2.3 Quick Start Guide for Linux . . . 10
- 2.4 Quick Start Guide for Windows . . . 13
- 2.5 Windows EXE Installation . . . 17
- 2.6 ROS Installation . . . 18
- 2.7 ROS Usage . . . 20

3 MYNT® EYE SDK SAMPLES **23**
- 3.1 Get camera image . . . 23
- 3.2 Get camera image(Compatible with USB2.0) . . . 23
- 3.3 Get depth image . . . 24
- 3.4 Get point image . . . 25
- 3.5 Get IMU data . . . 25
- 3.6 Get data from callbacks . . . 26
- 3.7 Get different types of image by options . . . 27
- 3.8 Get image calibration parameters . . . 28
- 3.9 Get IMU calibration parameters . . . 28
- 3.10 Set open parameters . . . 29
- 3.11 Camera control parameters API . . . 32

4 MYNT® EYE SDK TOOLS **35**
- 4.1 Analyze IMU data . . . 35
- 4.2 Analyze time stamps . . . 37
- 4.3 Record data sets . . . 39
- 4.4 Save device infomation and parameters . . . 40
- 4.5 Write IMU parameters . . . 40
- 4.6 Update HID Firmware . . . 41

4.7 Update Camera Firmware . . . 41

5 PROJECT DEMOS **43**
5.1 How to use SDK with Visual Studio 2017 . . . 43
5.2 How to use SDK with Qt Creator . . . 51
5.3 How to use SDK with CMake . . . 63

6 SLAM **67**
6.1 How to use in VINS-Mono . . . 67
6.2 How to use in ORB_SLAM2 . . . 68
6.3 How to use in OKVIS . . . 69
6.4 How to use in VIORB . . . 70
6.5 How to use in VINS-Fusion . . . 71

7 API REFERENCE **73**
7.1 Camera . . . 73
7.2 Device . . . 77
7.3 Enums . . . 79
7.4 Types . . . 83
7.5 Utils . . . 87

CHAPTER 1

MYNT® EYE PRODUCT INTRODUCTION

1.1 Product Description

The MYNT Depth utilizes the camera and the motion sensor to provide visually accurate SLAM results with higher precision, lower cost, simpler layout, along with the ability to achieve face and object recognition. The concept of combining binocular and IMU is the leading-edge technology in the current SLAM industry.The depth version of the product has a built-in depth calculation chip that can output depth images without the host computer. At the same time, the product is equipped with leading hardware solutions such as IR active light, IMU six-axis, hardware-level frame synchronization, global shutter, etc., up to 720p/60fps (120°FOV version) of synchronous image information, the recognition distance can reach 15m (50 °FOV version), accuracy up to millimeters (50° FOV version).

Using camera techniques such as frame synchronization, automatic exposure, and white balance control, the MYNT EYE Depth can produce synchronized image sources with high-precision, which decreases the difficulty of algorithms development, thus increasing efficiency. The Depth comes with six-axis sensor(IMU) and an infrared active light detector (IR). Among them, the six-axis sensor(IMU) can provide complementarity and correction of data from the visual positioning algorithms, and is suitable for visual inertial odometry(VIO) algorithms research to help improve the positioning accuracy. The infrared active light detector (IR) can help solve the problem of identification of objects such as indoor white walls and non-textured objects, as well as enhance the accuracy of image source recognition.The Binocular+IMU scheme provides accurate six-axis complementary data for VSLAM applications and is more accurate and robust than other single solutions. In addition, MYNT EYE Depth also provides a rich SDK interface and VSLAM open source project support, which can help customers quickly integrate solutions, accelerate product development process, and achieve rapid productization and implementation.

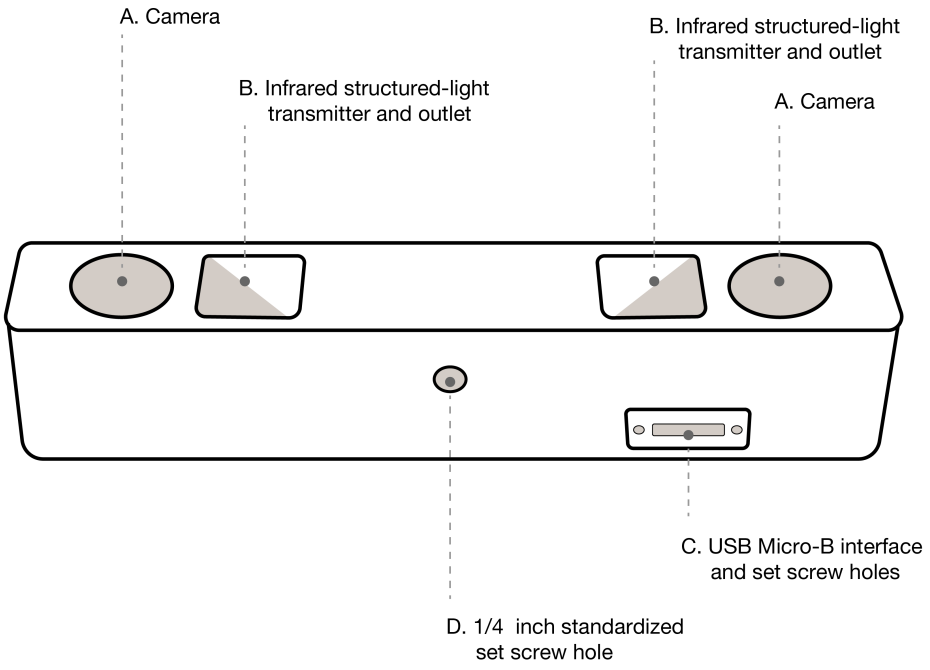
As a hardware product for in-depth research and development of stereo vision computing applications, MYNT EYE Depth can be widely used in the field of visual positioning navigation (vSLAM), including visual real-time positioning navigation system of driverless vehicle and robots, visual positioning system of UAV, obstacle avoidance navigation system for driverless Vehicle, Augmented Reality (AR), Virtual Reality (VR), etc. At the same time, it can be used in field of Visual recognition, including Stereoscopic face recognition, three-dimensional object recognition, space motion tracking, three-dimensional gestures and somatosensory recognition. And of course, you can use it for measurement which includes assisted driving system (ADAS), binocular volume calculation, industrial visual screening, etc.

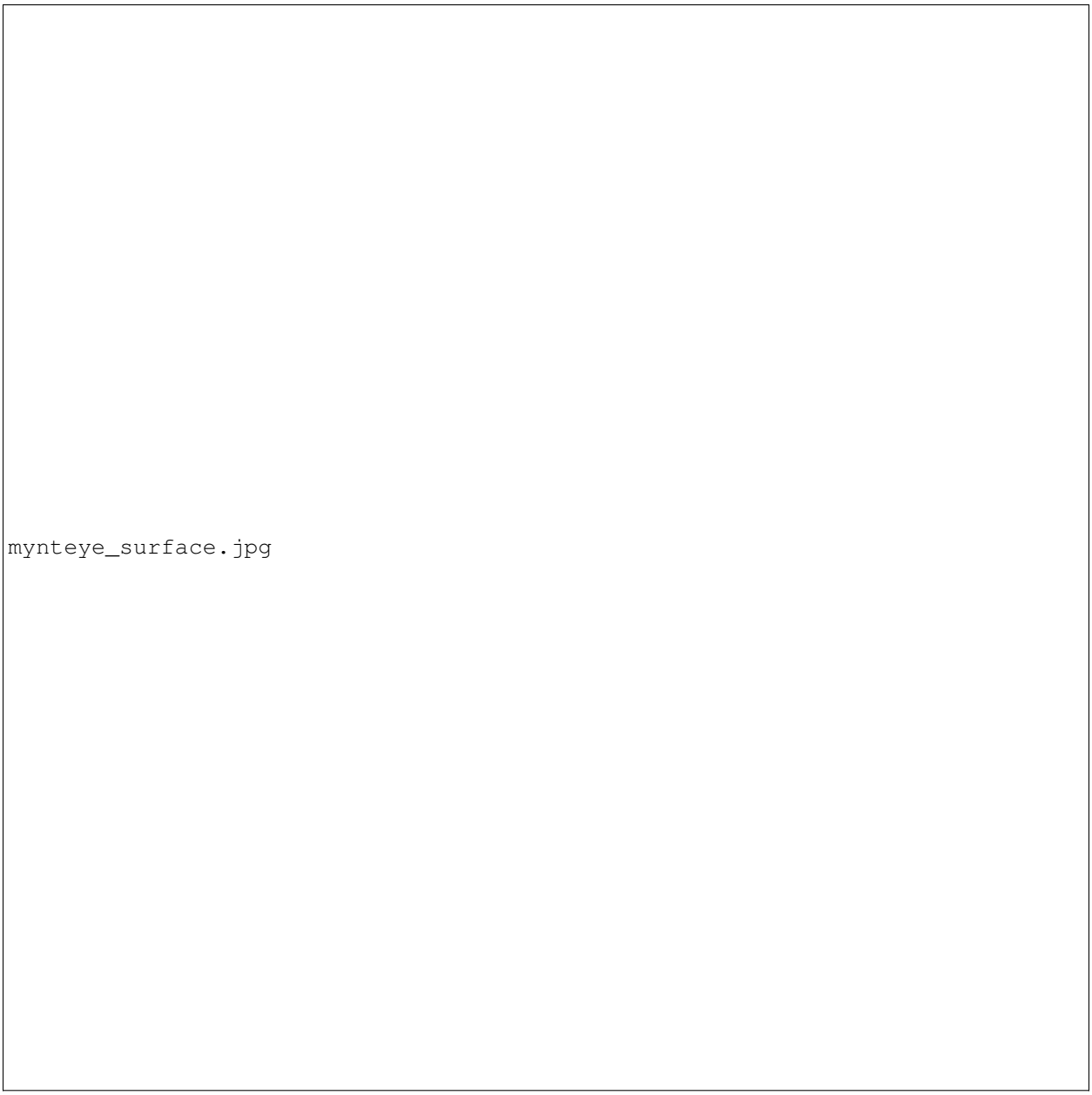
In order to ensure the quality of the output data of the camera products, we have calibrated the binocular and IMU. The product has passed various hardware stability tests, such as high temperature and humidity continuous work

and operation, low-temperature dynamic aging, high-temperature operation, low-temperature storage, whole-machine thermal shock, sinusoidal vibration and random vibration tests to ensure the stability and reliability of the product. In addition to the research and development of products and technologies, it can also be directly applied to mass production, accelerating the process from R&D to productization.

1.2 Product Surface

Shell(mm)	PCBA board(mm)
165x31.5x29.6	149x24





A. Camera:please pay attention to protect the camera sensor lenses, to avoid imaging quality degradation.

B. Infrared structured-light transmitter and outlet: the infrared structured-light can effectively solve the problem associated with the visual positioning calculations of white wall non-textured object(For non-IR version, the outlet is reserved but there is no internal structured-light emitter).

C. USB Micro-B interface and set screw holes: during usage, plug in the USB Micro-B cable and secure it by fastening the set screws to avoid damage to the interface and to ensure stability in connection.

D. ¼ inch standardized set screw hole: for fixing the stereo camera to tripods or other devices.

1.3 D1000-120/Color specification

1.3.1 Product parameters

Model	D1000-IR-120/Color
Size	PCB dimension:150x24mm, Total dimension:165x31.5x30.12mm
Frame Rate	Up to 60FPS
Resolution	2560x720;1280x480
Depth Resolution	On chip 1280x720 640x480
Pixel Size	3.75x3.75μm
Baseline	120.0mm
Camera Lens	Replacable Standard M12
Visual Angle	D:121° H:105° V:58°
Focal Length	2.45mm
IR Support	Yes
IR detectable range	3m
Color Mode	Color
Working Distance	0.37-8m
Scanning Mode	Global Shutter
Power	1.9~3.5W@5V DC from USB
Output data format	YUYV/MJPG
Data transfer Interface	USB2.0/3.0
Weight	184g
UVC MODE	Yes

1.3.2 Environment

Operating Temperature	-10°C~60°C
Storage Temperature	-20°C~70°C

1.4 D1000-50/Color specification

1.4.1 Product parameters

Model	D1000-50/Color
Size	PCB dimension:150x24mm, Total dimension:165x31.5x30.12mm
Frame Rate	Up to 60FPS
Resolution	2560x720;1280x480
Depth Resolution	On chip 1280x720 640x480
Pixel Size	3.75x3.75μm
Baseline	120.0mm
Camera Lens	Replacable Standard M12
Visual Angle	D:70° H:64° V:38°
Focal Length	2.45mm
IR Support	Yes
IR detectable range	3m
Color Mode	Color
Working Distance	0.52-15m
Scanning Mode	Global Shutter
Power	1.8W@5V DC from USB
Output data format	YUYV/MJPG
Data transfer Interface	USB2.0/3.0
Weight	152g
UVC MODE	Yes

1.4.2 Environment

Operating Temperature	-10°C~60°C
Storage Temperature	-20°C~70°C

1.5 Support Resolutions

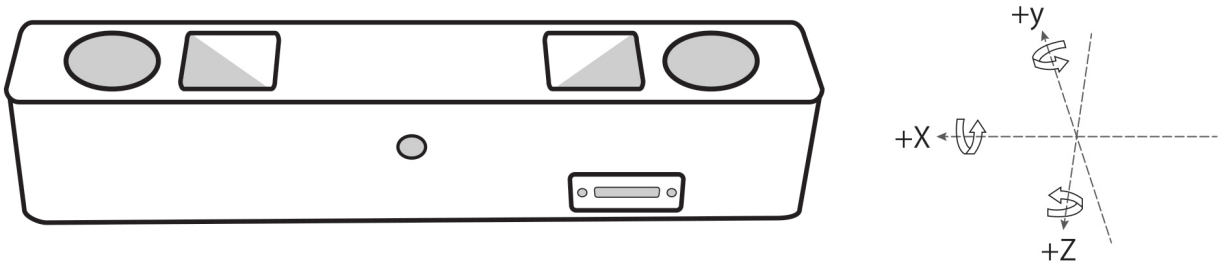
mode	interface	color resolution	color fps	depth resolution	depth fps
L'+D	USB3.0	1280x720	60/30/20/10	1280x720	60/30/20/10
L'+D	USB3.0	640x480	60/30	640x480	60/30
L'+R'+D	USB3.0	2560x720	30	1280x720	30
L'+R'+D	USB3.0	1280x480	60/30	640x480	60/30
L+D	USB3.0	1280x720	60/30/20/10	1280x720	60/30/20/10
L+D	USB3.0	640x480	60/30	640x480	60/30
L+R+D	USB3.0	2560x720	30	1280x720	30
L+R+D	USB3.0	1280x480	60/30	640x480	60/30
L+R	USB3.0	2560x720	30	not open	null
L'+R'	USB3.0	2560x720	30	not open	null
D	USB3.0	not open	null	1280x720	60/30
D	USB3.0	not open	null	640x480	60/30
L+R	USB2.0	2560x720	5	not open	null
L'+R'	USB2.0	2560x720	5	not open	null
L+R	USB2.0	1280x480	15	not open	null
L'+R'	USB2.0	1280x480	15	not open	null
L'+D	USB2.0	1280x720	5	640x720	5
L'+D	USB2.0	640x480	15	320x480	15
L+D	USB2.0	1280x720	5	640x720	5
L+D	USB2.0	640x480	15	320x480	15
L'	USB2.0	1280x720	5	not open	null
L	USB2.0	1280x720	5	not open	null
D	USB2.0	not open	null	640x720	5
D	USB2.0	not open	null	320x480	15
L+R	USB2.0/MJPG	2560x720	5	not open	null
L+R	USB2.0/MJPG	1280x480	15	not open	null
L	USB2.0/MJPG	1280x720	5	not open	null

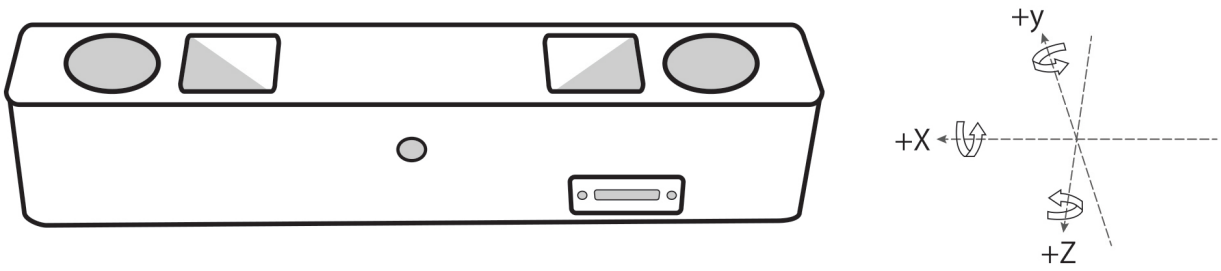
Note: L'=left rectify image, L=left image,R'=right rectify image, R=right image, D=depth image

In IR Depth Only mode, framerate only support 15fps and 30fps.

1.6 IMU Coordinata System

IMU coordinate system is right-handed,the axis directions are as follows:


+y
+X
+Z


+y
+X
+Z

CHAPTER 2

MYNT® EYE SDK INSTALLATION

2.1 Change log

1. Add support for external sensors (ultrasonic sensors, GPS).
2. Depth images and color images are synchronized by frame id.
3. Add sample which compatible with USB2.0.
4. Fix the problem that the frame rate of camera info released by left and right eyes under ROS is twice the normal value.
5. Document optimization.

2.2 Supported Platforms

SDK is built on CMake and can be used cross multiple platforms such as Linux, WIndows,etc.We provide two installation modes:Download pack file and install, Compile and install from source code.

These are the platforms that can be used:

```
* Windows 10
* Ubuntu 18.04/16.04
* Jetson TX1 TX2 Xavier
* RK3399
```

Tip: Ubuntu only support source installation mode. Only supports 64 bit systems.

Warning: Due to the requirement of hardware transmission rate, please use the USB 3 interface. In addition, virtual machines have USB driver compatibility problems, thus they are not recommended.

2.3 Quick Start Guide for Linux

2.3.1 1. Install SDK dependencies

1.1 Install OpenCV

If you have installed opencv already or you want use it in ROS, you can skip this part.

1.1.1 Install OpenCV with apt or compile (Choose one)

1.1.1.1 Install OpenCV with apt (Recommend)

```
sudo apt-get install libopencv-dev
```

1.1.1.2 Install OpenCV by Compile

To build and install Opencv, please refer to Installation in Linux

Alternatively, refer to the command below:

```
[compiler] sudo apt-get install build-essential
[required] sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev␣
↪libavformat-dev libswscale-dev
[optional] sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-
↪dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev
```

```
git clone https://github.com/opencv/opencv.git
cd opencv/
git checkout tags/3.4.5

cd opencv/
mkdir build
cd build/

cmake ..

make -j4
sudo make install
```

1.2 Install PCL for Point Cloud sample (Optional)

```
sudo apt-get install libpcl-dev libproj-dev libopenni2-dev libopenni-dev
```

1.3 Link libGL.so for TX1/TX2 compile bug (Optional)

```
sudo ln -sf /usr/lib/aarch64-linux-gnu/tegra/libGL.so /usr/lib/aarch64-linux-gnu/
↪libGL.so
```

2.3.2 2. Build SDK

```
git clone https://github.com/slightech/MYNT-EYE-D-SDK.git
cd MYNT-EYE-D-SDK
```

2.1 Init SDK

Note: Because of the problem of device permissions, you must reinsert the camera device after the command is executed and on the same computer, this operation only needs to be done once.

```
make init
```

2.2 Compile SDK

```
make all
```

2.3.3 3. Run Samples

Note: Open the rectified image by default (Run vio need to raw image, run depth or points cloud need to rectified image.)

1) get_image shows the left camera image and colorful depthmap (compatible with USB2.0)

```
./samples/_output/bin/get_image
```

2) get_stereo_image shows the left camera image and colorful depthmap

```
./samples/_output/bin/get_stereo_image
```

3) get_depth shows the left camera image, 16UC1 depthmap and depth value(mm) on mouse pointed pixal

```
./samples/_output/bin/get_depth
```

4) get_points shows the left camera image, 16UC1 depthmap and point cloud view

```
./samples/_output/bin/get_points
```

5) get_imu shows motion datas

```
./samples/_output/bin/get_imu
```

6) get_img_params show camera intrinsics and save in file

```
./samples/_output/bin/get_img_params
```

7) get_imu_params show imu intrinsics and save in file

```
./samples/_output/bin/get_imu_params
```

8) get_from_callbacks show image and imu data by callback

```
./samples/_output/bin/get_from_callbacks
```

9) get_all_with_options open device with different options

```
./samples/_output/bin/get_all_with_options
```

2.3.4 4 Install With OpenCV ROS

If you won't use ROS(The Robot Operating System), you can skip this part.

4.1 Install ROS Kinetic

```
cd ~
wget https://raw.githubusercontent.com/oroca/oroca-ros-pkg/master/ros_install.sh && \
chmod 755 ./ros_install.sh && bash ./ros_install.sh catkin_ws kinetic
```

Note: ROS Kinetic will install OpenCV, JPEG.

4.2 Build ROS Wrapper

```
make ros
```

Core:

```
roscore
```

RViz Display:

```
source ./wrappers/ros/devel/setup.bash
roslaunch mynteye_wrapper_d display.launch
```

Publish:

```
source ./wrappers/ros/devel/setup.bash
roslaunch mynteye_wrapper_d mynteye.launch
```

4.3 Build Beta Device ROS Wrapper

```
make ros
```

Core:

```
roscore
```

RViz Display:

```
source ./wrappers/beta_ros/devel/setup.bash
roslaunch mynteye_wrapper_d_beta display.launch
```

Publish:

```
source ./wrappers/beta_ros/devel/setup.bash
roslaunch mynteye_wrapper_d_beta mynteye.launch
```

Subscribe:

```
source ./wrappers/beta_ros/devel/setup.bash
rosrun mynteye_wrapper_d_beta mynteye_listener_d_beta
```

Subscribe:

```
source ./wrappers/beta_ros/devel/setup.bash
rosrun mynteye_wrapper_d_beta mynteye_listener_d_beta
```

2.3.5 5. Package

If you wanna package with specified OpenCV version:

```
cd <sdk>
make cleanall
export OpenCV_DIR=<install prefix>

export OpenCV_DIR=/usr/local
export OpenCV_DIR=$HOME/opencv-2.4.13.3
```

Packaging:

```
cd <sdk>  #local path of MYNT-EYE-D-SDK
make pkg
```

2.3.6 6. Clean

```
cd <sdk>   #local path of MYNT-EYE-D-SDK
make cleanall
make uninstall
```

2.4 Quick Start Guide for Windows

The following steps are how to install from source codes. If you wanna using prebuilt DLL, please see *Windows EXE Installation* .

2.4.1 1. Install Build Tools

1.1 Install Visual Studio

Download Visual Studio 2017 from https://visualstudio.microsoft.com/ and install

Tip: support Visual Studio 2015 and Visual Studio 2017.

1.2 Install CMake

Download CMake from https://cmake.org/ and install

1.3 Install MSYS2

1) Download MSYS2 from http://mirrors.ustc.edu.cn/msys2/distrib/x86_64/ and install
2) Add bin path to System PATH environment variable list

```
C:\msys64\usr\bin
```

3) Install make

```
pacman -Syu
pacman -S make
```

Finally, the CMD (Command Prompt) can run the following command:

```
>make --version
GNU Make 4.2.1
```

2.4.2 2. Install SDK dependencies

2.1 Install OpenCV

2.1.1 Install OpenCV with Pre-built Libraries (Recommend)

*For more details you can reference OpenCV offical document *

1) Go to OpenCV Sourceforge page http://sourceforge.net/projects/opencvlibrary/files/opencv-win/
2) Choose a build you want to use and download it. For example 3.4.2/opencv-3.4.2-vc14_vc15.exe
3) Make sure you have admin rights. Unpack the self-extracting archive
4) To finalize the installation, go to set the OpenCV environment variable and add it to the systems path

2.1.2 Set up environment variable

Start up a command window as admin and enter following command to add `OPENCV_DIR` environment variable:

Change the "D:OpenCV" to your opencv unpack path

```
setx -m OPENCV_DIR D:\OpenCV\Build\x64\vc14\lib     (suggested for Visual Studio 2015␣
↪- 64 bit Windows)
setx -m OPENCV_DIR D:\OpenCV\Build\x64\vc15\lib     (suggested for Visual Studio 2017␣
↪- 64 bit Windows)
```

Add OpenCV bin path to System PATH environment variable list

```
D:\OpenCV\Build\x64\vc14\bin      (suggested for Visual Studio 2015 - 64 bit Windows)
D:\OpenCV\Build\x64\vc15\bin      (suggested for Visual Studio 2017 - 64 bit Windows)
```

2.2 Install libjpeg-turbo

1) Download libjpeg-turbo from https://sourceforge.net/projects/libjpeg-turbo/files/ and install
2) Add bin path to System PATH environment variable list

```
C:\libjpeg-turbo64\bin
```

2.3 Install PCL for Point Cloud sample (Optional)

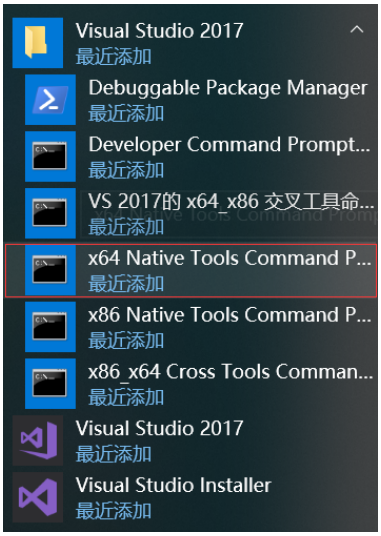
Download All-in-one installers (PCL + dependencies) from: https://github.com/PointCloudLibrary/pcl/releases

2.4.3 3. Build SDK

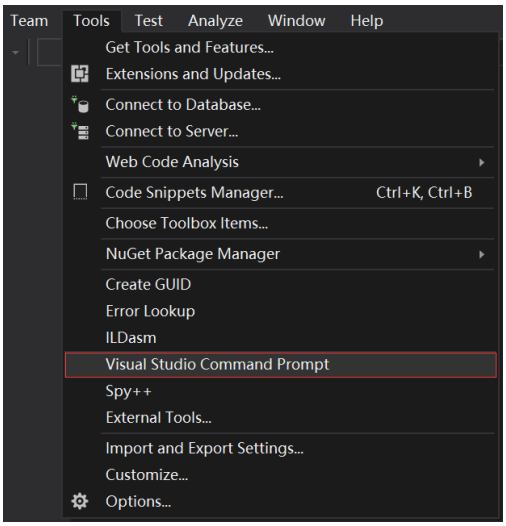
Open "x64 Native Tools Command Prompt for VS 2017"(VS 2017 x64) command shell

```
git clone https://github.com/slightech/MYNT-EYE-D-SDK.git
cd MYNT-EYE-D-SDK
make all
```

Tip: Visual Studio Command Prompt can be opened from the Start menu,



You can also open it from the Visual Studio Tools menu.



However, if you do not have the Visual Studio 2015 Tools menu, you can add one yourself.

Open Tools's External Tools... and Add the following:

Field	Value
Title	Visual Studio Command Prompt
Command	`C:\Windows\System32\cmd.exe`
Arguments	`/k "C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\Tools\VsDevCmd.bat"`
Initial Directory	`$(SolutionDir)`

2.4.4 4. Run Samples

Note: Open the rectified image by default (Run vio need to raw image, run depth or points cloud need to rectified image.)

1) get_image shows the left camera image and colorful depthmap (compatible with USB2.0)

```
.\samples\_output\bin\get_image.bat
```

2) get_stereo_image shows the left camera image and colorful depthmap

```
./samples/_output/bin/get_stereo_image.bat
```

3) get_depth shows the left camera image, 16UC1 depthmap and depth value(mm) on mouse pointed pixel

```
.\samples\_output\bin\get_depth.bat
```

4) get_points shows the left camera image, 16UC1 depthmap and point cloud view

```
.\samples\_output\bin\get_points.bat
```

5) get_imu shows motion datas

```
.\samples\_output\bin\get_imu
```

6) get_img_params show camera intrinsics and save in file

```
.\samples\_output\bin\get_img_params
```

7) get_imu_params show imu intrinsics and save in file

```
.\samples\_output\bin\get_imu_params
```

8) get_from_callbacks show image and imu data by callback

```
.\samples\_output\bin\get_from_callbacks
```

9) get_all_with_options open device with different options

```
.\samples\_output\bin\get_all_with_options
```

2.4.5 5. Clean

```
cd <sdk>  #local path of MYNT-EYE-D-SDK
make cleanall
```

2.5 Windows EXE Installation

Download here: mynteye-d-1.7.1-win-x64-opencv-3.4.3.exe Google Drive, Baidu Pan

After you install the win pack of SDK, there will be a shortcut to the SDK root directory on your desktop.

First, you should plug the MYNT® EYE camera in a USB 3.0 port.

Second, goto the "binsamples'" directory and click "get_image.exe" to run.

Finally, you will see the window that display the realtime frame of the camera.

2.5.1 Generate samples project of Visual Studio 2017

First, you should install Visual Studio 2017 https://visualstudio.microsoft.com/ and CMake https://cmake.org/.

Second, goto the "samples" directory and click "generate.bat" to run.

Finally, you could click `_build\mynteye_samples.sln` to open the samples project.

p.s. The example result of "generate.bat",

```
-- The C compiler identification is MSVC 19.15.26732.1
-- The CXX compiler identification is MSVC 19.15.26732.1
-- Check for working C compiler: C:/Program Files (x86)/Microsoft Visual Studio/2017/
↪Community/VC/Tools/MSVC/14.15.26726/bin/Hostx86/x64/cl.exe
-- Check for working C compiler: C:/Program Files (x86)/Microsoft Visual Studio/2017/
↪Community/VC/Tools/MSVC/14.15.26726/bin/Hostx86/x64/cl.exe -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual Studio/
↪2017/Community/VC/Tools/MSVC/14.15.26726/bin/Hostx86/x64/cl.exe
```

(continues on next page)

(continued from previous page)

```
-- Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual Studio/
↪2017/Community/VC/Tools/MSVC/14.15.26726/bin/Hostx86/x64/cl.exe -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- HOST_ARCH: x86_64
-- Visual Studio >= 2010, MSVC >= 10.0
-- C_FLAGS: /DWIN32 /D_WINDOWS /W3 -Wall -O3
-- CXX_FLAGS: /DWIN32 /D_WINDOWS /W3 /GR /EHsc -Wall -O3
-- Found mynteye: 1.3.6
-- OpenCV ARCH: x64
-- OpenCV RUNTIME: vc15
-- OpenCV STATIC: OFF
-- Found OpenCV: C:/Users/John/AppData/Roaming/Slightech/MYNTEYED/SDK/1.3.6/3rdparty/
↪opencv/build (found version "3.4.3")
-- Found OpenCV 3.4.3 in C:/Users/John/AppData/Roaming/Slightech/MYNTEYED/SDK/1.3.6/
↪3rdparty/opencv/build/x64/vc15/lib
-- You might need to add C:\Users\John\AppData\Roaming\Slightech\MYNTEYED\SDK\1.3.
↪6\3rdparty\opencv\build\x64\vc15\bin to your PATH to be able to run your␣
↪applications.
-- Generating executable get_image
-- Generating get_image.bat
-- Generating executable get_depth
-- Generating get_depth.bat
-- Generating executable get_imu
-- Generating get_imu.bat
-- Configuring done
-- Generating done
CMake Warning:
  Manually-specified variables were not used by the project:

    CMAKE_BUILD_TYPE


-- Build files have been written to: C:/Users/John/AppData/Roaming/Slightech/MYNTEYED/
↪SDK/1.3.6/samples/_build
Press any key to continue . . .
```

Tip: Right click sample and select `Set as StartUp Project`then launch with Release x64 mode.

2.5.2 Start using MYNT® EYE Depth SDK with Visual Studio 2017

Goto the `<SDK_ROOT_DIR>\projects\vs2017`, see the "README.md".

2.6 ROS Installation

2.6.1 1 Install With OpenCV ROS

If you won't use ROS(The Robot Operating System), you can skip this part.

1.1 Install ROS Kinetic

```
cd ~
wget https://raw.githubusercontent.com/oroca/oroca-ros-pkg/master/ros_install.sh && \
chmod 755 ./ros_install.sh && bash ./ros_install.sh catkin_ws kinetic
```

ROS Kinetic will install OpenCV, JPEG.

1.2 Build ROS Wrapper

```
make ros
```

Core:

```
roscore
```

RViz Display:

```
source ./wrappers/ros/devel/setup.bash
roslaunch mynteye_wrapper_d display.launch
```

Publish:

```
source ./wrappers/ros/devel/setup.bash
roslaunch mynteye_wrapper_d mynteye.launch
```

Subscribe:

```
source ./wrappers/ros/devel/setup.bash
rosrun mynteye_wrapper_d mynteye_listener_d
```

1.3 Build Beta Device ROS Wrapper

```
make ros
```

Core:

```
roscore
```

RViz Display:

```
source ./wrappers/beta_ros/devel/setup.bash
roslaunch mynteye_wrapper_d_beta display.launch
```

Publish:

```
source ./wrappers/beta_ros/devel/setup.bash
roslaunch mynteye_wrapper_d_beta mynteye.launch
```

Subscribe:

```
source ./wrappers/beta_ros/devel/setup.bash
rosrun mynteye_wrapper_d_beta mynteye_listener_d_beta
```

Subscribe:

```
source ./wrappers/beta_ros/devel/setup.bash
rosrun mynteye_wrapper_d_beta mynteye_listener_d_beta
```

2.7 ROS Usage

Compile and run the node according to *ROS Installation*.

`rostopic list` lists all released nodes:

```
/mynteye/depth/camera_info
/mynteye/depth/image_raw
/mynteye/depth/image_raw/compressed
/mynteye/depth/image_raw/compressed/parameter_descriptions
/mynteye/depth/image_raw/compressed/parameter_updates
/mynteye/depth/image_raw/compressedDepth
/mynteye/depth/image_raw/compressedDepth/parameter_descriptions
/mynteye/depth/image_raw/compressedDepth/parameter_updates
/mynteye/depth/image_raw/theora
/mynteye/depth/image_raw/theora/parameter_descriptions
/mynteye/depth/image_raw/theora/parameter_updates
/mynteye/imu/data_raw
/mynteye/imu/data_raw_processed
/mynteye/left/camera_info
/mynteye/left/image_color
/mynteye/left/image_color/compressed
...
```

`rostopic hz <topic>` checks the data:

```
subscribed to [/mynteye/imu/data_raw]
average rate: 202.806
    min: 0.000s max: 0.021s std dev: 0.00819s window: 174
average rate: 201.167
    min: 0.000s max: 0.021s std dev: 0.00819s window: 374
average rate: 200.599
    min: 0.000s max: 0.021s std dev: 0.00819s window: 574
average rate: 200.461
    min: 0.000s max: 0.021s std dev: 0.00818s window: 774
average rate: 200.310
    min: 0.000s max: 0.021s std dev: 0.00818s window: 974
  ...
```

`rostopic echo <topic>` can print and release data. Please read rostopic for more information.

The ROS file is structured like follows:

```
<sdk>/wrappers/ros/
├─src/
  └─mynteye_wrapper_d/
      ├─launch/
      │  ├─display.launch
      │  └─mynteye.launch
      ├─msg/
      ├─rviz/
```

(continues on next page)

(continued from previous page)

```
    ├─src/
    │  ├─mynteye_listener.cc
    │  └─mynteye_wrapper_nodelet.cc
    │  └─mynteye_wrapper_node.cc
    │  └─pointcloud_generatort.cc
    │  └─pointcloud_generator.h
    ├─CMakeLists.txt
    ├─nodelet_plugins.xml
    └─package.xml
```

In `mynteye.launch` ,you can configure `topics` and `frame_ids` ,decide which data to enable, and set the control options. Please set `gravity` to the local gravity acceleration.

```
<arg name="gravity" default="9.8" />
```

CHAPTER 3

MYNT® EYE SDK SAMPLES

3.1 Get camera image

Using the `DeviceMode::DEVICE_COLOR` function of the API, you can get color imageor use `DeviceMode::DEVICE_ALL` to get color and depth image.

Using `GetStreamData()` to get your data.

Reference code snippet:

```
// Device mode, default DEVICE_ALL
//   DEVICE_COLOR: IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH n
//   DEVICE_DEPTH: IMAGE_LEFT_COLOR n IMAGE_RIGHT_COLOR n IMAGE_DEPTH y
//   DEVICE_ALL:   IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH y
// Note: y: available, n: unavailable, -: depends on #stream_mode
params.dev_mode = DeviceMode::DEVICE_DEPTH;

auto left_color = cam.GetStreamData(ImageType::IMAGE_LEFT_COLOR);
    if (left_color.img) {
    cv::Mat left = left_color.img->To(ImageFormat::COLOR_BGR)->ToMat();
    painter.DrawSize(left, CVPainter::TOP_LEFT);
    painter.DrawStreamData(left, left_color, CVPainter::TOP_RIGHT);
    painter.DrawInformation(left, util::to_string(counter.fps()),
        CVPainter::BOTTOM_RIGHT);
    cv::imshow("left color", left);
```

Complete code samplessee get_stereo_image.cc .

3.2 Get camera image(Compatible with USB2.0)

Compatible with USB2.0 ,change to the resolution and frame rate for USB 2.0 automatically.Using the `DeviceMode::DEVICE_COLOR` function of the API, you can get color imageor use `DeviceMode::DEVICE_ALL` to get color and depth image.

Using `GetStreamData()` to get your data.

Reference code snippet:

```
// Device mode, default DEVICE_ALL
//   DEVICE_COLOR: IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH n
//   DEVICE_DEPTH: IMAGE_LEFT_COLOR n IMAGE_RIGHT_COLOR n IMAGE_DEPTH y
//   DEVICE_ALL:   IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH y
// Note: y: available, n: unavailable, -: depends on #stream_mode
params.dev_mode = DeviceMode::DEVICE_DEPTH;

auto left_color = cam.GetStreamData(ImageType::IMAGE_LEFT_COLOR);
    if (left_color.img) {
    cv::Mat left = left_color.img->To(ImageFormat::COLOR_BGR)->ToMat();
    painter.DrawSize(left, CVPainter::TOP_LEFT);
    painter.DrawStreamData(left, left_color, CVPainter::TOP_RIGHT);
    painter.DrawInformation(left, util::to_string(counter.fps()),
        CVPainter::BOTTOM_RIGHT);
    cv::imshow("left color", left);
```

Complete code samplessee get_image.cc .

3.3 Get depth image

Depth images belongs to the upper layer of synthetic data.

You can change `depth_mode` to change the display of the depth image.

```
// Depth mode: colorful(default), gray, raw
params.depth_mode = DepthMode::DEPTH_RAW;
```

Then you can get it through `GetStreamData()`.In addition, it should be check not be empty before use.

Reference code snippet:

```
auto image_depth = cam.GetStreamData(ImageType::IMAGE_DEPTH);
if (image_depth.img) {
  cv::Mat depth = image_depth.img->To(ImageFormat::DEPTH_RAW)->ToMat();

  cv::setMouseCallback("depth", OnDepthMouseCallback, &depth_region);
  // Note: DrawRect will change some depth values to show the rect.
  depth_region.DrawRect(depth);
  cv::imshow("depth", depth);

  depth_region.ShowElems<ushort>(depth, [](const ushort& elem) {
    return std::to_string(elem);
  }, 80, depth_info);
}
```

The above code uses OpenCV to display the image. When the display window is selected, pressing ESC/Q will end the program.

Note: *get_depth* sample only support *DEPTH_RAW* mode.You can modify `depth_mode` parameter of other samples to get depth images

---

Complete code examples, see get_depth.cc.

3.4 Get point image

Point images belongs to upper layer of synthetic data.You can get it through `GetStreamData()`.It should be check not empty before use.

Sample code snippet:

```
auto image_color = cam.GetStreamData(ImageType::IMAGE_LEFT_COLOR);
auto image_depth = cam.GetStreamData(ImageType::IMAGE_DEPTH);
if (image_color.img && image_depth.img) {
    cv::Mat color = image_color.img->To(ImageFormat::COLOR_BGR)
        ->ToMat();
    painter.DrawSize(color, CVPainter::TOP_LEFT);
    painter.DrawStreamData(color, image_color, CVPainter::TOP_RIGHT);
    painter.DrawInformation(color, util::to_string(counter.fps()),
        CVPainter::BOTTOM_RIGHT);

    cv::Mat depth = image_depth.img->To(ImageFormat::DEPTH_RAW)
        ->ToMat();

    cv::imshow("color", color);

    viewer.Update(color, depth);
}
```

PCL is used to display point images above. Program will close when point image window is closed.

Complete code examples, see get_points.cc.

3.5 Get IMU data

You need `EnableMotionDatas()`to enable caching in order to get IMU data from `GetMotionDatas()`.Otherwise, IMU data is only available through the callback interface, see *Get data from callbacks*.

Sample code snippet:

```
auto motion_datas = cam.GetMotionDatas();
if (motion_datas.size() > 0) {
    std::cout << "Imu count: " << motion_datas.size() << std::endl;
    for (auto data : motion_datas) {
        if (data.imu) {
            if (data.imu->flag == MYNTEYE_IMU_ACCEL) {
                counter.IncrAccelCount();
                std::cout << "[accel] stamp: " << data.imu->timestamp
                    << ", x: " << data.imu->accel[0]
                    << ", y: " << data.imu->accel[1]
                    << ", z: " << data.imu->accel[2]
                    << ", temp: " << data.imu->temperature
                    << std::endl;
            } else if (data.imu->flag == MYNTEYE_IMU_GYRO) {
                counter.IncrGyroCount();
                std::cout << "[gyro] stamp: " << data.imu->timestamp
                << ", x: " << data.imu->gyro[0]
                << ", y: " << data.imu->gyro[1]
                << ", z: " << data.imu->gyro[2]
```

(continues on next page)

(continued from previous page)

```
                << ", temp: " << data.imu->temperature
                << std::endl;
            } else {
                std::cerr << "Imu type is unknown" << std::endl;
            }
        } else {
          std::cerr << "Motion data is empty" << std::endl;
        }
    }
    std::cout << std::endl;
}
```

OpenCV is used to display image and data. When window is selected, press ESC/Q to exit program.

Complete code examples, see get_imu.cc.

3.6 Get data from callbacks

API offers function `SetStreamCallback()` and `SetMotionCallback()` to set callbacks for various data.

Reference code snippet:

```
cam.SetImgInfoCallback([](const std::shared_ptr<ImgInfo>& info) {
    std::cout << "  [img_info] fid: " << info->frame_id
        << ", stamp: " << info->timestamp
        << ", expos: " << info->exposure_time << std::endl
        << std::flush;
});
for (auto&& type : types) {
    // Set stream data callback
    cam.SetStreamCallback(type, [](const StreamData& data) {
    std::cout << "  [" << data.img->type() << "] fid: "
        << data.img->frame_id() << std::endl
        << std::flush;
    });
}

// Set motion data callback
cam.SetMotionCallback([](const MotionData& data) {
    if (data.imu->flag == MYNTEYE_IMU_ACCEL) {
        std::cout << "[accel] stamp: " << data.imu->timestamp
        << ", x: " << data.imu->accel[0]
        << ", y: " << data.imu->accel[1]
        << ", z: " << data.imu->accel[2]
        << ", temp: " << data.imu->temperature
        << std::endl;
    } else if (data.imu->flag == MYNTEYE_IMU_GYRO) {
        std::cout << "[gyro] stamp: " << data.imu->timestamp
        << ", x: " << data.imu->gyro[0]
        << ", y: " << data.imu->gyro[1]
        << ", z: " << data.imu->gyro[2]
        << ", temp: " << data.imu->temperature
        << std::endl;
    }
    std::cout << std::flush;
});
```

OpenCV is used to display images and data above. When the window is selected, pressing ESC/Q will exit program.

Complete code examples, see get_from_callbacks.cc.

3.7 Get different types of image by options

`get_all_with_options` sample can add different options to control device.

`get_all_with_options -h`:

```
Open device with different options.

Options:
  -h, --help            show this help message and exit
  -m, --imu             Enable imu datas

  Open Params:
    The open params

    -i INDEX, --index=INDEX
                        Device index
    -f RATE, --rate=RATE
                        Framerate, range [0,60], [30](STREAM_2560x720),
                        default: 10
    --dev-mode=MODE       Device mode, default 2 (DEVICE_ALL)
                          0: DEVICE_COLOR, left y right - depth n
                          1: DEVICE_DEPTH, left n right n depth y
                          2: DEVICE_ALL,   left y right - depth y
                          Note: y: available, n: unavailable, -: depends on
                        stream mode
    --cm=MODE           Color mode, default 0 (COLOR_RAW)
                          0: COLOR_RAW, color raw
                          1: COLOR_RECTIFIED, color rectified
    --dm=MODE           Depth mode, default 2 (DEPTH_COLORFUL)
                          0: DEPTH_RAW
                          1: DEPTH_GRAY
                          2: DEPTH_COLORFUL
    --sm=MODE           Stream mode of color & depth,
                        default 2 (STREAM_1280x720)
                          0: STREAM_640x480, 480p, vga, left
                          1: STREAM_1280x480, 480p, vga, left+right
                          2: STREAM_1280x720, 720p, hd, left
                          3: STREAM_2560x720, 720p, hd, left+right
    --csf=MODE          Stream format of color,
                        default 1 (STREAM_YUYV)
                          0: STREAM_MJPG
                          1: STREAM_YUYV
    --dsf=MODE          Stream format of depth,
                        default 1 (STREAM_YUYV)
                          1: STREAM_YUYV
    --ae                Enable auto-exposure
    --awb               Enable auto-white balance
    --ir=VALUE          IR intensity, range [0,6], default 0
    --ir-depth          Enable ir-depth-only

  Feature Toggles:
    The feature toggles
```

(continues on next page)

(continued from previous page)

```
    --proc=MODE         Enable process mode, e.g. imu assembly, temp_drift
                          0: PROC_NONE
                          1: PROC_IMU_ASSEMBLY
                          2: PROC_IMU_TEMP_DRIFT
                          3: PROC_IMU_ALL
    --img-info          Enable image info, and sync with image
```

e.g. `./samples/_output/bin/get_all_with_options -f 60 --dev-mode=0 --sm=2` displays 1280x720 60fps left unrectified image.

Complete code samplessee get_all_with_options.cc .

3.8 Get image calibration parameters

Use `GetStreamIntrinsics()` and `GetStreamExtrinsics()` to get image calibration parameters.

Reference code snippet

```
auto vga_intrinsics = cam.GetStreamIntrinsics(StreamMode::STREAM_1280x480, &in_ok);
auto vga_extrinsics = cam.GetStreamExtrinsics(StreamMode::STREAM_1280x480, &ex_ok);
std::cout << "VGA Intrinsics left: {" << vga_intrinsics.left << "}" << std::endl;
std::cout << "VGA Intrinsics right: {" << vga_intrinsics.right << "}" << std::endl;
std::cout << "VGA Extrinsics left to right: {" << vga_extrinsics << "}" << std::endl;
out << "VGA Intrinsics left: {" << vga_intrinsics.left << "}" << std::endl;
out << "VGA Intrinsics right: {" << vga_intrinsics.right << "}" << std::endl;
out << "VGA Extrinsics left to right: {" << vga_extrinsics << "}" << std::endl;
```

The result will be saved in the current file directory.Reference result on Linux:

```
VGA Intrinsics left: {width: [640], height: [480], fx: [358.45721435546875000], fy:␣
↪[359.53115844726562500], cx: [311.12109375000000000], cy: [242.
↪63494873046875000]coeffs: [-0.28297042846679688, 0.06178283691406250, -0.
↪00030517578125000, 0.00218200683593750, 0.00000000000000000]}
VGA Intrinsics right: {width: [640], height: [480], fx: [360.13885498046875000], fy:␣
↪[360.89624023437500000], cx: [325.11029052734375000], cy: [251.
↪46371459960937500]coeffs: [-0.30667877197265625, 0.08611679077148438, -0.
↪00030136108398438, 0.00155639648437500, 0.00000000000000000]}
VGA Extrinsics left to right: {rotation: [0.99996054172515869, 0.00149095058441162, 0.
↪00875246524810791, -0.00148832798004150, 0.99999880790710449, -0.00030362606048584,␣
↪-0.00875294208526611, 0.00029063224792480, 0.99996161460876465], translation: [-120.
↪36341094970703125, 0.00000000000000000, 0.00000000000000000]}
```

Complete code examples, see get_img_params.cc.

3.9 Get IMU calibration parameters

Use `GetMotionIntrinsics()` and `GetMotionExtrinsics` to get current IMU calibration parameters.

Reference code snippet:

```
auto intrinsics = cam.GetMotionIntrinsics(&in_ok);
std::cout << "Motion Intrinsics: {" << intrinsics << "}" << std::endl;
out << "Motion Intrinsics: {" << intrinsics << "}" << std::endl;
```

The result will be saved in the current file directory.Reference result on Linux:

```
Motion Intrinsics: {accel: {scale: [1.00205999990004191, 0.00000000000000000, 0.
↪00000000000000000, 0.00000000000000000, 1.00622999999999996, 0.00000000000000000, 0.
↪00000000000000000, 0.00000000000000000, 1.00171999999999994], assembly: [1.
↪00000000000000000, 0.00672262000000000, -0.00364474000000000, 0.00000000000000000,␣
↪1.00000000000000000, 0.00101348000000000, -0.00000000000000000, 0.00000000000000000,
↪ 1.00000000000000000, 1.00000000000000000], drift: [0.00000000000000000, 0.
↪00000000000000000, 0.00000000000000000], noise: [0.00000000000000000, 0.
↪00000000000000000, 0.00000000000000000], bias: [0.00000000000000000, 0.
↪00000000000000000, 0.00000000000000000], x: [0.00856165620000000, -0.
↪00009840052800000], y: [0.05968393300000000, -0.00130967680000000], z: [0.
↪01861442050000000, -0.00016033523000000]}, gyro: {scale: [1.00008999999999992, 0.
↪00000000000000000, 0.00000000000000000, 0.00000000000000000, 0.99617599999999995, 0.
↪00000000000000000, 0.00000000000000000, 0.00000000000000000, 1.00407000000000002],␣
↪assembly: [1.00000000000000000, -0.00700362000000000, -0.00326206000000000, 0.
↪00549571000000000, 1.00000000000000000, 0.00224867000000000, 0.00236088000000000, 0.
↪00044507800000000, 1.00000000000000000, 1.00000000000000000], drift: [0.
↪00000000000000000, 0.00000000000000000, 0.00000000000000000], noise: [0.
↪00000000000000000, 0.00000000000000000, 0.00000000000000000], bias: [0.
↪00000000000000000, 0.00000000000000000, 0.00000000000000000], x: [0.
↪18721455299999998, 0.00077411070000000], y: [0.60837032000000002, -0.
↪00939702710000000], z: [-0.78549276000000001, 0.02584820200000000]}}
```

Complete code examples, see get_imu_params.cc.

3.10 Set open parameters

3.10.1 Set the resolution of image

Using the `params.stream_mode` parameter, you can set the resolution of the image.

> Attention: Now image resolution supports 4 types: 640X4801280x720 for single camera. 1280x480 2560x720 for left and right camera.

Reference code snippet:

```
// Stream mode: left color only
// params.stream_mode = StreamMode::STREAM_640x480;  // vga
// params.stream_mode = StreamMode::STREAM_1280x720;  // hd
// Stream mode: left+right color
// params.stream_mode = StreamMode::STREAM_1280x480;  // vga
params.stream_mode = StreamMode::STREAM_2560x720;  // hd
```

3.10.2 Set the frame rate of image

Using the `params.framerate` parameter, you can set the frame rate of image.

Note: The effective fps of the image(0-60) - The effective fps of the image in 2560x720 resolution (30)

Reference code snippet:

```
// Framerate: 30(default), [0,60], [30](STREAM_2560x720)
params.framerate = 30;
```

3.10.3 Set color mode

Using the `params.color_mode` parameteryou can set the color mode of image.

`COLOR_RAW` is original image `COLOR_RECTIFIED` is rectified image.

Reference code snippet:

```
// Color mode: raw(default), rectified
// params.color_mode = ColorMode::COLOR_RECTIFIED;
```

3.10.4 Set depth mode

Using the `params.depth_mode` parameteryou can set the depth mode.

`DEPTH_COLORFUL` is colorful depth image`DEPTH_GRAY` is grey depth image `DEPTH_RAW` is original depth image

Reference code snippet:

```
// Depth mode: colorful(default), gray, raw
// params.depth_mode = DepthMode::DEPTH_GRAY;
```

3.10.5 Enable auto exposure and auto white balance

Set `params.state_ae` and `params.state_awb` to `true` , you can enable auto exposure and auto white balance.

By default auto exposure and auto white balance are enabledif you want to disableyou can set parameters to `false` .

Reference code snippet:

```
// Auto-exposure: true(default), false
// params.state_ae = false;

// Auto-white balance: true(default), false
// params.state_awb = false;
```

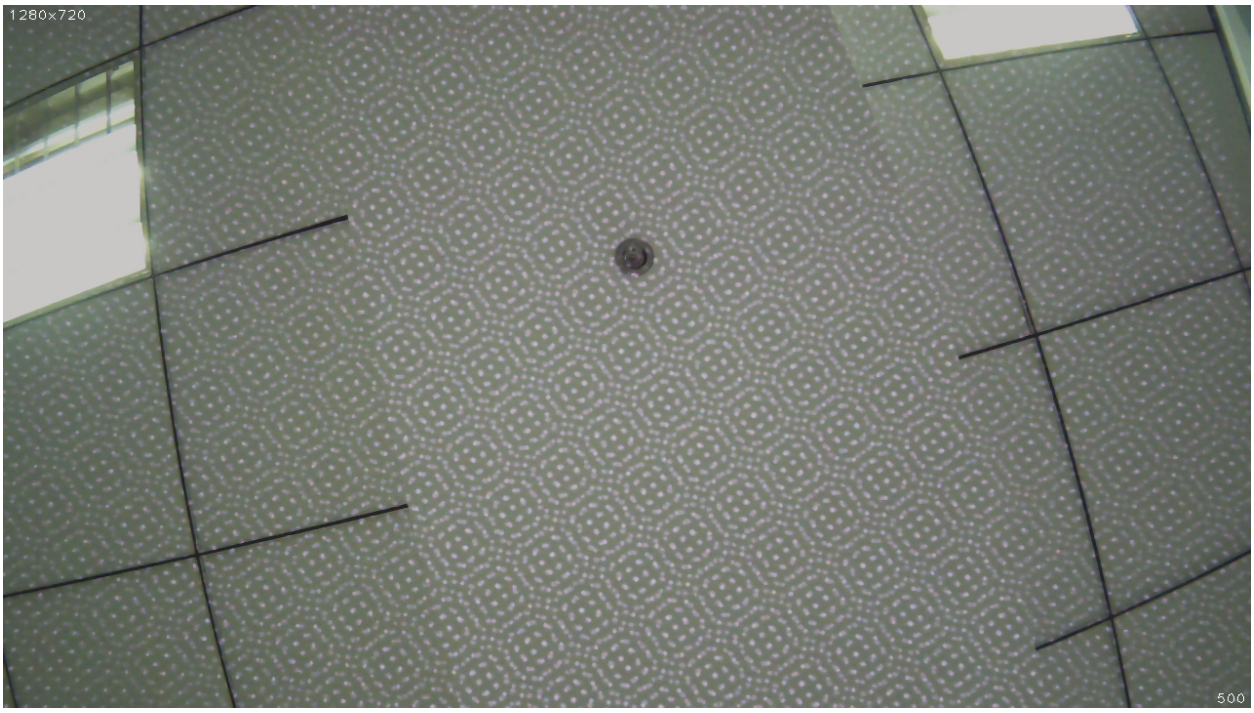
3.10.6 Enable IR and its adjustments function

Using the `params.ir_intensity` parameteryou can set IR's intensity of image. Enabling IR is setting `params.ir_intensity` greater than 0. The greater the value, the greater the IR's intensity.(max is 10).

Reference code snippet:

```
// Infrared intensity: 0(default), [0,10]
params.ir_intensity = 4;
```

Note: After turning this fuction on, you can see ir pattern:



3.10.7 Enable IR Depth Only

Using the `params.ir_depth_only` parameteryou can set IR Depth Only function. This is disabled by default. After turning this function on, IR only works on depth images. IR pattern will not show in color images.

Note: This function doesn't work on 15 frame rate below. After turning this function on, frame rate will be divided equally.For example,when set frame rate of image to 30 fps, the frame rate of color image is 15 fps. The frame rate of depth image is 15 fps too.

Reference code snippet:

```
// IR Depth Only: true, false(default)
// Note: IR Depth Only mode support frame rate between 15fps and 30fps.
//     When dev_mode != DeviceMode::DEVICE_ALL,
//       IR Depth Only mode not be supported.
//     When stream_mode == StreamMode::STREAM_2560x720,
//       frame rate only be 15fps in this mode.
//     When frame rate less than 15fps or greater than 30fps,
//       IR Depth Only mode will be not available.
// params.ir_depth_only = false;
```

3.10.8 Adjust colour depth value

Using the `params.colour_depth_value` parameter, The value is 1000 by default.

Reference code snippet:

```
// Colour depth image, default 1000. [0, 16384]
// params.colour_depth_value = 1000;
```

Reference running results on Linux:

```
Open device: 0, /dev/video1

D/eSPDI_API: SetPropertyValue control=7 value=0D/eSPDI_API: SetPropertyValue␣
↪control=7 value=35D/eSPDI_API: SetPropertyValue control=7 value=1-- Auto-exposure␣
↪state: enabled
D/eSPDI_API: SetPropertyValue control=7 value=0D/eSPDI_API: SetPropertyValue␣
↪control=7 value=12D/eSPDI_API: SetPropertyValue control=7 value=1-- Auto-white␣
↪balance state: enabled
-- Framerate: 5
D/eSPDI_API: SetPropertyValue control=7 value=4 SetDepthDataType: 4
-- Color Stream: 1280x720 YUYV
-- Depth Stream: 1280x720 YUYV

D/eSPDI_API: SetPropertyValue control=7 value=0D/eSPDI_API: SetPropertyValue␣
↪control=7 value=3D/eSPDI_API: SetPropertyValue control=7 value=4
-- IR intensity: 4
D/eSPDI_API: CVideoDevice::OpenDevice 1280x720 fps=5

Open device success
```

Note: After changing the parameters, you need to run in the sdk directory

```
make samples
```

to make the set parameters take effect.

Complete code samplessee get_image.cc .

3.11 Camera control parameters API

3.11.1 Open or close auto exposure

```
/** Auto-exposure enabled or not  default enabled*/
bool AutoExposureControl(bool enable);    see "camera.h"
```

3.11.2 Open or close auto white balance

```
/** Auto-white-balance enabled or not  default enabled*/
bool AutoWhiteBalanceControl(bool enable);    see "camera.h"
```

3.11.3 Set infrared(IR) intensity

```
/** set infrared(IR) intensity [0, 10] default 4*/
void SetIRIntensity(const std::uint16_t &value);     see "camera.h"
```

3.11.4 Set global gain

Note: You have to close auto exposure first after opening camera.

```
/** Set global gain [1 - 16]
 * value -- global gain value
 * */
void SetGlobalGain(const float &value);     see "camera.h"
```

3.11.5 Set the exposure time

Note: You have to close auto exposure first after opening camera.

```
/** Set exposure time [1ms - 2000ms]
 * value -- exposure time value
 * */
void SetExposureTime(const float &value);     see "camera.h"
```

Reference code snippet:

```
cam.Open(params);
cam.AutoExposureControl(false);
cam.SetGlobalGain(1);
cam.SetExposureTime(0.3);
```

Note: After changing the parameters, you need to run in the sdk directory

```
make samples
```

to make the set parameters take effect.

CHAPTER 4

MYNT® EYE SDK TOOLS

4.1 Analyze IMU data

The SDK provides the script imu_analytics.py for IMU analysis. The tool details can be seen in tools/README.md .

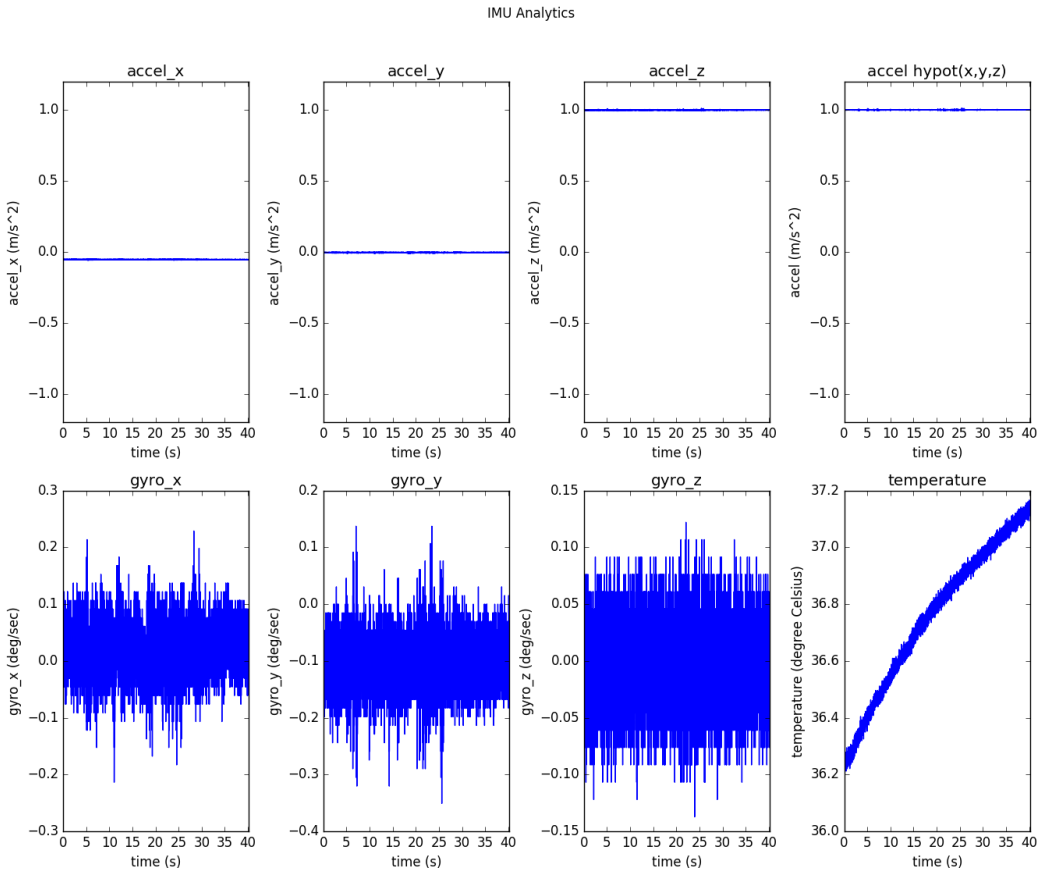
Reference to run commands on Linux:

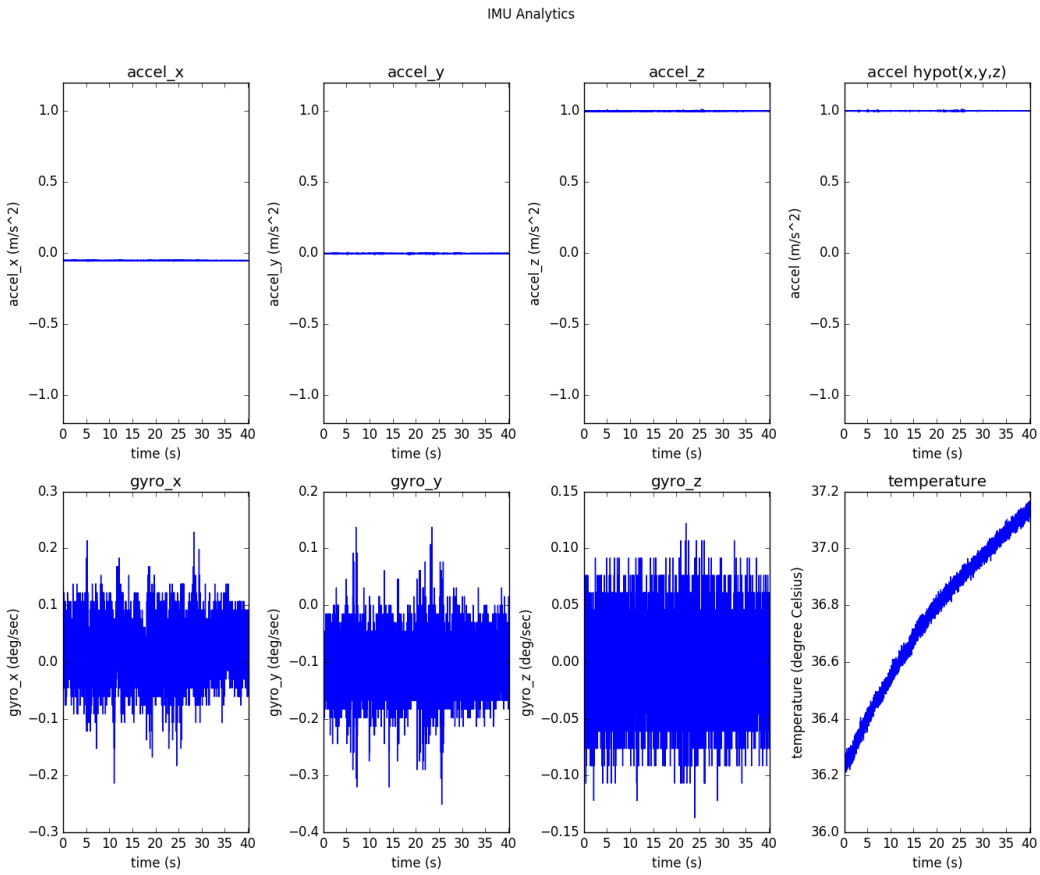
```
$ python tools/analytics/imu_analytics.py -i dataset -c tools/config/mynteye/mynteye_
↪config.yaml -al=-1.2,1.2 -gl= -gdu=d -gsu=d -kl=
```

Reference to results on Linux:

```
$ python tools/analytics/imu_analytics.py -i dataset -c tools/config/mynteye/mynteye_
↪config.yaml -al=-1.2,1.2 -gl= -gdu=d -gsu=d -kl=
imu analytics ...
  input: dataset
  outdir: dataset
  gyro_limits: None
  accel_limits: [(-1.2, 1.2), (-1.2, 1.2), (-1.2, 1.2), (-1.2, 1.2)]
  time_unit: None
  time_limits: None
  auto: False
  gyro_show_unit: d
  gyro_data_unit: d
  temp_limits: None
open dataset ...
  imu: 20040, temp: 20040
  timebeg: 4.384450, timeend: 44.615550, duration: 40.231100
save figure to:
  dataset/imu_analytics.png
imu analytics done
```

The analysis result graph will be saved in the data set directory. as follows:


IMU Analytics
accel_x
accel_y
accel_z
accel hypot(x,y,z)
accel_x (m/s^2)
accel_y (m/s^2)
accel_z (m/s^2)
accel (m/s^2)
gyro_x
gyro_y
gyro_z
temperature
gyro_x (deg/sec)
gyro_y (deg/sec)
gyro_z (deg/sec)
temperature (degree Celsius)
time (s)



In addition, the script specific options can be executed -h:

```
$ python tools/analytics/imu_analytics.py -h
```

4.2 Analyze time stamps

SDK provides a script for timestamp analysis stamp_analytics.py . Tool details are visible in tools/README.md .

Reference run commands on Linux:

```
$ python tools/analytics/stamp_analytics.py -i dataset -c tools/config/mynteye/
↪mynteye_config.yaml
```

Reference to results on Linux:

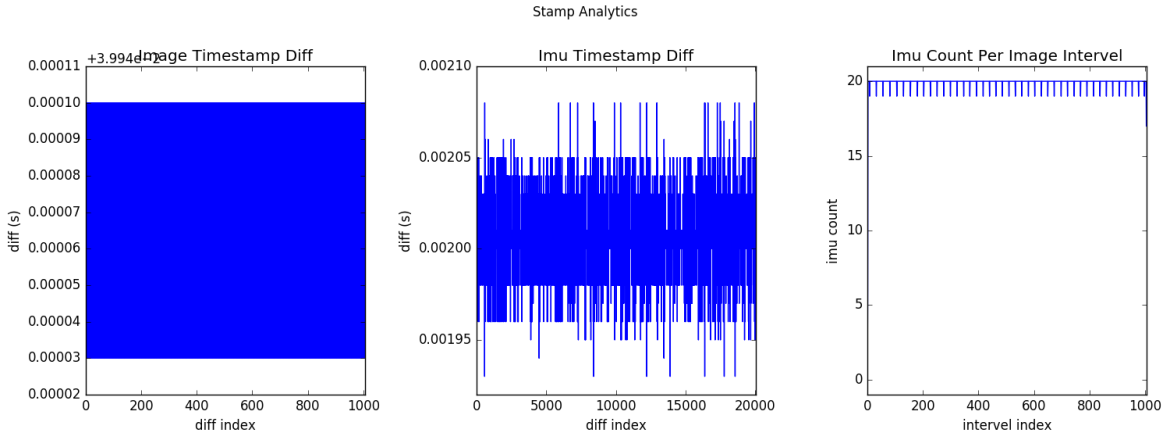
```
$ python tools/analytics/stamp_analytics.py -i dataset -c tools/config/mynteye/
↪mynteye_config.yaml
stamp analytics ...
  input: dataset
  outdir: dataset
open dataset ...
save to binary files ...
  binimg: dataset/stamp_analytics_img.bin
```

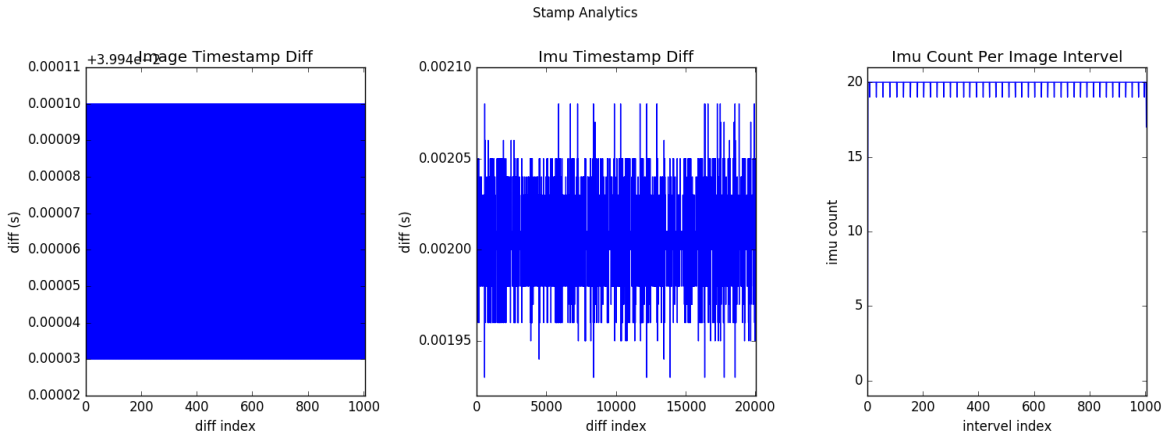
(continues on next page)

(continued from previous page)

```
  binimu: dataset/stamp_analytics_imu.bin
  img: 1007, imu: 20040

rate (Hz)
  img: 25, imu: 500
sample period (s)
  img: 0.04, imu: 0.002

diff count
  imgs: 1007, imus: 20040
  imgs_t_diff: 1006, imus_t_diff: 20039

diff where (factor=0.1)
  imgs where diff > 0.04*1.1 (0)
  imgs where diff < 0.04*0.9 (0)
  imus where diff > 0.002*1.1 (0)
  imus where diff < 0.002*0.9 (0)

image timestamp duplicates: 0

save figure to:
  dataset/stamp_analytics.png
stamp analytics done
```

The analysis result graph will be saved in the dataset directory. as follow:





In addition, the script specific options can be executed -h to understand:

```
$ python tools/analytics/stamp_analytics.py -h
```

4.3 Record data sets

The SDK provides the tool record for recording data sets. Tool details can be seen in tools/README.md .

Reference run command on Linux:

```
./tools/_output/bin/dataset/record
```

Reference run command on Windows:

```
.\tools\_output\bin\dataset\record.bat
```

Reference run results on Linux:

```
$ ./tools/_output/bin/dataset/record
Saved 1007 imgs, 20040 imus to ./dataset
I0513 21:29:38.608772 11487 record.cc:118] Time beg: 2018-05-13 21:28:58.255395, end:␣
↪2018-05-13 21:29:38.578696, cost: 40323.3ms
I0513 21:29:38.608853 11487 record.cc:121] Img count: 1007, fps: 24.9732
I0513 21:29:38.608873 11487 record.cc:123] Imu count: 20040, hz: 496.983
```

Results save into `<workdir>/dataset` by default. You can also add parameter, select other directory to save.

Record contents:

```
<workdir>/
└─dataset/
    ├─left/
    │   ├─stream.txt  # Image infomation
    │   └─...
    ├─right/
    │   ├─stream.txt  # Image information
    │   └─...
    └─motion.txt  # IMU information
```

Tip: When recording data, `dataset.cc` has annotated display image inside `cv::imwrite()` . Because these operations are time-consuming, they can cause images to be discarded. In other words, consumption can't keep up with production, so some images are discarded. `GetStreamDatas()` used in `record.cc` only caches the latest 4 images.

4.4 Save device infomation and parameters

The SDK provides a tool `save_all_infos` for save information and parameters.

Reference commands:

```
./tools/_output/bin/writer/save_all_infos
```

```
# Windows
.\tools\_output\bin\writer\save_all_infos.bat
```

Reference result on Linux:

```
I/eSPDI_API: eSPDI: EtronDI_Init
Device descriptors:
  name: MYNT-EYE-D1000
  serial_number: 203837533548500F002F0028
  firmware_version: 1.0
  hardware_version: 2.0
  spec_version: 1.0
  lens_type: 0000
  imu_type: 0000
  nominal_baseline: 120
```

Result save into `<workdir>/config` by default. You can also add parameters to select other directory for save.

Saved contents:

```
<workdir>/
└─config/
   └─SN0610243700090720/
      ├─device.info
      └─imu.params
```

Complete code samplessee save_all_infos.cc .

4.5 Write IMU parameters

SDK provides the tool `imu_params_writer` to write IMU parameters.

Information about how to get IMU parameters, please read *Get IMU calibration parameters* .

Reference commands:

```
./tools/_output/bin/writer/imu_params_writer tools/writer/config/imu.params

# Windows
.\tools\_output\bin\writer\imu_params_writer.bat tools\writer\config\imu.params
```

The path of parameters file can be found in tools/writer/config/imu.params . If you calibrated the parameters yourself, you can edit the file and run above commands to write them into the device.

Warning - Please don't override parameters, you can use `save_all_infos` to backup parameters.

Complete code samplessee imu_params_writer.cc .

4.6 Update HID Firmware

4.6.1 Get Firmware

Latest firmware: mynteye-d-hid-firmware-1.2.bin Google Drive, Baidu Pan

4.6.2 Compile SDK Tools

```
cd <sdk> #local path of MYNT-EYE-D-SDK
make tools
```

4.6.3 Update Firmware

```
./tools/_output/bin/writer/device_hid_update <firmware-file-path>
```

4.7 Update Camera Firmware

Note: This tool does not support beta device upgrade.

4.7.1 Get Firmware

Latest firmware: SICI-B12-B0135P-016-005-ISO_Plugout2M(Interleave).bin Google Drive, Baidu Pan

4.7.2 Get Update Tool

Latest tool: eSPWriter_1.0.6.zip Google Drive, Baidu Pan

4.7.3 Update Firmware

Note: Please follow the steps to upgrade firmware.(Otherwise, the camera calibration parameters will be lost.)

1, Select camera device.

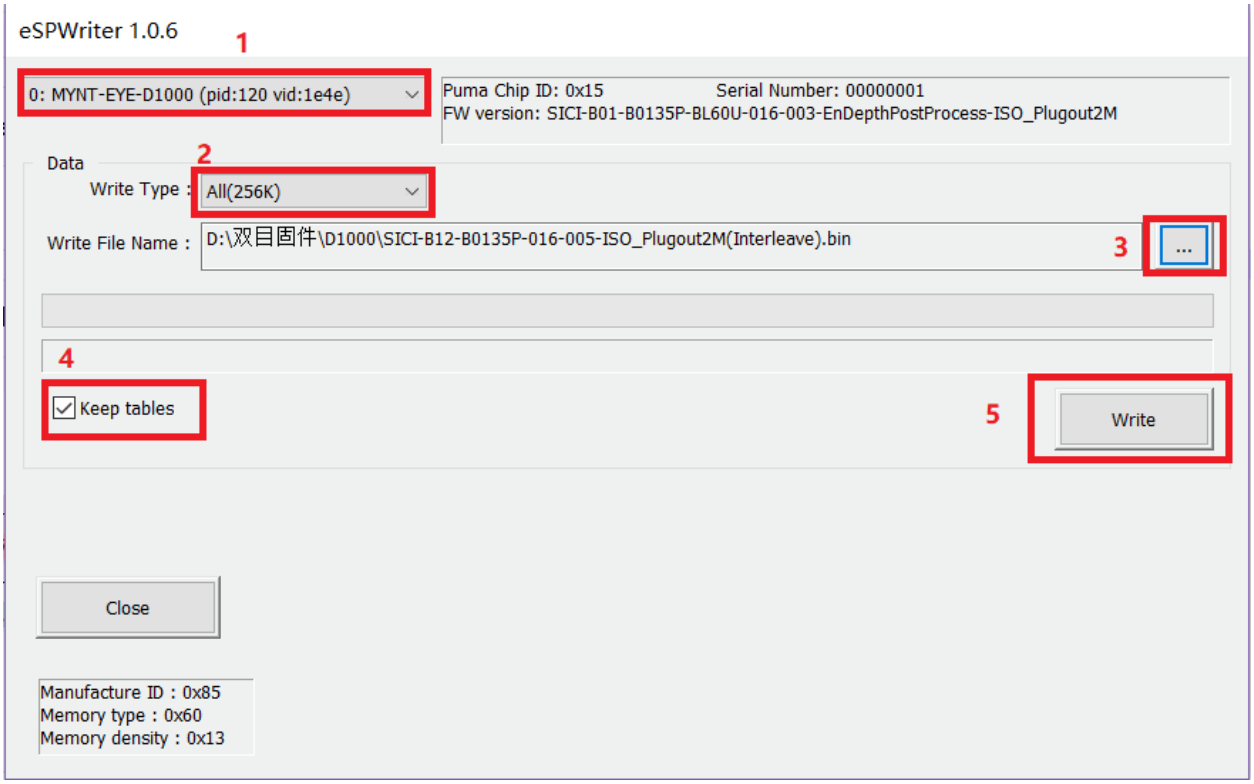
2, Select data type(256KB).

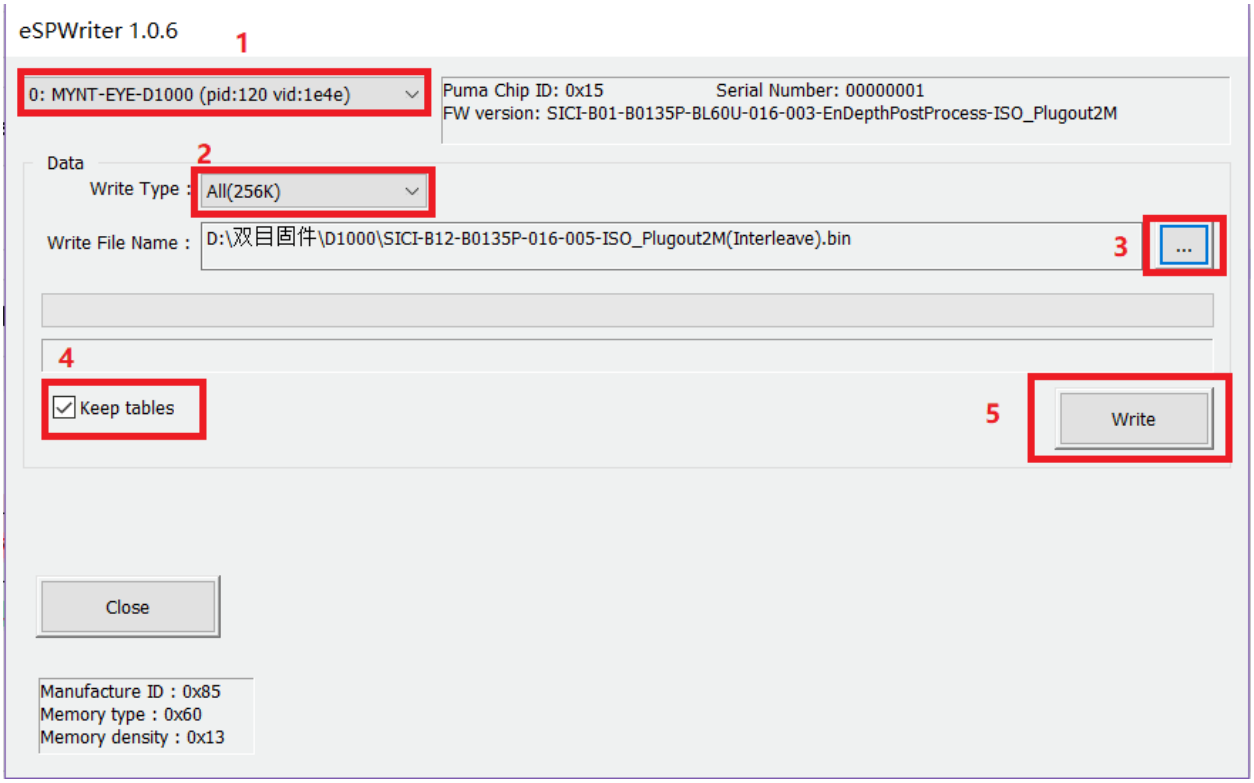
3, Select camera firmware.

4, Select `Keep tables` (in order to keep calibration parameters).

5, Click `Write`.

Use the tool according to diagram:





CHAPTER 5

PROJECT DEMOS

5.1 How to use SDK with Visual Studio 2017

This tutorial will create a project with Visual Studio 2017 to start using SDK.

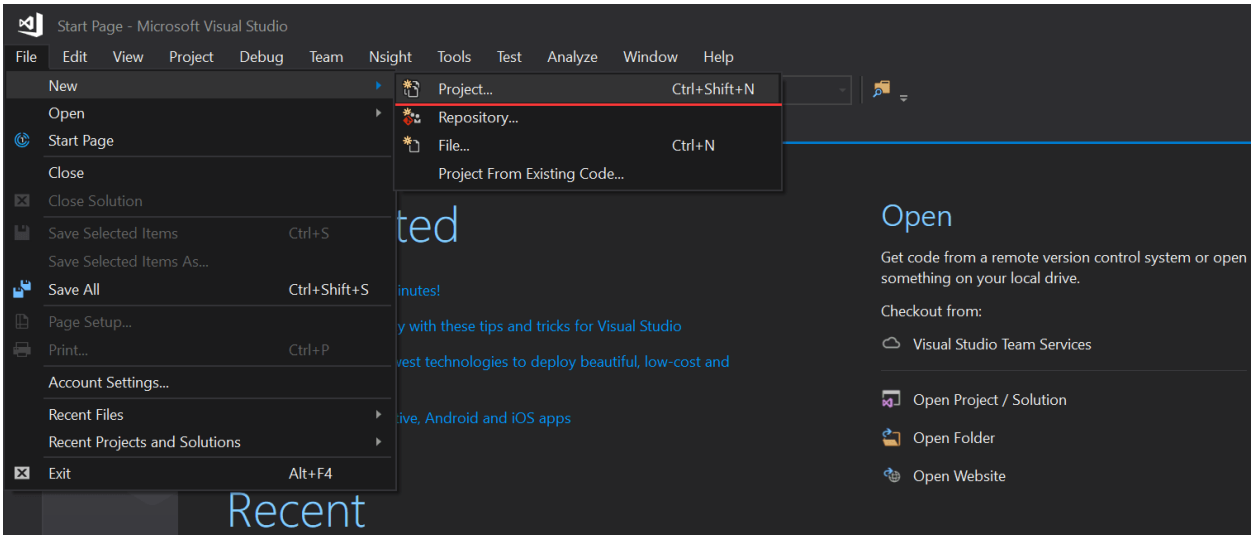
> You could find the project demo in `<sdk>/platforms/projects/vs2017` directory.

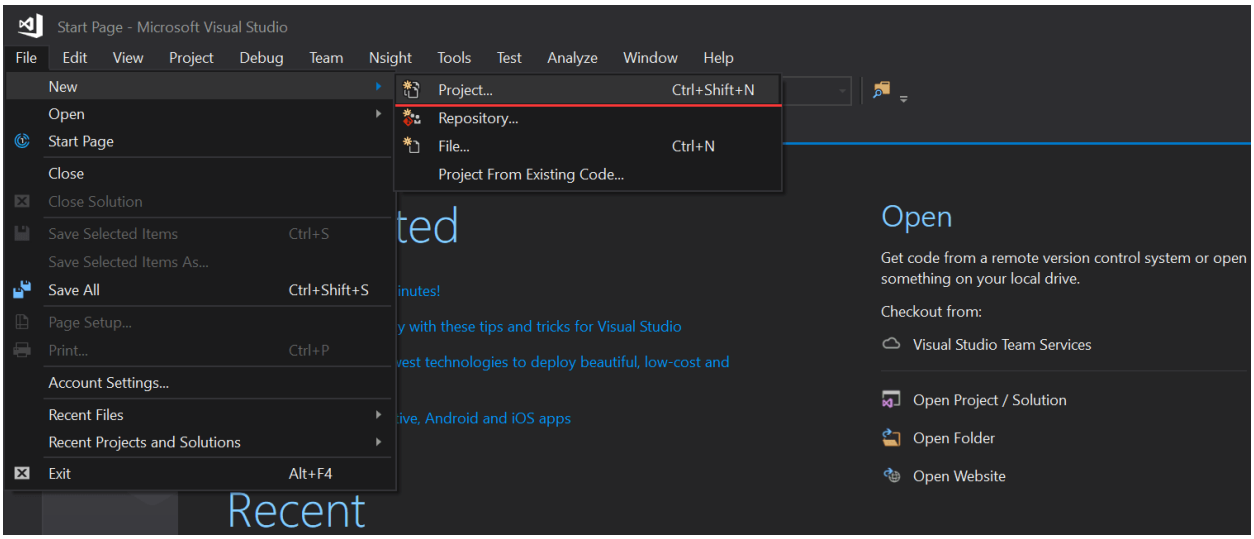
5.1.1 Preparation

- Windows: install the win pack of SDK

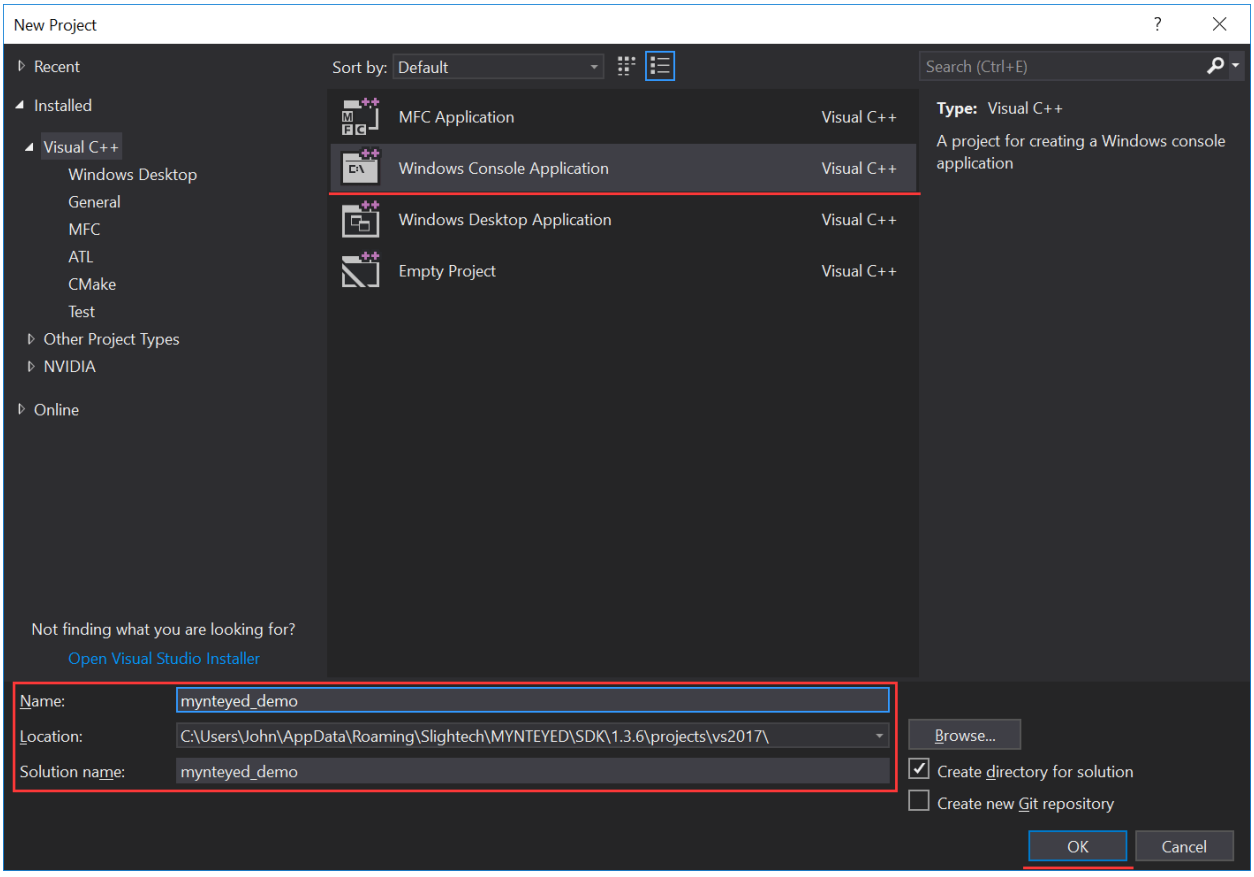
5.1.2 Create Project

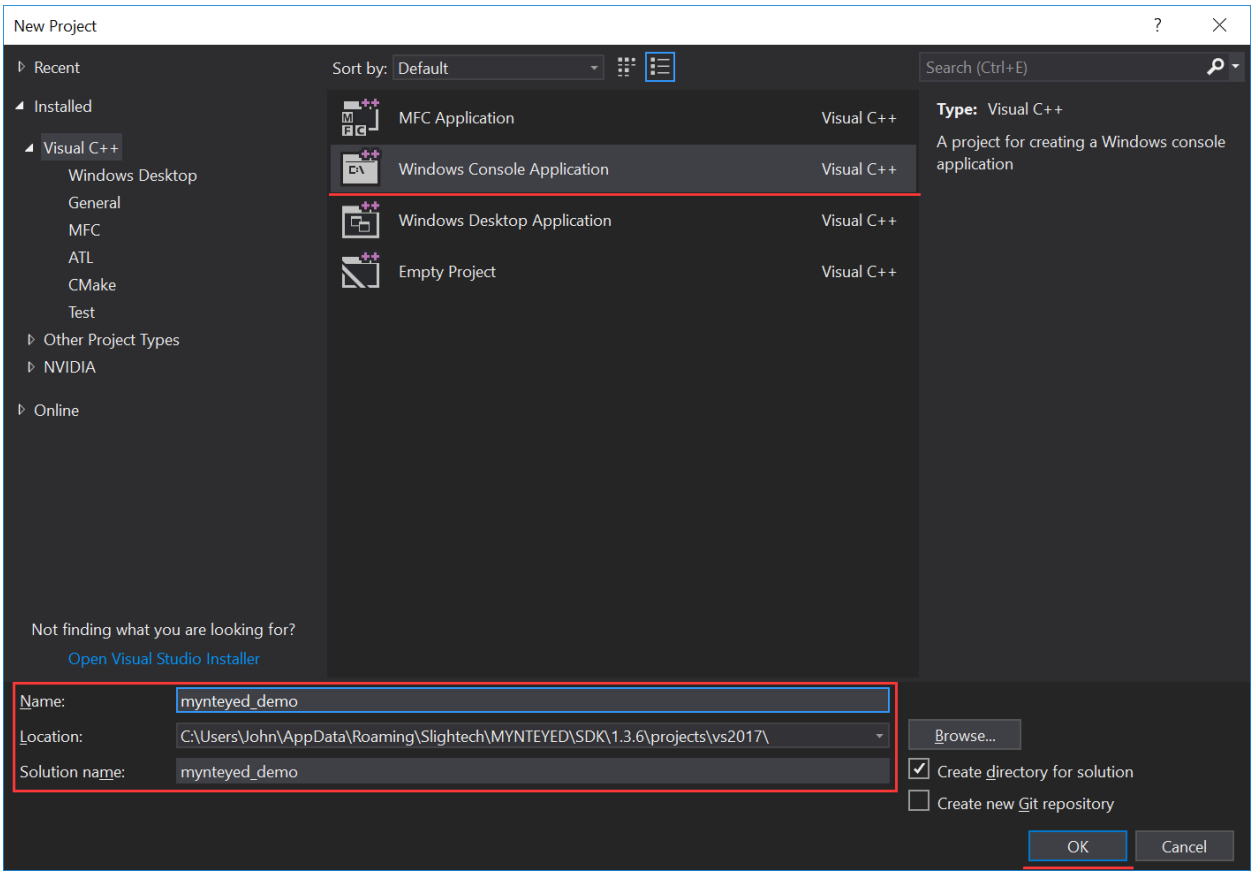
Open Visual Studio 2017, then `File > New > Project`,



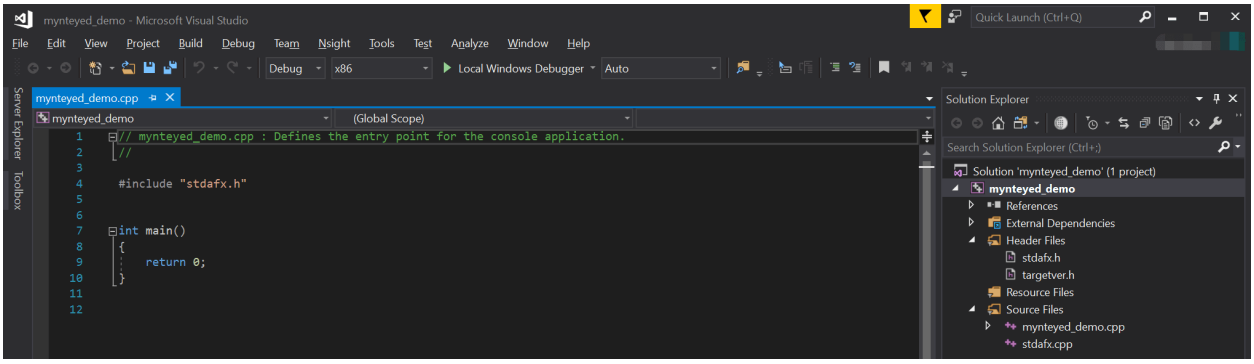


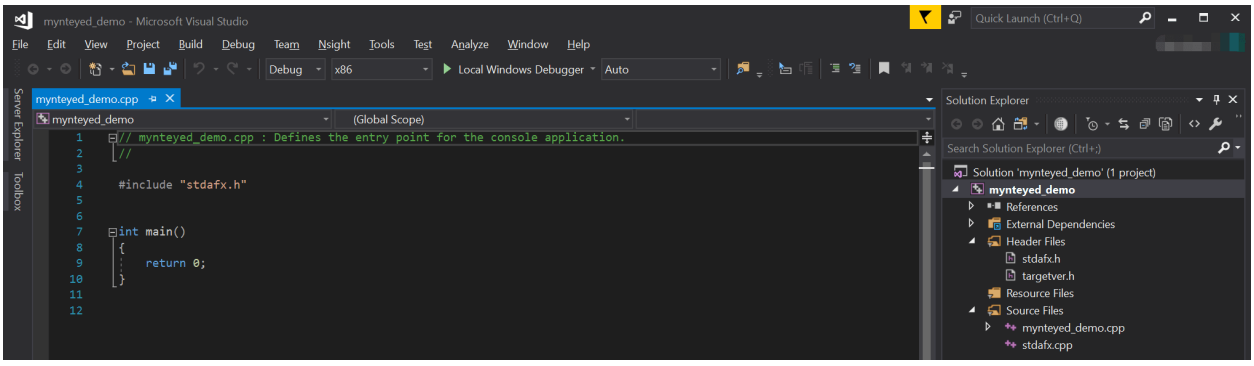
Select “Windows Console Application”, set the project’s name and location, click “OK”,





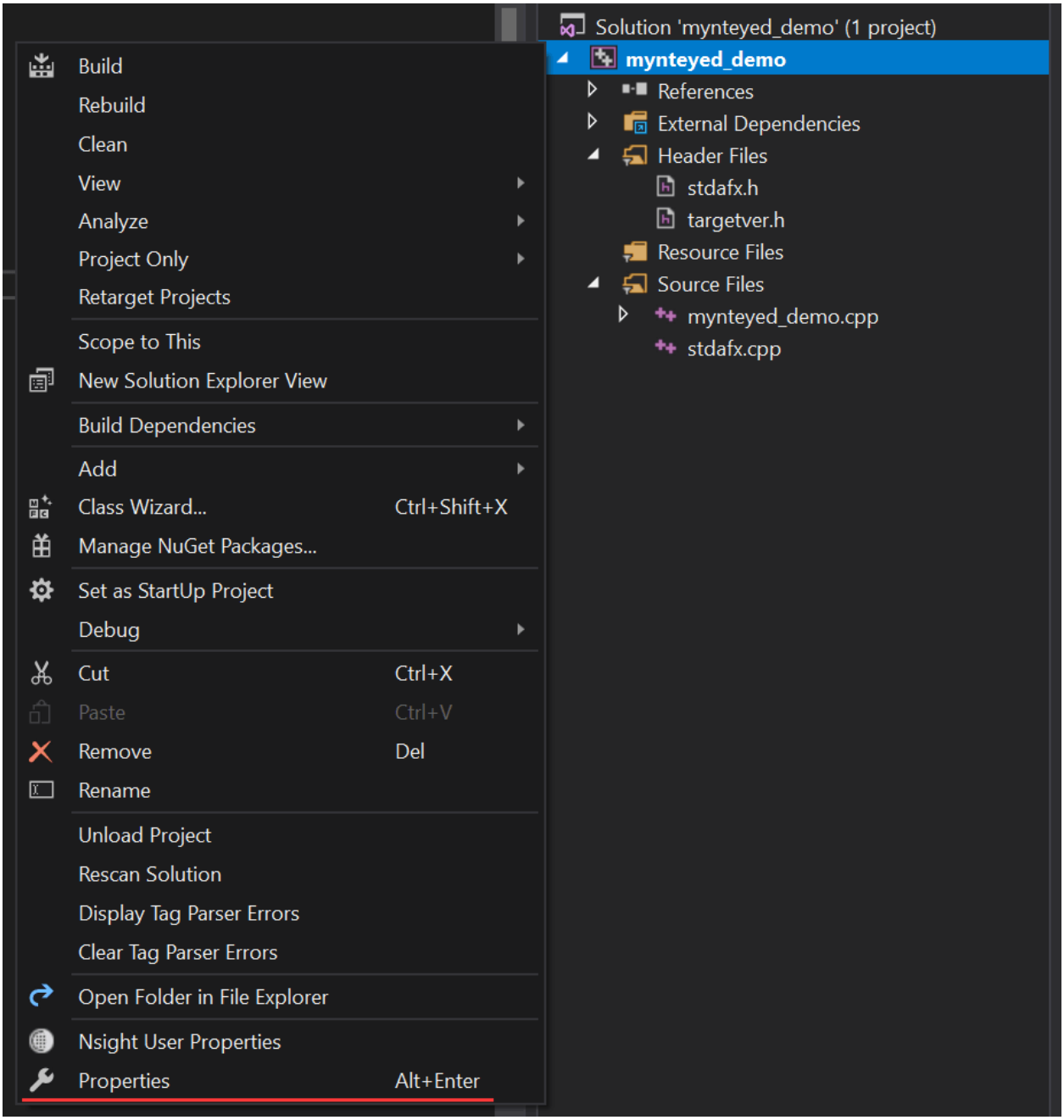
Finally, you will see the new project like this,

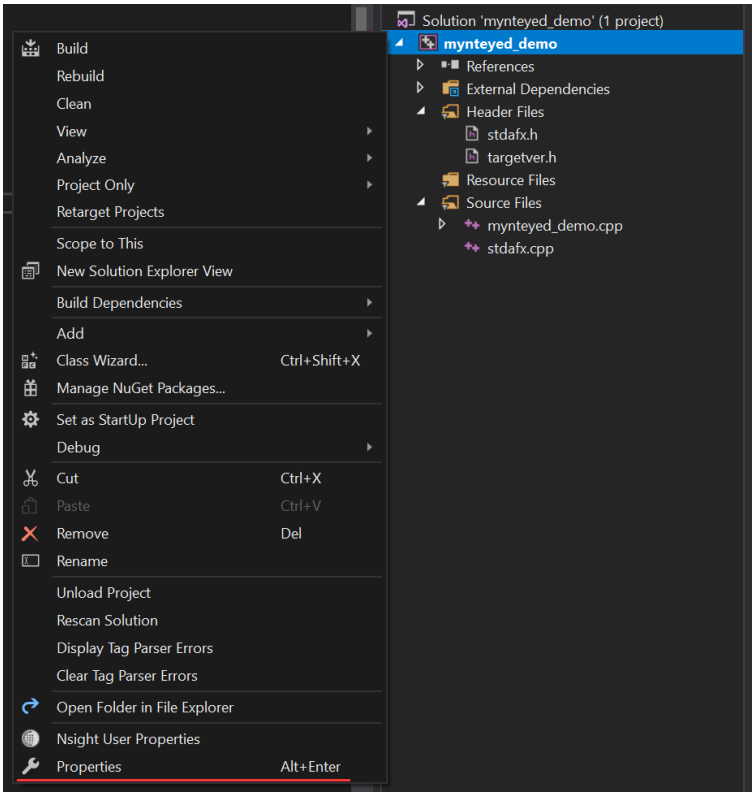




5.1.3 Config Properties

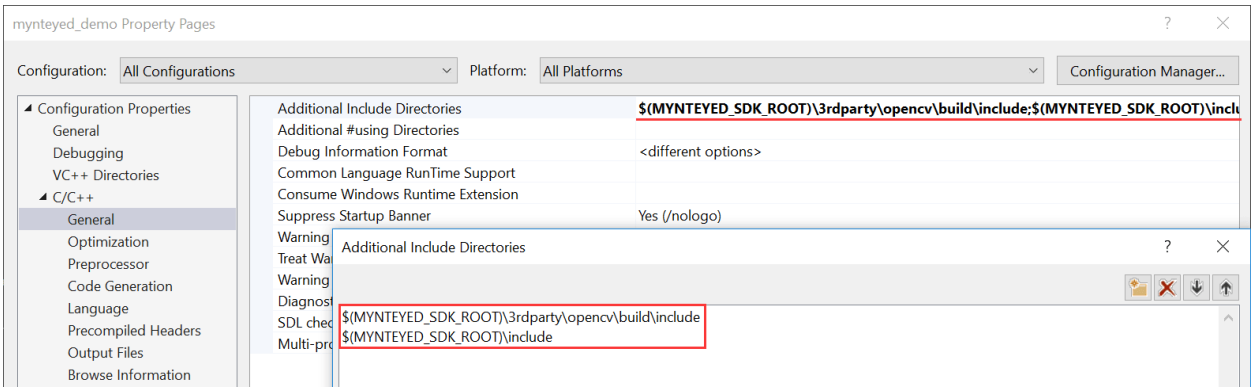
Right click the project, and open its “Properties” window,

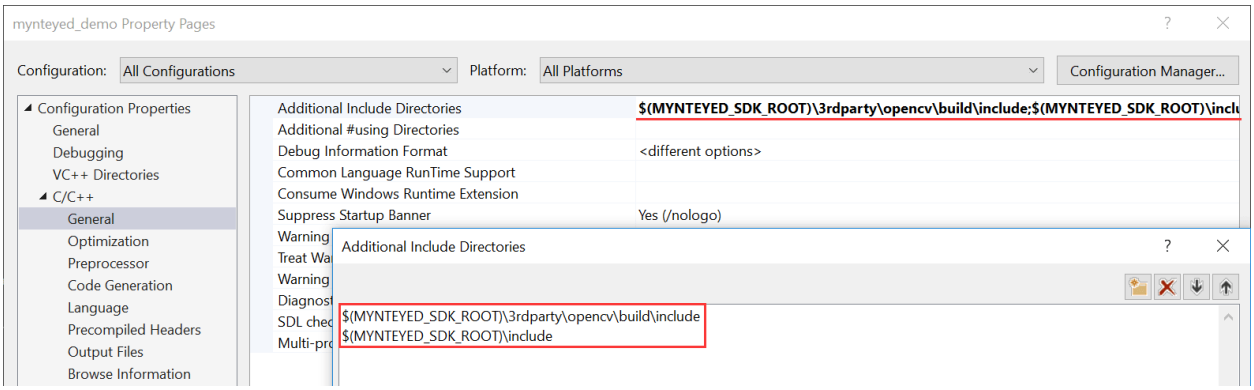




Change “Configuration” to “All Configurations”, then add the following paths to “Additional Include Directories”,

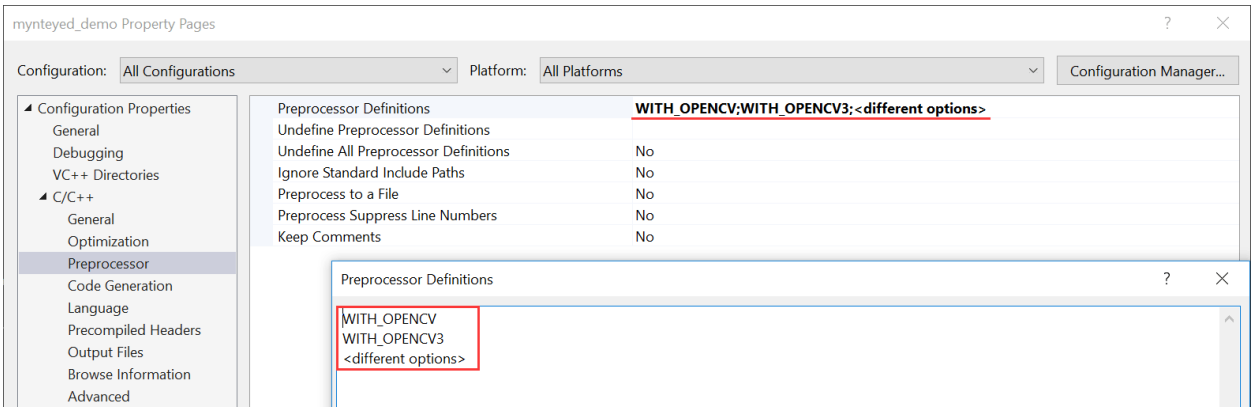
```
$(MYNTEYED_SDK_ROOT)\include
$(MYNTEYED_SDK_ROOT)\3rdparty\opencv\build\include
```

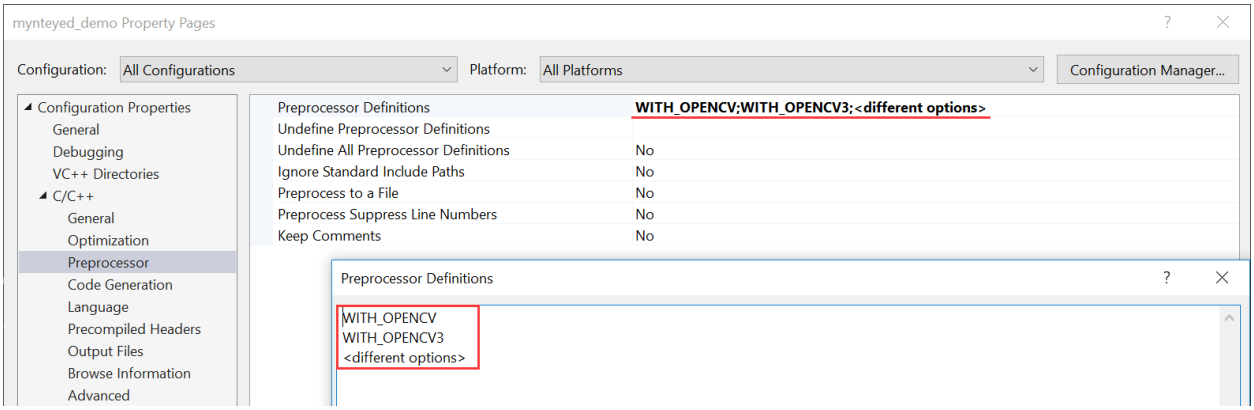




Add the following definitions to "Preprocessor Definitions",

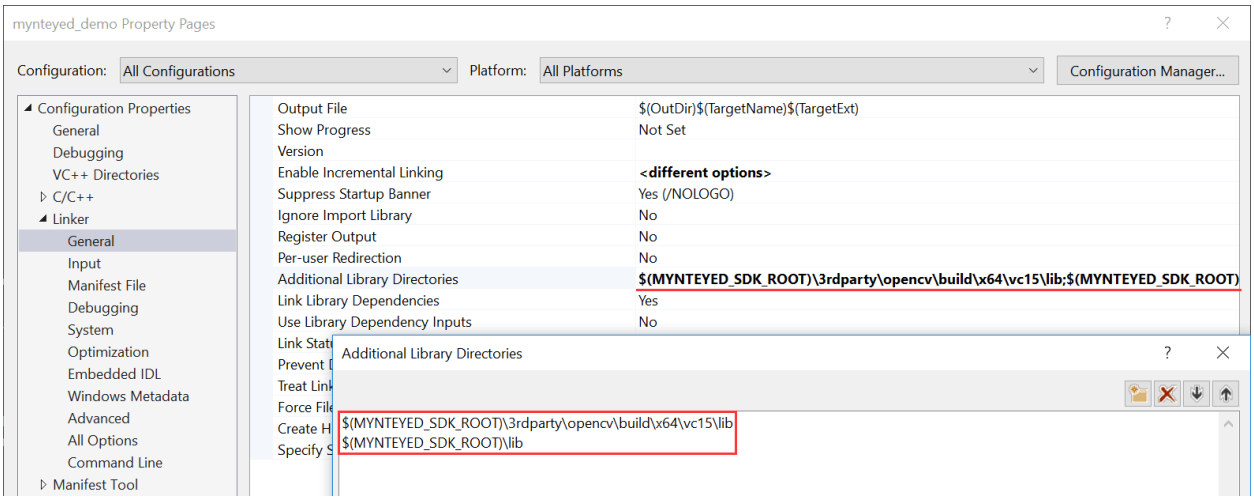
```
WITH_OPENCV
WITH_OPENCV3
```

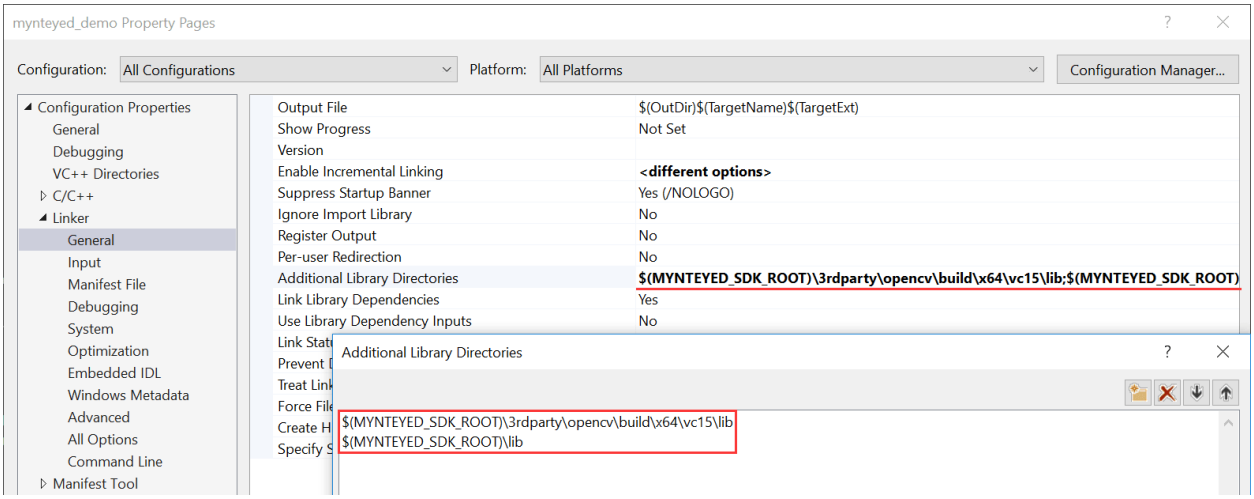




Add the following paths to "Additional Library Directories",

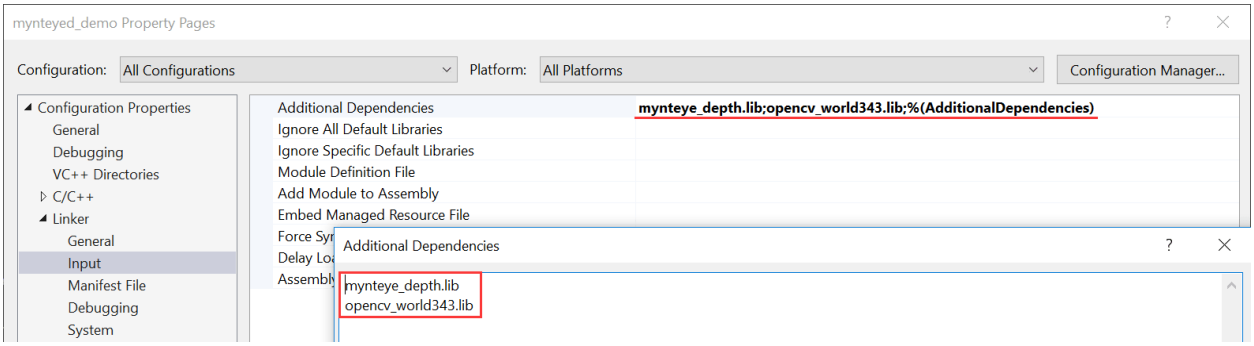
```
$(MYNTEYED_SDK_ROOT)\lib
$(MYNTEYED_SDK_ROOT)\3rdparty\opencv\build\x64\vc15\lib
```

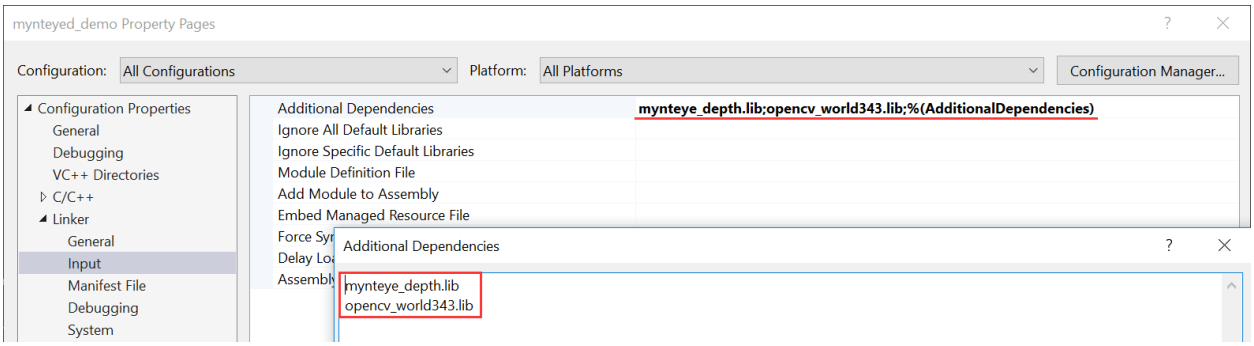




Add the following libs to “Additional Dependencies”,

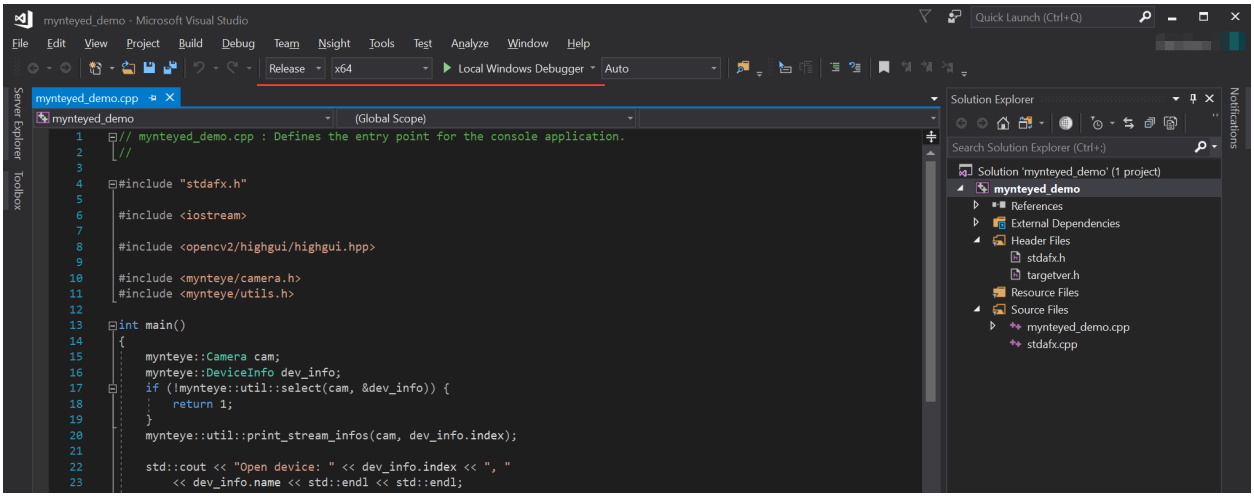
```
mynteye_depth.lib
opencv_world343.lib
```

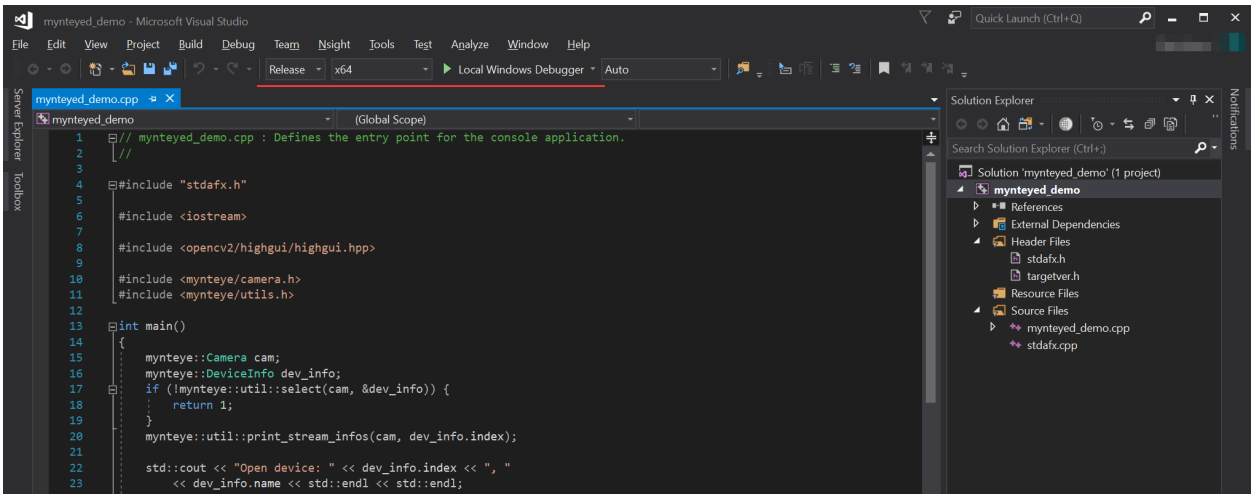




5.1.4 Start using SDK

Include the headers of SDK and start using its APIs,





Select “Release x64” to run the project.

5.2 How to use SDK with Qt Creator

This tutorial will create a Qt project with Qt Creator to start using SDK.

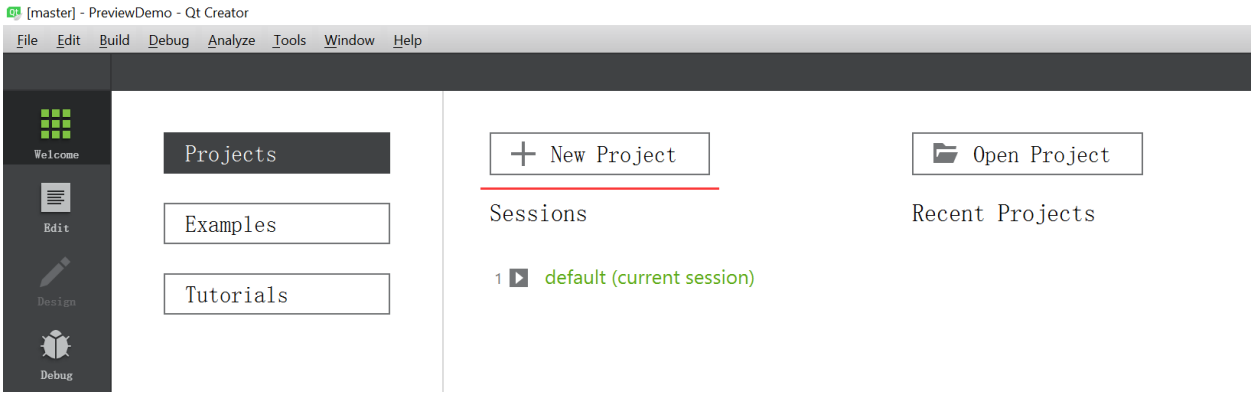
> You could find the project demo in `<sdk>/platforms/projects/qtcreator` directory.

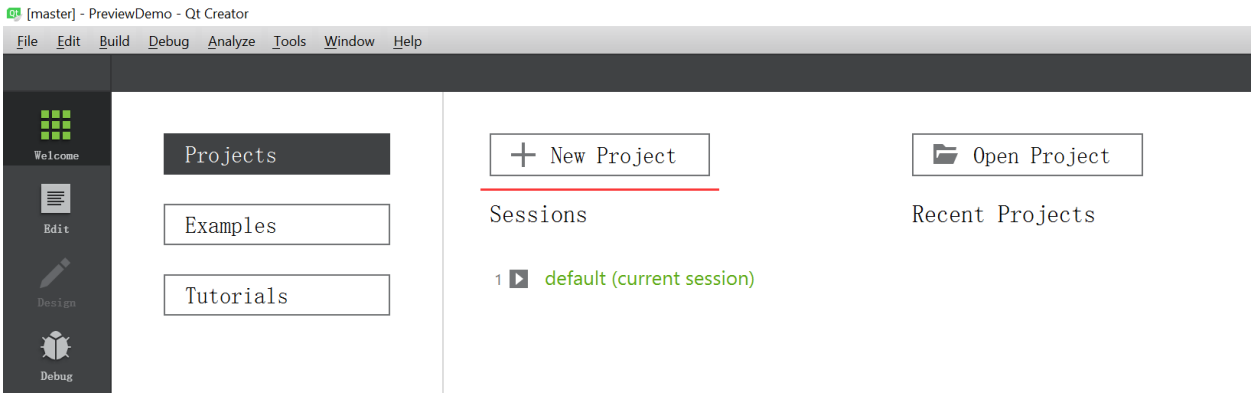
5.2.1 Preparation

- Windows: install the win pack of SDK
- Linux: build from source and `make install`

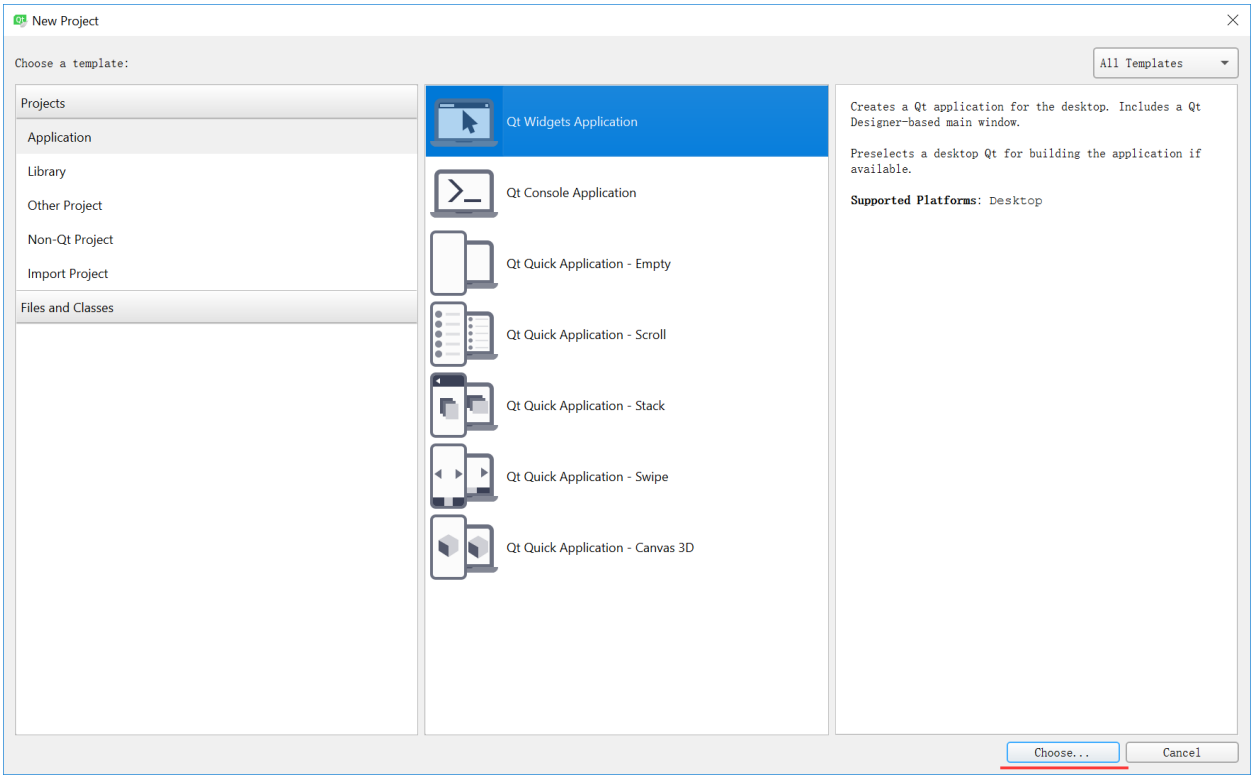
5.2.2 Create Project

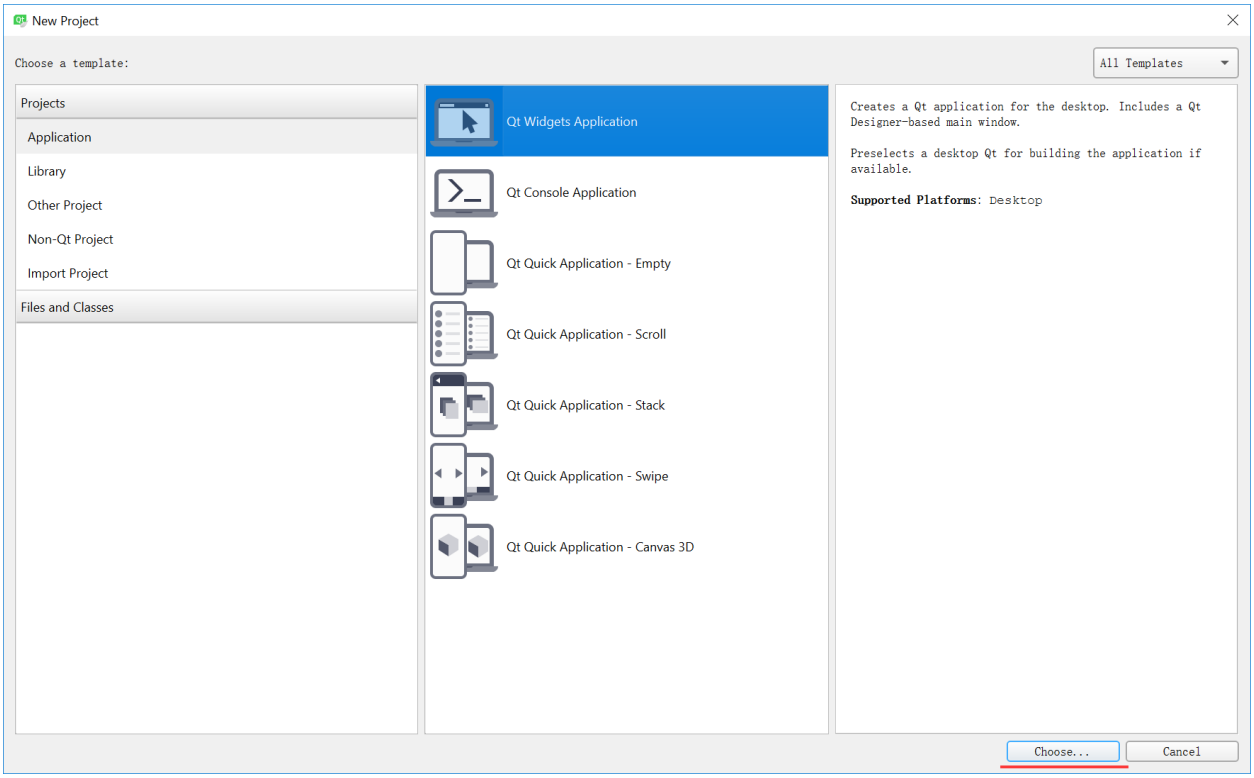
Open Qt Creator, then `New Project`,



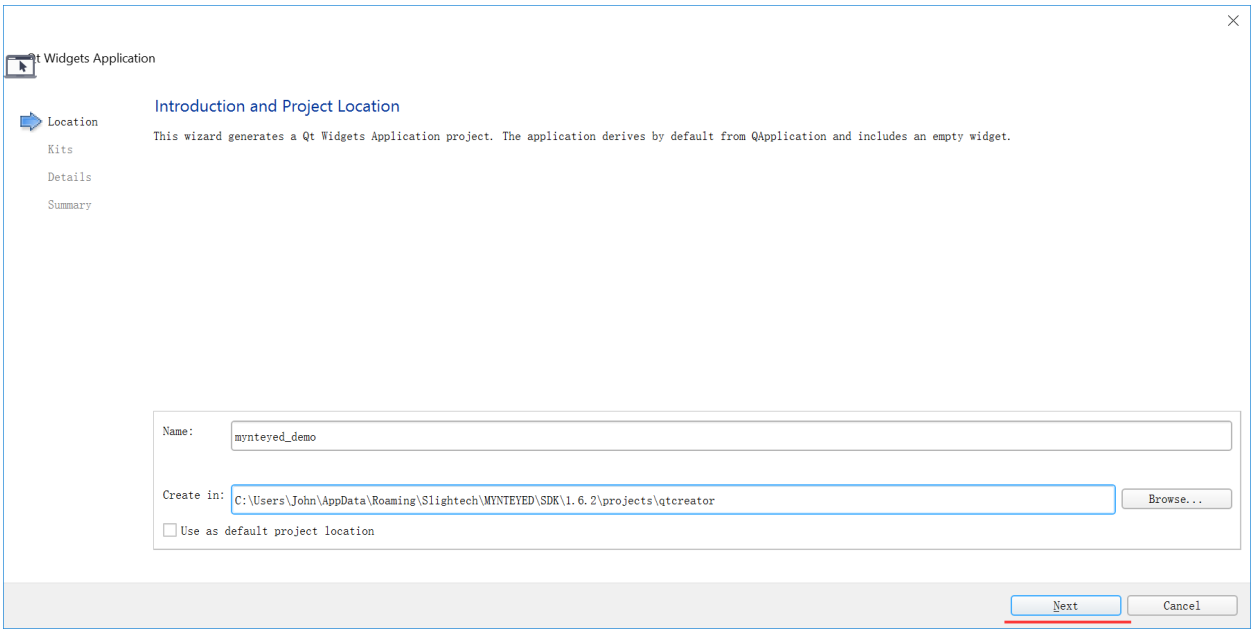


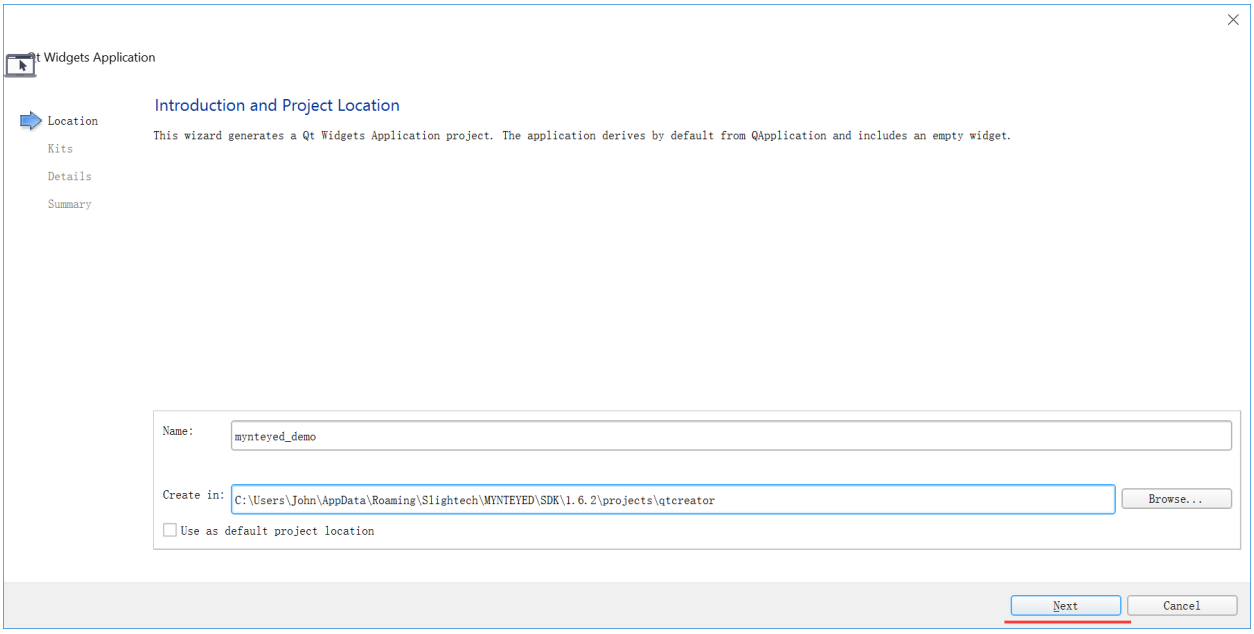
Choose `Qt Widgets Application`,



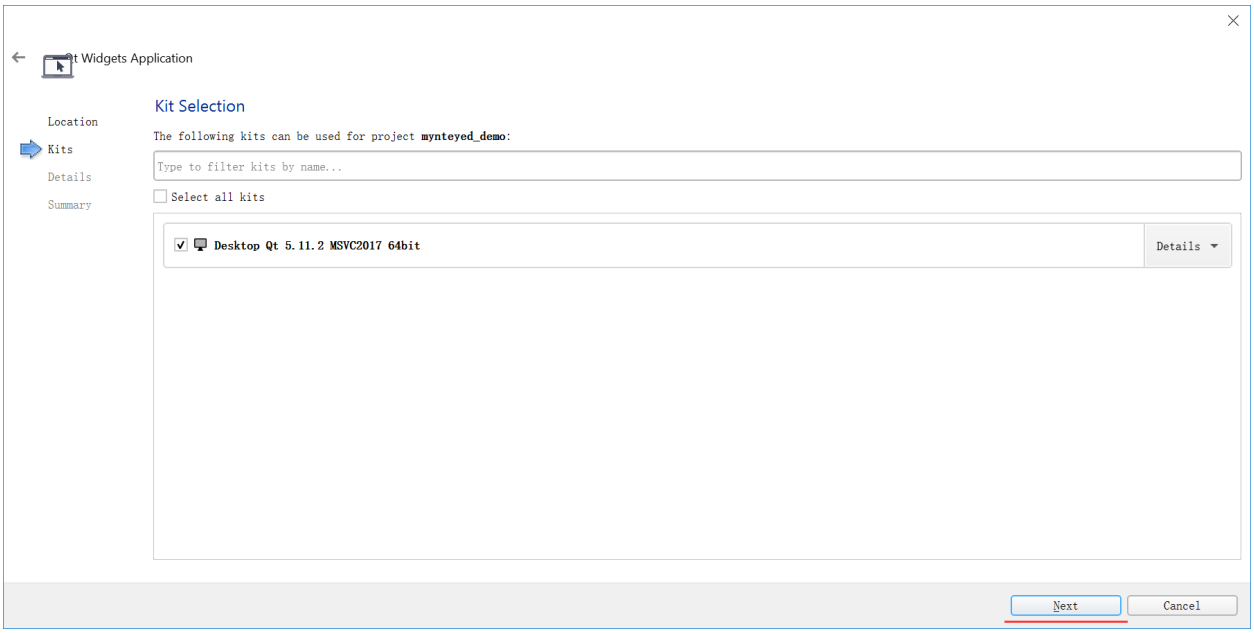


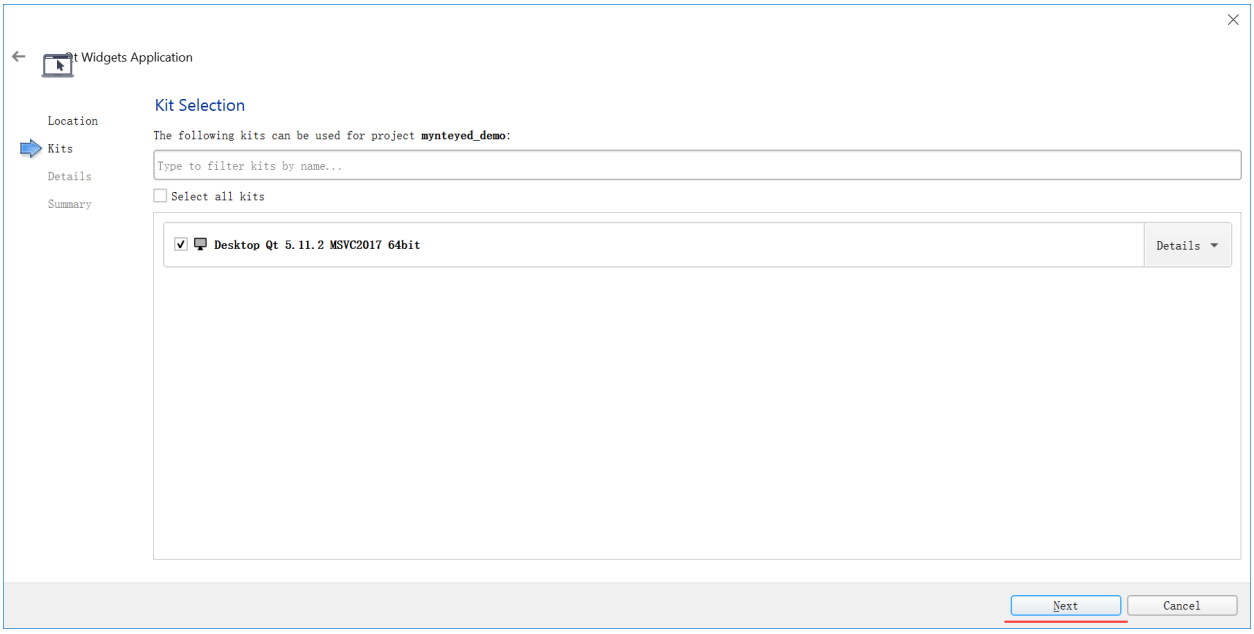
Set project location and its name,



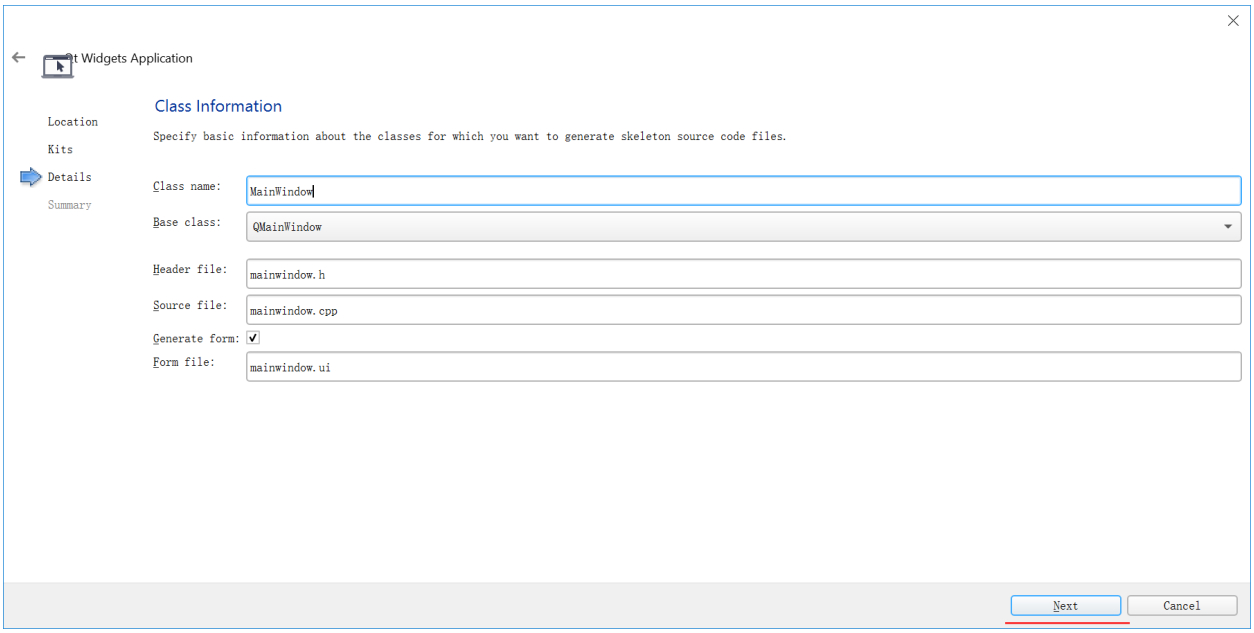


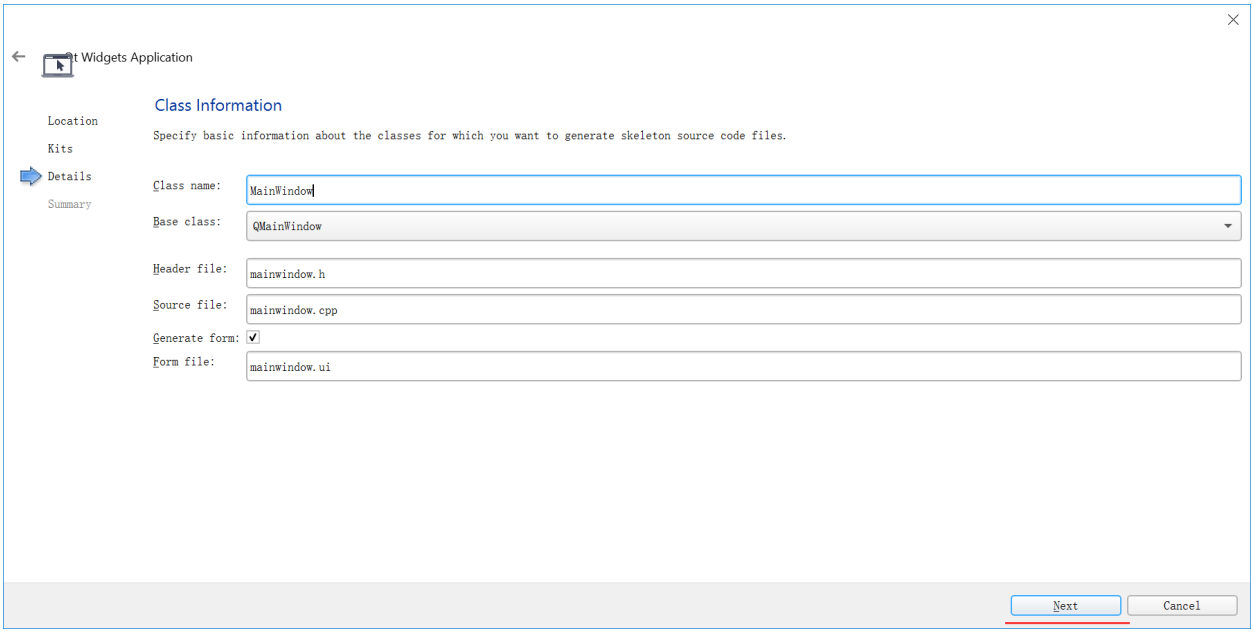
Select the build kits,

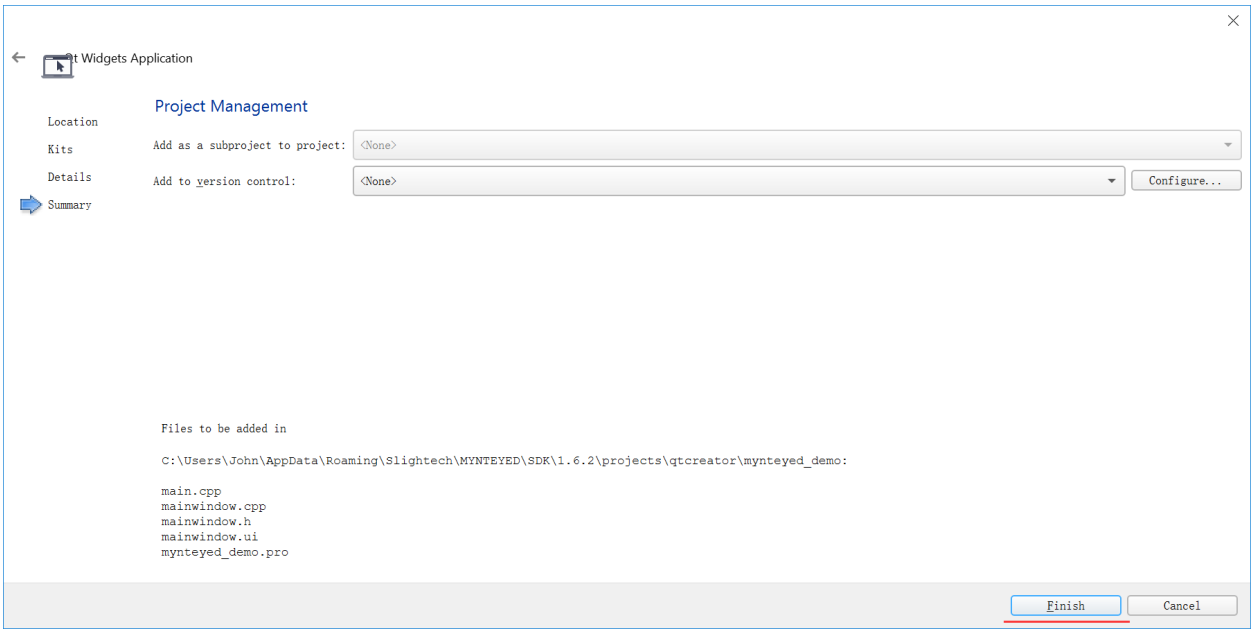


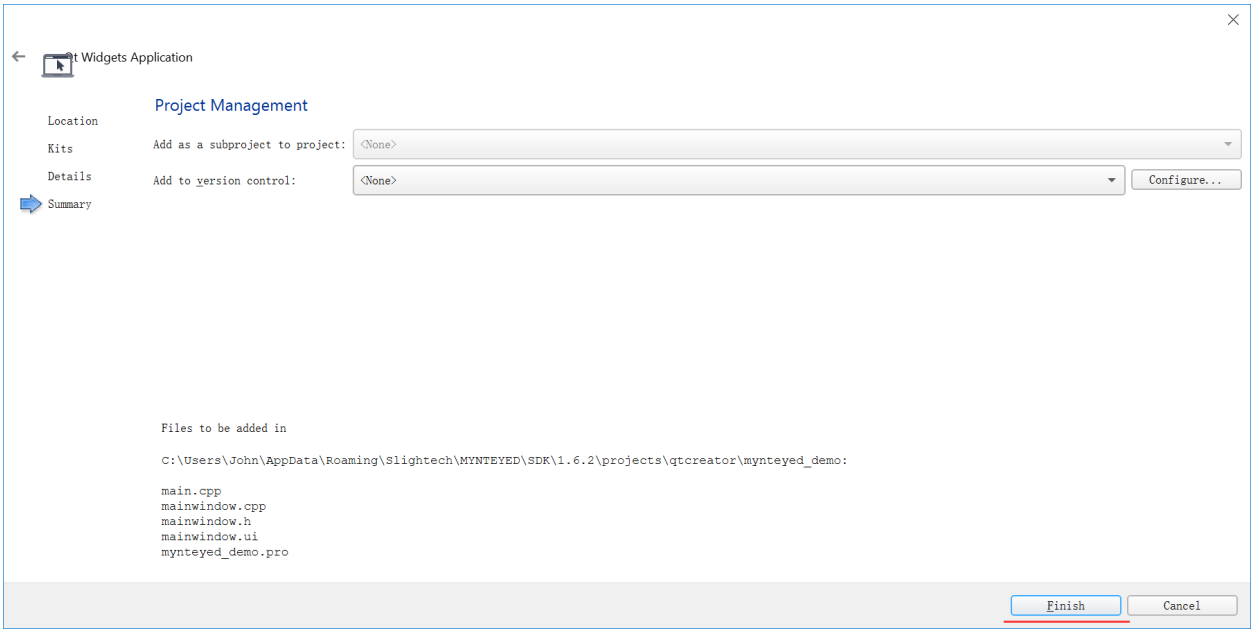


Then, it will generate the skeleton source files,

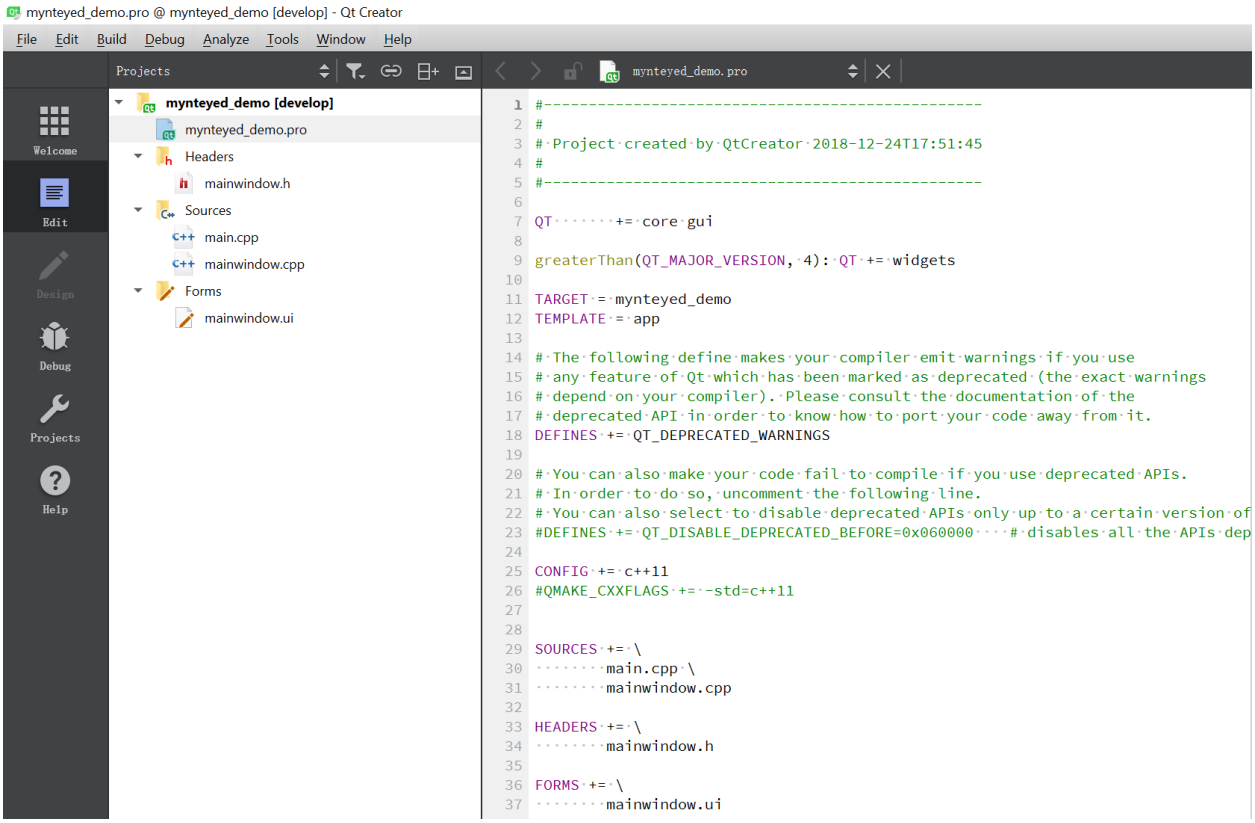

t Widgets Application
Location
Kits
Details
Summary
Class Information
Specify basic information about the classes for which you want to generate skeleton source code files.
Class name:
MainWindow
Base class:
QMainWindow
Header file:
mainwindow.h
Source file:
mainwindow.cpp
Generate form:
Form file:
mainwindow.ui
Next
Cancel

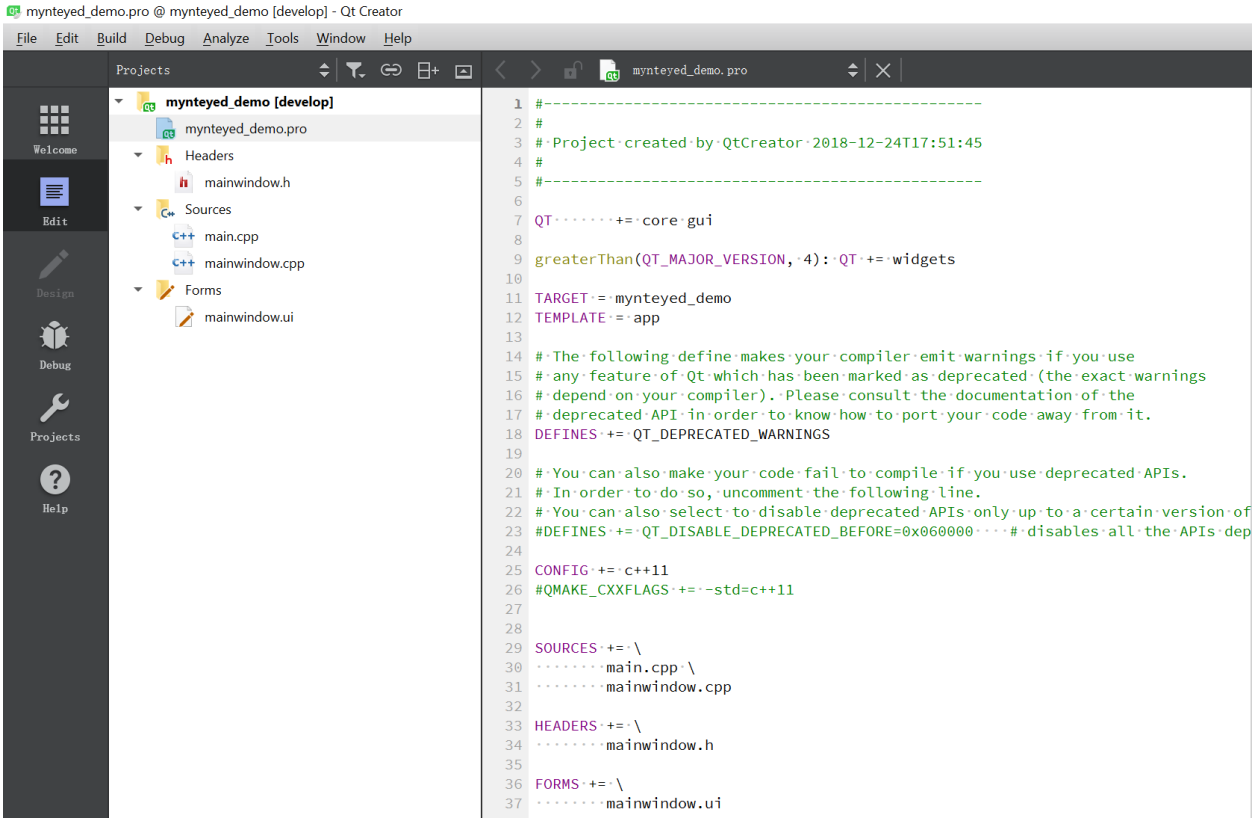

t Widgets Application
Location
Kits
Details
Summary
Class Information
Specify basic information about the classes for which you want to generate skeleton source code files.
Class name:
MainWindow
Base class:
QMainWindow
Header file:
mainwindow.h
Source file:
mainwindow.cpp
Generate form:
Form file:
mainwindow.ui
Next
Cancel





Finally, you will see the new project like this,


mynteyed_demo.pro @ mynteyed_demo [develop] - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Projects
mynteyed_demo.pro
mynteyed_demo [develop]
mynteyed_demo.pro
Headers
mainwindow.h
Sources
main.cpp
mainwindow.cpp
Forms
mainwindow.ui
Welcome
Edit
Design
Debug
Projects
Help
#---
#
Project created by QtCreator 2018-12-24T17:51:45
#
#---
QT += core gui
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
TARGET = mynteyed_demo
TEMPLATE = app
# The following define makes your compiler emit warnings if you use
# any feature of Qt which has been marked as deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS
You can also make your code fail to compile if you use deprecated APIs.
In order to do so, uncomment the following line.
You can also select to disable deprecated APIs only up to a certain version of
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000 # disables all the APIs dep
CONFIG += c++11
#QMAKE_CXXFLAGS += -std=c++11
SOURCES += \
main.cpp \
mainwindow.cpp
HEADERS += \
mainwindow.h
FORMS += \
mainwindow.ui


mynteyed_demo.pro @ mynteyed_demo [develop] - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Projects
mynteyed_demo.pro
mynteyed_demo [develop]
mynteyed_demo.pro
Headers
mainwindow.h
Sources
main.cpp
mainwindow.cpp
Forms
mainwindow.ui
Welcome
Edit
Design
Debug
Projects
Help
#---
#
Project created by QtCreator 2018-12-24T17:51:45
#
#---
QT += core gui
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
TARGET = mynteyed_demo
TEMPLATE = app
# The following define makes your compiler emit warnings if you use
# any feature of Qt which has been marked as deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS
You can also make your code fail to compile if you use deprecated APIs.
In order to do so, uncomment the following line.
You can also select to disable deprecated APIs only up to a certain version of
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000 # disables all the APIs dep
CONFIG += c++11
#QMAKE_CXXFLAGS += -std=c++11
SOURCES += \
main.cpp \
mainwindow.cpp
HEADERS += \
mainwindow.h
FORMS += \
mainwindow.ui

5.2.3 Config Project

Edit `mynteyed_demo.pro` to add `INCLUDEPATH` and `LIBS`.

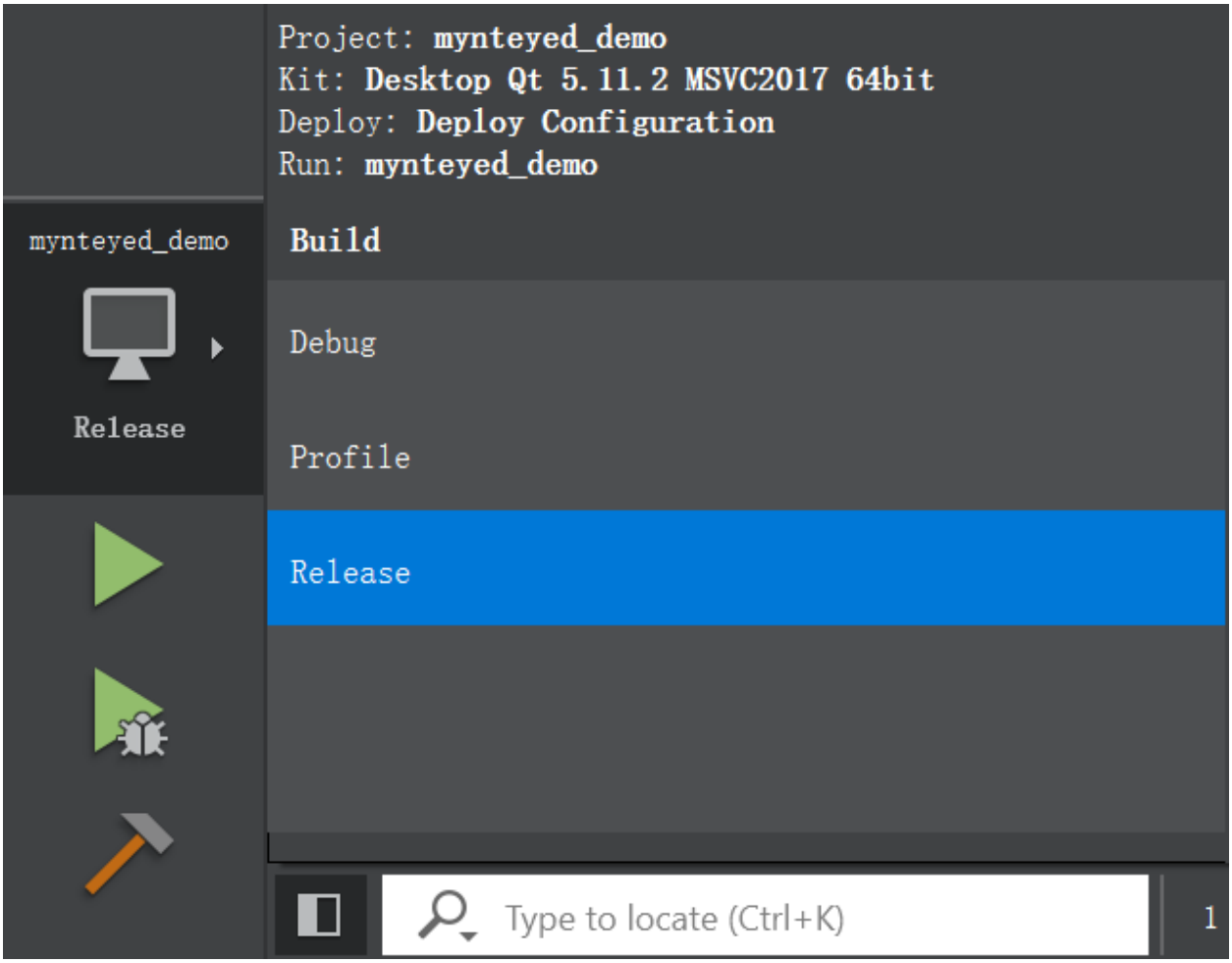
```
win32 {
    SDK_ROOT = "$$(MYNTEYED_SDK_ROOT)"
    isEmpty(SDK_ROOT) {
        error( "MYNTEYED_SDK_ROOT not found, please install SDK firstly" )
    }
    message("SDK_ROOT: $$SDK_ROOT")

    INCLUDEPATH += "$$SDK_ROOT/include"
    LIBS += "$$SDK_ROOT/lib/mynteye_depth.lib"
}

unix {
    INCLUDEPATH += /usr/local/include
    LIBS += -L/usr/local/lib -lmynteye_depth
}
```

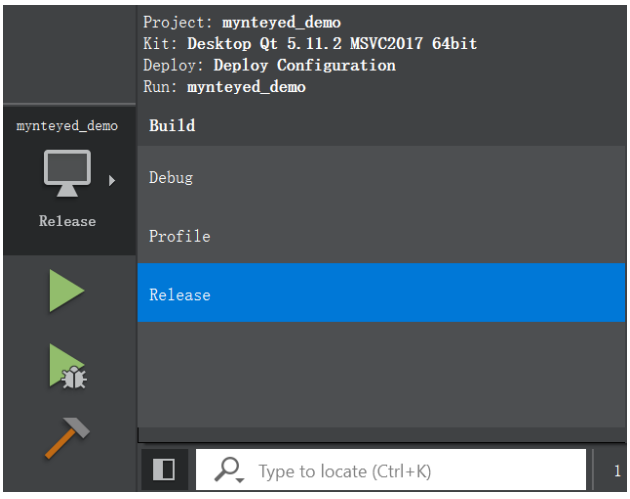
5.2.4 Start using SDK

Include the headers of SDK and start using its APIs, could see the project demo.

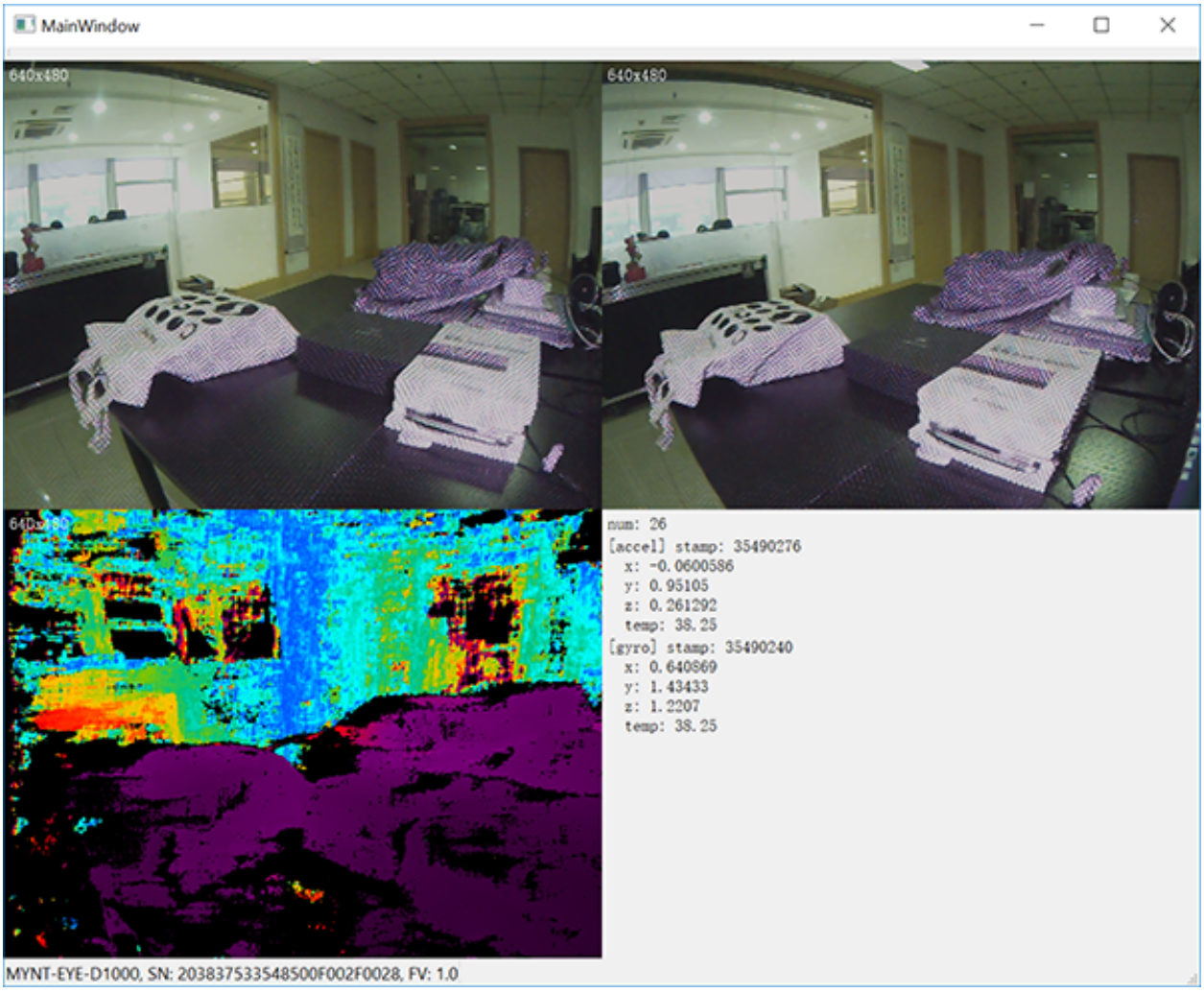
Windows

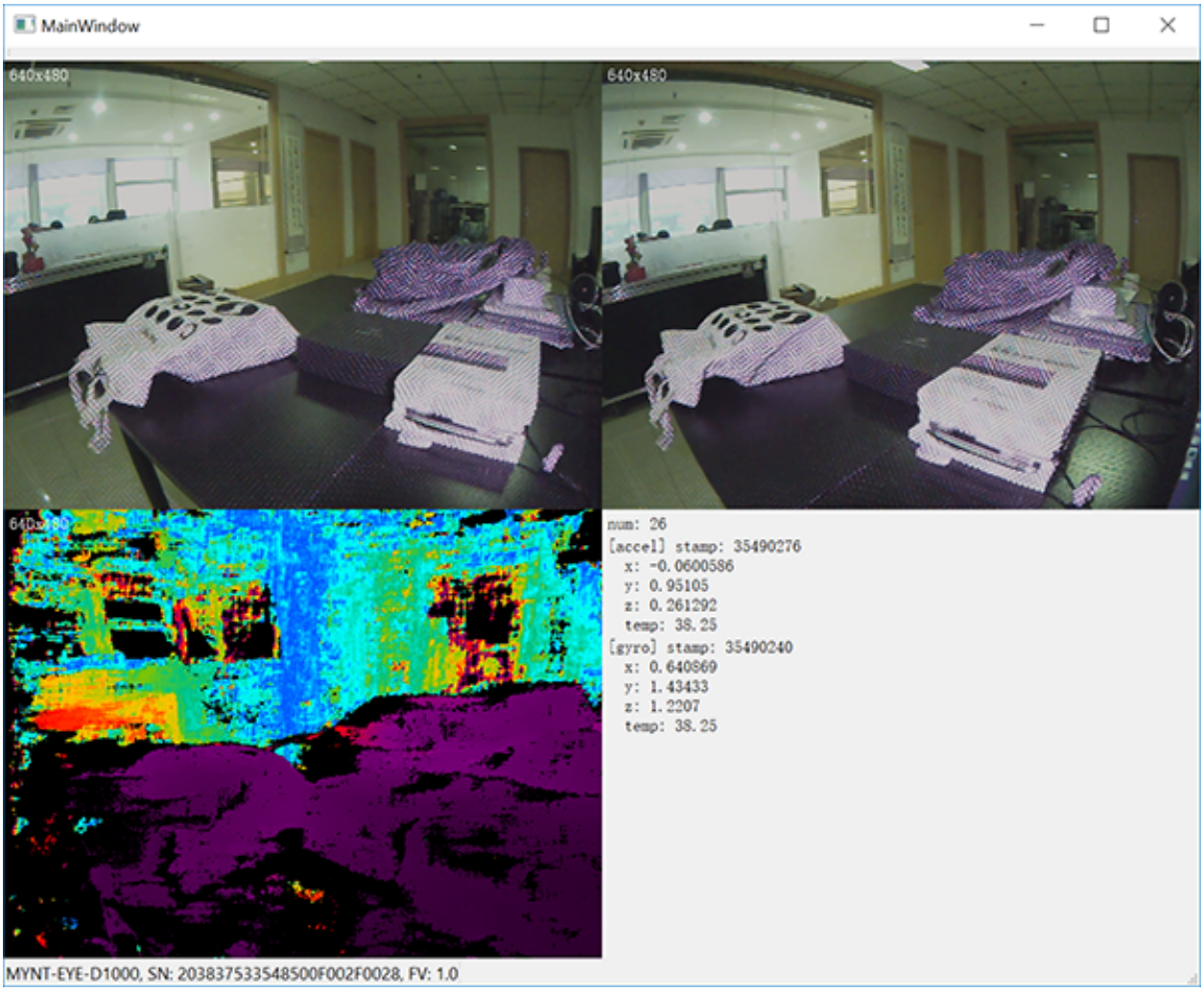
Should select “Release” to run the project.





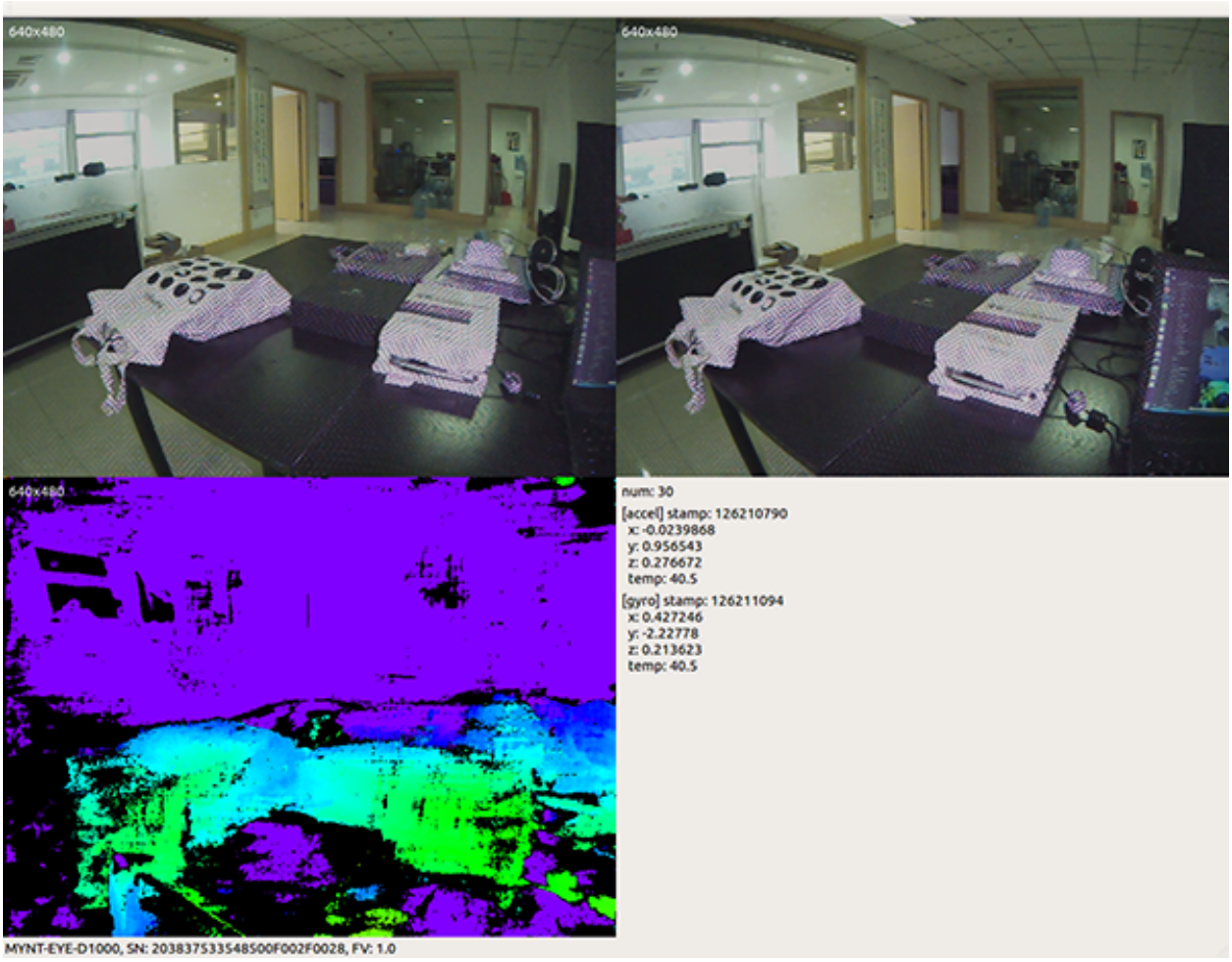
Then you will see the main window,

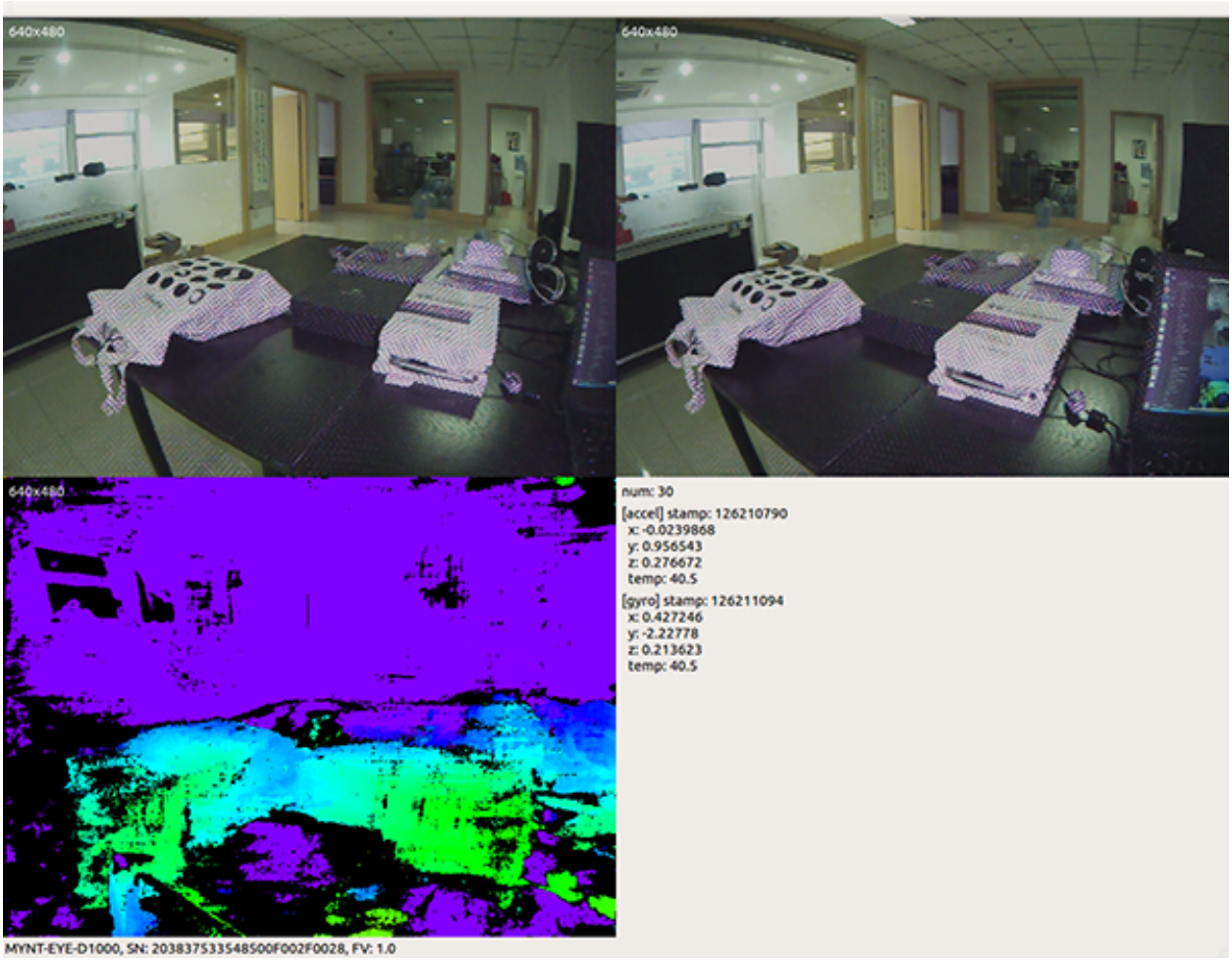

MainWindow
640x480
640x480
640x480
num: 26
[accel] stamp: 35490276
x: -0.0600586
y: 0.95105
z: 0.261292
temp: 38.25
[gyro] stamp: 35490240
x: 0.640869
y: 1.43433
z: 1.2207
temp: 38.25
MYNT-EYE-D1000, SN: 203837533548500F002F0028, FV: 1.0



Linux

Run the project and you will see the main window,


640x480
640x480
640x480
num: 30
[accel] stamp: 126210790
x: -0.0239868
y: 0.956543
z: 0.276672
temp: 40.5
[gyro] stamp: 126211094
x: 0.427246
y: -2.22778
z: 0.213623
temp: 40.5
MYNT-EYE-D1000, SN: 2038375333548500F002F0028, FV: 1.0



5.3 How to use SDK with CMake

This tutorial will create a project with CMake to start using SDK.

> You could find the project demo in `<sdk>/platforms/projects/cmake` directory.

5.3.1 Preparation

- Windows: install the win pack of SDK
- Linux: build from source and `make install`

5.3.2 Create Project

Add `CMakeLists.txt` and `mynteyed_demo.cc` files,

```
cmake_minimum_required(VERSION 3.0)

project(mynteyed_demo VERSION 1.0.0 LANGUAGES C CXX)

```

(continues on next page)

(continued from previous page)

```
# flags

set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wall -O3")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -O3")

set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -std=c++11 -march=native")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11 -march=native")

## mynteyed_demo

add_executable(mynteyed_demo mynteyed_demo.cc)
```

5.3.3 Config Project

Add `mynteyed` and `OpenCV` packages to `CMakeLists.txt`,

```
# packages

if(MSVC)
  set(SDK_ROOT "$ENV{MYNTEYED_SDK_ROOT}")
  if(SDK_ROOT)
    message(STATUS "MYNTEYED_SDK_ROOT: ${SDK_ROOT}")
    list(APPEND CMAKE_PREFIX_PATH
      "${SDK_ROOT}/lib/cmake"
      "${SDK_ROOT}/3rdparty/opencv/build"
    )
  else()
    message(FATAL_ERROR "MYNTEYED_SDK_ROOT not found, please install SDK firstly")
  endif()
endif()

## mynteyed

find_package(mynteyed REQUIRED)
message(STATUS "Found mynteye: ${mynteyed_VERSION}")

# When SDK build with OpenCV, we can add WITH_OPENCV macro to enable some
# features depending on OpenCV, such as ToMat().
if(mynteyed_WITH_OPENCV)
  add_definitions(-DWITH_OPENCV)
endif()

## OpenCV

# Set where to find OpenCV
#set(OpenCV_DIR "/usr/share/OpenCV")

# When SDK build with OpenCV, we must find the same version here.
find_package(OpenCV REQUIRED)
message(STATUS "Found OpenCV: ${OpenCV_VERSION}")
```

Add `include_directories` and `target_link_libraries` to `mynteyed_demo` target,

```
# targets

```

(continues on next page)

(continued from previous page)

```
include_directories(
  ${OpenCV_INCLUDE_DIRS}
)

## mynteyed_demo

add_executable(mynteyed_demo mynteyed_demo.cc)
target_link_libraries(mynteyed_demo mynteye_depth ${OpenCV_LIBS})
```

5.3.4 Start using SDK

Include the headers of SDK and start using its APIs, could see the project demo.

Windows

See *Quick Start Guide for Windows* to "Install Build Tools".

Then open "x64 Native Tools Command Prompt for VS 2017" command shell to build and run,

```
mkdir _build
cd _build

cmake -G "Visual Studio 15 2017 Win64" ..

msbuild.exe ALL_BUILD.vcxproj /property:Configuration=Release

.\Release\mynteyed_demo.exe
```

Linux

Open "Terminal" to build and run,

```
mkdir _build
cd _build/

cmake ..

make

./mynteyed_demo
```

CHAPTER 6

SLAM

6.1 How to use in VINS-Mono

6.1.1 If you wanna run VINS-Mono with MYNT EYE camera, please follow the steps:

1. Download MYNT-EYE-D-SDK and *ROS Installation*.
2. Follow the normal procedure to install VINS-Mono.
3. Update `distortion_parameters` and `projection_parameters` to here .
4. Run mynteye_wrapper_d and VINS-Mono.

6.1.2 Install ROS Kinetic conveniently (if already installed, please ignore)

```
cd ~
wget https://raw.githubusercontent.com/oroca/oroca-ros-pkg/master/ros_install.sh && \
chmod 755 ./ros_install.sh && bash ./ros_install.sh catkin_ws kinetic
```

6.1.3 Install MYNT-EYE-VINS-Sample

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
git clone -b mynteye https://github.com/slightech/MYNT-EYE-VINS-Sample.git
cd ..
catkin_make
source devel/setup.bash
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

6.1.4 Get image calibration parameters

Use MYNT® EYE's left eye camera and IMU. By MYNT-EYE-D-SDK API `GetIntrinsics()` function and `GetExtrinsics()` function, you can "get the image calibration parameters of the current working device:

```
cd MYNT-EYE-D-SDK
./samples/_output/bin/get_img_params
```

After running the above type, pinhole's `distortion_parameters` and `projection_parameters` is obtained , and then update to here .

6.1.5 Run VINS-Mono with MYNT® EYE

1. Launch mynteye node

```
cd (local path of MYNT-EYE-D-SDK)
source ./wrappers/ros/devel/setup.bash
roslaunch mynteye_wrapper_d mynteye.launch
```

2. Open another terminal and run vins

```
cd ~/catkin_ws
roslaunch vins_estimator mynteye_d.launch
```

6.2 How to use in ORB_SLAM2

6.2.1 If you wanna run ORB_SLAM2 with MYNT EYE camera, please follow the steps:

1. Download MYNT-EYE-D-SDK and *ROS Installation*.
2. Follow the normal procedure to install ORB_SLAM2.
3. Update `distortion_parameters` and `projection_parameters` to `<ORB_SLAM2>/config/mynteye_*.yaml`.
4. Run examples by MYNT® EYE.

6.2.2 Building the nodes for stereo (ROS)

- Add the path including `Examples/ROS/ORB_SLAM2` to the `ROS_PACKAGE_PATH` environment variable. Open .bashrc file and add at the end the following line. Replace PATH by the folder where you cloned ORB_SLAM2:

```
export ROS_PACKAGE_PATH=${ROS_PACKAGE_PATH}:PATH/ORB_SLAM2/Examples/ROS
```

- Execute *build_ros.sh*:

```
chmod +x build_ros.sh
./build_ros.sh
```

Stereo_ROS Example

- Reference `Get camera calibration parameters` in *How to use in OKVIS* to get `distortion_parameters` and `projection_parameters`, and update `<ORB_SLAM2>/config/mynteye_d_stereo.yaml`.
- Launch ORB_SLAM2 `Stereo_ROS`

Run camera `mynteye_wrapper_d`

```
cd [path of mynteye-d-sdk]
make ros
source ./wrappers/ros/devel/setup.bash
roslaunch mynteye_wrapper_d mynteye.launch
```

Open another terminal and run ORB_SLAM2

```
rosrun ORB_SLAM2 mynteye_d_stereo ./Vocabulary/ORBvoc.txt ./config/mynteye_d_stereo.
↪yaml true /mynteye/left/image_mono /mynteye/right/image_mono
```

6.3 How to use in OKVIS

6.3.1 If you wanna run OKVIS with MYNT EYE camera, please follow the steps:

1. Download MYNT-EYE-D-SDK and *ROS Installation*.
2. Install dependencies and build MYNT-EYE-OKVIS-Sample follow the procedure of the original OKVIS.
3. Update camera parameters to `<OKVIS>/config/config_mynteye.yaml`.
4. Run OKVIS using MYNT® EYE.

6.3.2 Install MYNTEYE OKVIS

First install dependencies based on the original OKVIS, and the follow:

```
sudo apt-get install libgoogle-glog-dev

git clone -b mynteye https://github.com/slightech/MYNT-EYE-OKVIS-Sample.git
cd MYNT-EYE-OKVIS-Sample/
mkdir build && cd build
cmake ..
make
```

6.3.3 Get camera calibration parameters

Through the `GetIntrinsics()` and `GetExtrinsics()` function of the MYNT-EYE-D-SDK API, you can get the camera calibration parameters of the currently open device, follow the steps:

```
cd MYNT-EYE-D-SDK
./samples/_output/bin/get_img_params
```

After running the above type, pinhole's `distortion_parameters` and `camera parameters` is obtained, and then update to here .

according to following format. It should be noted that only first four parameters of coeffs need to be filled in the distortion_coefficients.

```
distortion_coefficients: [coeffs]   # only first four parameters of coeffs need to be␣
↪filled
focal_length: [fx, fy]
principal_point: [cx, cy]
distortion_type: radialtangential
```

6.3.4 Run MYNTEYE OKVIS

Run camera `mynteye_wrapper_d`

```
cd MYNT-EYE-D-SDK
source wrappers/ros/devel/setup.bash
roslaunch mynteye_wrapper_d mynteye.launch
```

Run `MYNT-EYE-OKVIS-Sample` open another terminal and follow the steps.

```
cd MYNT-EYE-OKVIS-Sample/build
source devel/setup.bash
roslaunch okvis_ros mynteye_d.launch
```

And use rviz to display

```
cd ~/catkin_okvis/src/MYNT-EYE-OKVIS-Sample/config
rosrun rviz rviz -d rviz.rvi
```

6.4 How to use in VIORB

6.4.1 If you wanna run VIORB with MYNT® EYEplease follow the steps:

1. Download MYNT-EYE-D-SDK and *ROS Installation*.
2. Follow the normal procedure to install VIORB.
3. Update camera parameters to `<VIO>/config/mynteye_d.yaml`.
4. Run mynteye_wrapper_d and VIORB.

6.4.2 Install MYNT-EYE-VIORB-Sample.

```
git clone -b mynteye https://github.com/slightech/MYNT-EYE-VIORB-Sample.git
cd MYNT-EYE-VIORB-Sample
```

`ROS_PACKAGE_PATH` environment variable. Open `.bashrc` file and add at the end the following line. Replace `PATH` by the folder where you cloned `MYNT-EYE-VIORB-Sample`:

```
export ROS_PACKAGE_PATH=${ROS_PACKAGE_PATH}:PATH/Examples/ROS/ORB_VIO
```

Execute:

```
cd MYNT-EYE-VIORB-Sample
./build.sh
```

6.4.3 Get image calibration parameters

Assume that the left eye of the mynteye camera is used with IMU. Through the `GetIntrinsics()` and `GetExtrinsics()` function of the MYNT-EYE-D-SDK API, you can get the image calibration parameters of the currently open device:

```
cd MYNT-EYE-S-SDK
./samples/_output/bin/get_img_params
```

After running the above type, pinhole's `distortion_parameters` and `projection_parameters` is obtained, and then update to `<MYNT-EYE-VIORB-Sample>/config/mynteye_d.yaml`.

6.4.4 Run VIORB and mynteye_wrapper_d

1. Launch mynteye node

```
roslaunch mynteye_wrapper_d mynteye.launch
```

2. Open another terminal and run viorb

```
roslaunch ORB_VIO testmynteye_d.launch
```

Finally, `pyplotscripts` can be used to visualize some results.

6.5 How to use in VINS-Fusion

6.5.1 If you wanna run VINS-Fusion with MYNT EYE camera, please follow the steps

1. Download MYNT-EYE-D-SDK and *ROS Installation*
2. Follow the normal procedure to install VINS-Fusion
3. Run mynteye_wrapper_d and VINS-Fusion

6.5.2 Preparation

1. Install Ubuntu 64 16.04/18.04. ROS Kinetic/Melodic(If you have installed ROS, you can skip this part). ROS Installation 2. Install Ceres Installation

6.5.3 Build VINS-Fusion

Clone the repository and catkin_make:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
git clone -b mynteye https://github.com/slightech/MYNT-EYE-VINS-FUSION-Samples.git
cd ..
catkin_make
source ~/catkin_ws/devel/setup.bash
```

(if you fail in this step, try to find another computer with clean system or reinstall Ubuntu and ROS)

6.5.4 MYNT® EYE VINS-FUSION

1.Launch mynteye node

```
cd (local path of MYNT-EYE-D-SDK)
source ./wrappers/ros/devel/setup.bash
roslaunch mynteye_wrapper_d mynteye.launch
```

2.Open another terminal and run vins-fusion

```
cd ~/catkin_ws
roslaunch vins mynteye-d-mono-imu.launch  # mono+imu fusion
# roslaunch vins mynteye-d-stereo.launch  # Stereo fusion / Stereo+imu fusion
```

CHAPTER 7

API REFERENCE

7.1 Camera

class Camera

Public Functions

std::vector<*DeviceInfo*> **GetDeviceInfos**() **const**
Get all device infos.

void **GetDeviceInfos**(std::vector<*DeviceInfo*> **dev_infos*) **const**
Get all device infos.

void **GetStreamInfos**(**const** std::int32_t *&dev_index*, std::vector<*StreamInfo*> **color_infos*, std::vector<*StreamInfo*> **depth_infos*) **const**
Get all stream infos.

ErrorCode **Open**()
Open camera.

ErrorCode **Open**(**const** *OpenParams &params*)
Open camera with params.

bool **IsOpened**() **const**
Whethor camera is opened or not.

std::shared_ptr<device::Descriptors> **GetDescriptors**() **const**
Get all device descriptors.

std::string **GetDescriptor**(**const** *Descriptor &desc*) **const**
Get one device descriptor.

StreamIntrinsics **GetStreamIntrinsics**(**const** *StreamMode &stream_mode*) **const**
Get the intrinsics of camera.

StreamIntrinsics **GetStreamIntrinsics**(**const** *StreamMode* &*stream_mode*, bool **ok*) **const**
Get the intrinsics of camera.

StreamExtrinsics **GetStreamExtrinsics**(**const** *StreamMode* &*stream_mode*) **const**
Get the extrinsics of camera.

StreamExtrinsics **GetStreamExtrinsics**(**const** *StreamMode* &*stream_mode*, bool **ok*) **const**
Get the extrinsics of camera.

bool **WriteCameraCalibrationBinFile**(**const** std::string &*filename*)
Write camera calibration bin file.

MotionIntrinsics **GetMotionIntrinsics**() **const**
Get the intrinsics of motion.

MotionIntrinsics **GetMotionIntrinsics**(bool **ok*) **const**
Get the intrinsics of motion.

MotionExtrinsics **GetMotionExtrinsics**() **const**
Get the extrinsics from left to motion.

MotionExtrinsics **GetMotionExtrinsics**(bool **ok*) **const**
Get the extrinsics from left to motion.

bool **IsWriteDeviceSupported**() **const**
Whethor write device supported or not.

bool **WriteDeviceFlash**(device::Descriptors **desc*, device::ImuParams **imu_params*, Version **spec_version* = nullptr)
Write device flash.

void **EnableProcessMode**(**const** *ProcessMode* &*mode*)
Enable process mode, e.g.

imu assembly, temp_drift

void **EnableProcessMode**(**const** std::int32_t &*mode*)
Enable process mode, e.g.

imu assembly, temp_drift

bool **IsImageInfoSupported**() **const**
Whethor image info supported or not.

void **EnableImageInfo**(bool *sync*)
Enable image infos.

If sync is false, indicates only can get infos from callback. If sync is true, indicates can get infos from callback or access it from *StreamData*.

void **DisableImageInfo**()
Disable image info.

bool **IsImageInfoEnabled**() **const**
Whethor image info enabled or not.

bool **IsImageInfoSynced**() **const**
Whethor image info synced or not.

bool **IsStreamDataEnabled**(**const** *ImageType* &*type*) **const**
Whethor stream data of certain image type enabled or not.

bool **HasStreamDataEnabled**() **const**
Has any stream data enabled.

StreamData **GetStreamData**(**const** *ImageType* &*type*)
Get latest stream data.

std::vector<*StreamData*> **GetStreamDatas**(**const** *ImageType* &*type*)
Get cached stream datas.

bool **IsMotionDatasSupported**() **const**
Whethor motion datas supported or not.

void **EnableMotionDatas**(std::size_t *max_size* = std::numeric_limits<std::size_t>::max())
Enable motion datas.

If max_size <= 0, indicates only can get datas from callback. If max_size > 0, indicates can get datas from callback or using *GetMotionDatas()*.

Note: if max_size > 0, the motion datas will be cached until you call *GetMotionDatas()*.

void **DisableMotionDatas**()
Disable motion datas.

bool **IsMotionDatasEnabled**() **const**
Whethor motion datas enabled or not.

std::vector<*MotionData*> **GetMotionDatas**()
Get cached motion datas.

Besides, you can also get them from callback

void **SetImgInfoCallback**(img_info_callback_t *callback*, bool *async* = true)
Set image info callback.

void **SetStreamCallback**(**const** *ImageType* &*type*, stream_callback_t *callback*, bool *async* = true)
Set stream data callback.

void **SetMotionCallback**(motion_callback_t *callback*, bool *async* = true)
Set motion data callback.

void **Close**()
Close the camera.

bool **HidFirmwareUpdate**(**const** char **filepath*)
Update hid device firmware.

void **SetExposureTime**(**const** float &*value*)
Set exposure time [1ms - 2000ms] value exposure time value.

void **GetExposureTime**(float &*value*)
Get exposure time value return exposure time value.

void **SetGlobalGain**(**const** float &*value*)
Set global gain [1 - 16] value global gain value.

void **GetGlobalGain**(float &*value*)
Get global gain value return global gain value.

void **SetIRIntensity**(**const** std::uint16_t &*value*)
set infrared(IR) intensity [0, 10] default 4

bool **AutoExposureControl**(bool *enable*)
Auto-exposure enabled or not default enabled.

bool **AutoWhiteBalanceControl**(bool *enable*)
Auto-white-balance enabled or not default enabled.

bool **IsLocationDatasSupported**() **const**
Whethor location datas supported or not.

void **EnableLocationDatas**(std::size_t *max_size* = std::numeric_limits<std::size_t>::max())
Enable location datas.

If max_size <= 0, indicates only can get datas from callback. If max_size > 0, indicates can get datas from callback or using *GetLocationDatas()*.

Note: if max_size > 0, the distance datas will be cached until you call *GetLocationDatas()*.

void **DisableLocationDatas**()
Disable location datas.

bool **IsLocationDatasEnabled**() **const**
Whethor location datas enabled or not.

std::vector<LocationData> **GetLocationDatas**()
Get cached location datas.

Besides, you can also get them from callback

void **SetLocationCallback**(location_callback_t *callback*, bool *async* = true)
Set location data callback.

bool **IsDistanceDatasSupported**() **const**
Whethor distance datas supported or not.

void **EnableDistanceDatas**(std::size_t *max_size* = std::numeric_limits<std::size_t>::max())
Enable distance datas.

If max_size <= 0, indicates only can get datas from callback. If max_size > 0, indicates can get datas from callback or using *GetDistanceDatas()*.

Note: if max_size > 0, the distance datas will be cached until you call *GetDistanceDatas()*.

void **DisableDistanceDatas**()
Disable distance datas.

bool **IsDistanceDatasEnabled**() **const**
Whethor distance datas enabled or not.

std::vector<DistanceData> **GetDistanceDatas**()
Get cached distance datas.

Besides, you can also get them from callback

void **SetDistanceCallback**(distance_callback_t *callback*, bool *async* = true)
Set distance data callback.

7.2 Device

7.2.1 DeviceInfo

struct DeviceInfo
Device information.

Public Members

std::int32_t **index**
The device index.

std::string **name**
The device name.

std::uint16_t **type**
The device type.

std::uint16_t **pid**
The product id.

std::uint16_t **vid**
The vendor id.

std::uint16_t **chip_id**
The chip id.

std::string **fw_version**
The firmware version.

7.2.2 Image

class Image
Subclassed by mynteyed::ImageColor, mynteyed::ImageDepth

7.2.3 OpenParams

struct OpenParams
Device open parameters.

Public Functions

OpenParams()
Constructor.

~OpenParams()
Destructor.

Public Members

std::int32_t **dev_index**
Device index.

std::int32_t **framerate**
Framerate, range [0,60], [0,30](STREAM_2560x720), default 10.

DeviceMode **dev_mode**
Device mode, default DEVICE_ALL.

- DEVICE_COLOR: IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH n
- DEVICE_DEPTH: IMAGE_LEFT_COLOR n IMAGE_RIGHT_COLOR n IMAGE_DEPTH y
- DEVICE_ALL: IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH y

Could detect image type is enabled after opened through *Camera::IsStreamDataEnabled()*.

Note: y: available, n: unavailable, -: depends on *stream_mode*

ColorMode **color_mode**
Color mode, default COLOR_RAW.

DepthMode **depth_mode**
Depth mode, default DEPTH_COLORFUL.

StreamMode **stream_mode**
Stream mode of color & depth, default STREAM_1280x720.

StreamFormat **color_stream_format**
Stream format of color, default STREAM_YUYV.

StreamFormat **depth_stream_format**
Stream format of depth, default STREAM_YUYV.

bool **state_ae**
Auto-exposure, default true.

bool **state_awb**
Auto-white balance, default true.

std::uint8_t **ir_intensity**
IR (Infrared), range [0,10], default 0.

bool **ir_depth_only**
IR Depth Only mode, default false.

Note: When frame rate less than 30fps, IR Depth Only will be not available.

float **colour_depth_value**
Colour depth image, default 5000.

[0, 16384]

7.2.4 StreamInfo

struct StreamInfo
Stream information.

Public Members

std::int32_t **index**
The stream index.

std::int32_t **width**
The stream width.

std::int32_t **height**
The stream height.

StreamFormat **format**
The stream format.

7.3 Enums

7.3.1 ErrorCode

enum mynteyed::**ErrorCode**
List error codes.

Values:

SUCCESS = 0
Standard code for successful behavior.

ERROR_FAILURE
Standard code for unsuccessful behavior.

ERROR_FILE_OPEN_FAILED
File cannot be opened for not exist, not a regular file or any other reason.

ERROR_CAMERA_OPEN_FAILED
Camera cannot be opened for not plugged or any other reason.

ERROR_CAMERA_NOT_OPENED
Camera is not opened now.

ERROR_CAMERA_RETRIEVE_FAILED
Camera retrieve the image failed.

ERROR_IMU_OPEN_FAILED
Imu cannot be opened for not plugged or any other reason.

ERROR_IMU_RECV_TIMEOUT
Imu receive data timeout.

ERROR_IMU_DATA_ERROR
Imu receive data error.

ERROR_CODE_LAST
Last guard.

7.3.2 Descriptor

enum mynteyed::**Descriptor**
The descriptor fields.

Values:

DEVICE_NAME
Device name.

SERIAL_NUMBER
Serial number.

FIRMWARE_VERSION
Firmware version.

HARDWARE_VERSION
Hardware version.

SPEC_VERSION
Spec version.

LENS_TYPE
Lens type.

IMU_TYPE
IMU type.

NOMINAL_BASELINE
Nominal baseline.

DESC_LAST
Last guard.

7.3.3 ProcessMode

enum mynteyed::**ProcessMode**
Process modes.

Values:

PROC_NONE = 0

PROC_IMU_ASSEMBLY = 1

PROC_IMU_TEMP_DRIFT = 2

PROC_IMU_ALL = *PROC_IMU_ASSEMBLY* | *PROC_IMU_TEMP_DRIFT*

7.3.4 DeviceMode

enum mynteyed::**DeviceMode**
List device modes.

Control the color & depth streams enabled or not.

Note: y: available, n: unavailable, -: depends on StreamMode

Values:

DEVICE_COLOR = 0
IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH n.

DEVICE_DEPTH = 1
IMAGE_LEFT_COLOR n IMAGE_RIGHT_COLOR n IMAGE_DEPTH y.

DEVICE_ALL = 2
IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH y.

7.3.5 ColorMode

enum mynteyed::**ColorMode**
List color modes.

Values:

COLOR_RAW = 0
color raw

COLOR_RECTIFIED = 1
color rectified

COLOR_MODE_LAST

7.3.6 DepthMode

enum mynteyed::**DepthMode**
List depth modes.

Values:

DEPTH_RAW = 0
ImageFormat::DEPTH_RAW.

DEPTH_GRAY = 1
ImageFormat::DEPTH_GRAY_24.

DEPTH_COLORFUL = 2
ImageFormat::DEPTH_RGB.

DEPTH_MODE_LAST

7.3.7 StreamMode

enum mynteyed::**StreamMode**
List stream modes.

Values:

STREAM_640x480 = 0
480p, vga, left

STREAM_1280x480 = 1
480p, vga, left+right

STREAM_1280x720 = 2
720p, hd, left

STREAM_2560x720 = 3
720p, hd, left+right

STREAM_MODE_LAST

7.3.8 StreamFormat

enum mynteyed::**StreamFormat**
List stream formats.

Values:

STREAM_MJPG = 0

STREAM_YUYV = 1

STREAM_FORMAT_LAST

7.3.9 ImageType

enum mynteyed::**ImageType**

List image types.

Values:

IMAGE_LEFT_COLOR
LEFT Color.

IMAGE_RIGHT_COLOR
RIGHT Color.

IMAGE_DEPTH
Depth.

IMAGE_ALL
All.

7.3.10 ImageFormat

enum mynteyed::**ImageFormat**

List image formats.

Values:

IMAGE_BGR_24
8UC3

IMAGE_RGB_24
8UC3

IMAGE_GRAY_8
8UC1

IMAGE_GRAY_16
16UC1

IMAGE_GRAY_24
8UC3

IMAGE_YUYV
8UC2

IMAGE_MJPG

COLOR_BGR = *IMAGE_BGR_24*

COLOR_RGB = *IMAGE_RGB_24*

COLOR_YUYV = *IMAGE_YUYV*

COLOR_MJPG = *IMAGE_MJPG*

DEPTH_RAW = *IMAGE_GRAY_16*

DEPTH_GRAY = *IMAGE_GRAY_8*

DEPTH_GRAY_24 = *IMAGE_GRAY_24*

DEPTH_BGR = *IMAGE_BGR_24*

DEPTH_RGB = *IMAGE_RGB_24*

IMAGE_FORMAT_LAST
Last guard.

7.3.11 SensorType

enum mynteyed::**SensorType**
SensorType types.

Values:

SENSOR_TYPE_H22 = 0

SENSOR_TYPE_OV7740

SENSOR_TYPE_AR0134

SENSOR_TYPE_AR0135

SENSOR_TYPE_OV9714

7.3.12 SensorMode

enum mynteyed::**SensorMode**
SensorMode modes.

Values:

LEFT = 0

RIGHT

ALL

7.4 Types

7.4.1 Data

ImgInfo

struct ImgInfo
Image information.

Public Members

std::uint16_t **frame_id**
Image frame id.

std::uint32_t **timestamp**
Image timestamp.

std::uint16_t **exposure_time**
Image exposure time.

ImuData

struct ImuData
Imu data.

Public Members

std::uint8_t **flag**
Data type MYNTEYE_IMU_ACCEL: accelerometer MYNTEYE_IMU_GYRO: gyroscope.

std::uint64_t **timestamp**
Imu gyroscope or accelerometer or frame timestamp.

double **temperature**
temperature

double **accel**[3]
Imu accelerometer data for 3-axis: X, Y, X.

double **gyro**[3]
Imu gyroscope data for 3-axis: X, Y, Z.

StreamData

struct StreamData
Stream data.

Public Members

std::shared_ptr<*Image*> **img**
Image data.

std::shared_ptr<*ImgInfo*> **img_info**
Image information.

MotionData

struct MotionData
Motion data.

Public Members

std::shared_ptr<*ImuData*> **imu**
ImuData.

7.4.2 Calib

CameraIntrinsics

struct CameraIntrinsics

Camera intrinsics: size, coeffs and camera matrix.

Public Members

std::uint16_t **width**

The width of the image in pixels.

std::uint16_t **height**

The height of the image in pixels.

double **fx**

The focal length of the image plane, as a multiple of pixel width.

double **fy**

The focal length of the image plane, as a multiple of pixel height.

double **cx**

The horizontal coordinate of the principal point of the image.

double **cy**

The vertical coordinate of the principal point of the image.

double **coeffs**[5]

The distortion coefficients: k1,k2,p1,p2,k3.

double **p**[12]

3x4 projection matrix in the (rectified) coordinate systems left: fx' cx' fy' cy' 1 right: fx' cx' tx fy' cy' 1

double **r**[9]

3x3 rectification transform (rotation matrix) for the left camera.

StreamIntrinsics

struct StreamIntrinsics

Camera intrinsics: size, coeffs and camera matrix.

ImuIntrinsics

struct ImuIntrinsics

IMU intrinsics: scale, drift and variances.

Public Members

double **scale**[3][3]

Scale matrix.

```
Scale X     cross axis  cross axis
cross axis  Scale Y     cross axis
cross axis  cross axis  Scale Z
```

double **assembly**[3][3]
Assembly error [3][3].

double **noise**[3]
Noise density variances.

double **bias**[3]
Random walk variances.

double **x**[2]
Temperature drift.

```
0 - Constant value
1 - Slope
```

MotionIntrinsics

struct MotionIntrinsics
Motion intrinsics, including accelerometer and gyroscope.

Public Members

ImuIntrinsics **accel**
Accelerometer intrinsics.

ImuIntrinsics **gyro**
Gyroscope intrinsics.

Extrinsics

struct Extrinsics
Extrinsics, represent how the different datas are connected.

Public Functions

Extrinsics **Inverse**() **const**
Inverse this extrinsics.

Return the inversed extrinsics.

Public Members

double **rotation**[3][3]
Rotation matrix left camera to right camera.

double **translation**[3]
Translation vector left camera to right camera.

7.5 Utils

7.5.1 select

bool mynteyed::util::**select**(**const** *Camera* &*cam*, *DeviceInfo* **info*)

7.5.2 print_stream_infos

void mynteyed::util::**print_stream_infos**(**const** *Camera* &*cam*, **const** std::int32_t &*dev_index*)

7.5.3 is_right_color_supported

bool mynteyed::util::**is_right_color_supported**(**const** *StreamMode* &*mode*)

Index

M

mynteyed::ALL (*C++ enumerator*), 83
mynteyed::Camera (*C++ class*), 73
mynteyed::Camera::AutoExposureControl (*C++ function*), 76
mynteyed::Camera::AutoWhiteBalanceControl (*C++ function*), 76
mynteyed::Camera::Close (*C++ function*), 75
mynteyed::Camera::DisableDistanceDatas (*C++ function*), 76
mynteyed::Camera::DisableImageInfo (*C++ function*), 74
mynteyed::Camera::DisableLocationDatas (*C++ function*), 76
mynteyed::Camera::DisableMotionDatas (*C++ function*), 75
mynteyed::Camera::EnableDistanceDatas (*C++ function*), 76
mynteyed::Camera::EnableImageInfo (*C++ function*), 74
mynteyed::Camera::EnableLocationDatas (*C++ function*), 76
mynteyed::Camera::EnableMotionDatas (*C++ function*), 75
mynteyed::Camera::EnableProcessMode (*C++ function*), 74
mynteyed::Camera::GetDescriptor (*C++ function*), 73
mynteyed::Camera::GetDescriptors (*C++ function*), 73
mynteyed::Camera::GetDeviceInfos (*C++ function*), 73
mynteyed::Camera::GetDistanceDatas (*C++ function*), 76
mynteyed::Camera::GetExposureTime (*C++ function*), 75
mynteyed::Camera::GetGlobalGain (*C++ function*), 75
mynteyed::Camera::GetLocationDatas (*C++ function*), 76
mynteyed::Camera::GetMotionDatas (*C++ function*), 75
mynteyed::Camera::GetMotionExtrinsics (*C++ function*), 74
mynteyed::Camera::GetMotionIntrinsics (*C++ function*), 74
mynteyed::Camera::GetStreamData (*C++ function*), 75
mynteyed::Camera::GetStreamDatas (*C++ function*), 75
mynteyed::Camera::GetStreamExtrinsics (*C++ function*), 74
mynteyed::Camera::GetStreamInfos (*C++ function*), 73
mynteyed::Camera::GetStreamIntrinsics (*C++ function*), 73, 74
mynteyed::Camera::HasStreamDataEnabled (*C++ function*), 75
mynteyed::Camera::HidFirmwareUpdate (*C++ function*), 75
mynteyed::Camera::IsDistanceDatasEnabled (*C++ function*), 76
mynteyed::Camera::IsDistanceDatasSupported (*C++ function*), 76
mynteyed::Camera::IsImageInfoEnabled (*C++ function*), 74
mynteyed::Camera::IsImageInfoSupported (*C++ function*), 74
mynteyed::Camera::IsImageInfoSynced (*C++ function*), 74
mynteyed::Camera::IsLocationDatasEnabled (*C++ function*), 76
mynteyed::Camera::IsLocationDatasSupported (*C++ function*), 76
mynteyed::Camera::IsMotionDatasEnabled (*C++ function*), 75
mynteyed::Camera::IsMotionDatasSupported (*C++ function*), 75
mynteyed::Camera::IsOpened (*C++ function*),

73
mynteyed::Camera::IsStreamDataEnabled (C++ function), 74
mynteyed::Camera::IsWriteDeviceSupported (C++ function), 74
mynteyed::Camera::Open (C++ function), 73
mynteyed::Camera::SetDistanceCallback (C++ function), 76
mynteyed::Camera::SetExposureTime (C++ function), 75
mynteyed::Camera::SetGlobalGain (C++ function), 75
mynteyed::Camera::SetImgInfoCallback (C++ function), 75
mynteyed::Camera::SetIRIntensity (C++ function), 76
mynteyed::Camera::SetLocationCallback (C++ function), 76
mynteyed::Camera::SetMotionCallback (C++ function), 75
mynteyed::Camera::SetStreamCallback (C++ function), 75
mynteyed::Camera::WriteCameraCalibrationBinFile (C++ function), 74
mynteyed::Camera::WriteDeviceFlash (C++ function), 74
mynteyed::CameraIntrinsics (C++ class), 85
mynteyed::CameraIntrinsics::coeffs (C++ member), 85
mynteyed::CameraIntrinsics::cx (C++ member), 85
mynteyed::CameraIntrinsics::cy (C++ member), 85
mynteyed::CameraIntrinsics::fx (C++ member), 85
mynteyed::CameraIntrinsics::fy (C++ member), 85
mynteyed::CameraIntrinsics::height (C++ member), 85
mynteyed::CameraIntrinsics::p (C++ member), 85
mynteyed::CameraIntrinsics::r (C++ member), 85
mynteyed::CameraIntrinsics::width (C++ member), 85
mynteyed::COLOR_BGR (C++ enumerator), 82
mynteyed::COLOR_MJPG (C++ enumerator), 82
mynteyed::COLOR_MODE_LAST (C++ enumerator), 81
mynteyed::COLOR_RAW (C++ enumerator), 81
mynteyed::COLOR_RECTIFIED (C++ enumerator), 81
mynteyed::COLOR_RGB (C++ enumerator), 82
mynteyed::COLOR_YUYV (C++ enumerator), 82
mynteyed::ColorMode (C++ enum), 81
mynteyed::DEPTH_BGR (C++ enumerator), 83
mynteyed::DEPTH_COLORFUL (C++ enumerator), 81
mynteyed::DEPTH_GRAY (C++ enumerator), 81, 82
mynteyed::DEPTH_GRAY_24 (C++ enumerator), 83
mynteyed::DEPTH_MODE_LAST (C++ enumerator), 81
mynteyed::DEPTH_RAW (C++ enumerator), 81, 82
mynteyed::DEPTH_RGB (C++ enumerator), 83
mynteyed::DepthMode (C++ enum), 81
mynteyed::DESC_LAST (C++ enumerator), 80
mynteyed::Descriptor (C++ enum), 79
mynteyed::DEVICE_ALL (C++ enumerator), 80
mynteyed::DEVICE_COLOR (C++ enumerator), 80
mynteyed::DEVICE_DEPTH (C++ enumerator), 80
mynteyed::DEVICE_NAME (C++ enumerator), 79
mynteyed::DeviceInfo (C++ class), 77
mynteyed::DeviceInfo::chip_id (C++ member), 77
mynteyed::DeviceInfo::fw_version (C++ member), 77
mynteyed::DeviceInfo::index (C++ member), 77
mynteyed::DeviceInfo::name (C++ member), 77
mynteyed::DeviceInfo::pid (C++ member), 77
mynteyed::DeviceInfo::type (C++ member), 77
mynteyed::DeviceInfo::vid (C++ member), 77
mynteyed::DeviceMode (C++ enum), 80
mynteyed::ERROR_CAMERA_NOT_OPENED (C++ enumerator), 79
mynteyed::ERROR_CAMERA_OPEN_FAILED (C++ enumerator), 79
mynteyed::ERROR_CAMERA_RETRIEVE_FAILED (C++ enumerator), 79
mynteyed::ERROR_CODE_LAST (C++ enumerator), 79
mynteyed::ERROR_FAILURE (C++ enumerator), 79
mynteyed::ERROR_FILE_OPEN_FAILED (C++ enumerator), 79
mynteyed::ERROR_IMU_DATA_ERROR (C++ enumerator), 79
mynteyed::ERROR_IMU_OPEN_FAILED (C++ enumerator), 79
mynteyed::ERROR_IMU_RECV_TIMEOUT (C++ enumerator), 79
mynteyed::ErrorCode (C++ enum), 79
mynteyed::Extrinsics (C++ class), 86
mynteyed::Extrinsics::Inverse (C++ function), 86

mynteyed::Extrinsics::rotation (C++ member), 86
mynteyed::Extrinsics::translation (C++ member), 86
mynteyed::FIRMWARE_VERSION (C++ enumerator), 80
mynteyed::HARDWARE_VERSION (C++ enumerator), 80
mynteyed::Image (C++ class), 77
mynteyed::IMAGE_ALL (C++ enumerator), 82
mynteyed::IMAGE_BGR_24 (C++ enumerator), 82
mynteyed::IMAGE_DEPTH (C++ enumerator), 82
mynteyed::IMAGE_FORMAT_LAST (C++ enumerator), 83
mynteyed::IMAGE_GRAY_16 (C++ enumerator), 82
mynteyed::IMAGE_GRAY_24 (C++ enumerator), 82
mynteyed::IMAGE_GRAY_8 (C++ enumerator), 82
mynteyed::IMAGE_LEFT_COLOR (C++ enumerator), 82
mynteyed::IMAGE_MJPG (C++ enumerator), 82
mynteyed::IMAGE_RGB_24 (C++ enumerator), 82
mynteyed::IMAGE_RIGHT_COLOR (C++ enumerator), 82
mynteyed::IMAGE_YUYV (C++ enumerator), 82
mynteyed::ImageFormat (C++ enum), 82
mynteyed::ImageType (C++ enum), 82
mynteyed::ImgInfo (C++ class), 83
mynteyed::ImgInfo::exposure_time (C++ member), 83
mynteyed::ImgInfo::frame_id (C++ member), 83
mynteyed::ImgInfo::timestamp (C++ member), 83
mynteyed::IMU_TYPE (C++ enumerator), 80
mynteyed::ImuData (C++ class), 84
mynteyed::ImuData::accel (C++ member), 84
mynteyed::ImuData::flag (C++ member), 84
mynteyed::ImuData::gyro (C++ member), 84
mynteyed::ImuData::temperature (C++ member), 84
mynteyed::ImuData::timestamp (C++ member), 84
mynteyed::ImuIntrinsics (C++ class), 85
mynteyed::ImuIntrinsics::assembly (C++ member), 85
mynteyed::ImuIntrinsics::bias (C++ member), 86
mynteyed::ImuIntrinsics::noise (C++ member), 86
mynteyed::ImuIntrinsics::scale (C++ member), 85
mynteyed::ImuIntrinsics::x (C++ member), 86
mynteyed::LEFT (C++ enumerator), 83
mynteyed::LENS_TYPE (C++ enumerator), 80
mynteyed::MotionData (C++ class), 84
mynteyed::MotionData::imu (C++ member), 84
mynteyed::MotionIntrinsics (C++ class), 86
mynteyed::MotionIntrinsics::accel (C++ member), 86
mynteyed::MotionIntrinsics::gyro (C++ member), 86
mynteyed::NOMINAL_BASELINE (C++ enumerator), 80
mynteyed::OpenParams (C++ class), 77
mynteyed::OpenParams::~OpenParams (C++ function), 77
mynteyed::OpenParams::color_mode (C++ member), 78
mynteyed::OpenParams::color_stream_format (C++ member), 78
mynteyed::OpenParams::colour_depth_value (C++ member), 78
mynteyed::OpenParams::depth_mode (C++ member), 78
mynteyed::OpenParams::depth_stream_format (C++ member), 78
mynteyed::OpenParams::dev_index (C++ member), 78
mynteyed::OpenParams::dev_mode (C++ member), 78
mynteyed::OpenParams::framerate (C++ member), 78
mynteyed::OpenParams::ir_depth_only (C++ member), 78
mynteyed::OpenParams::ir_intensity (C++ member), 78
mynteyed::OpenParams::OpenParams (C++ function), 77
mynteyed::OpenParams::state_ae (C++ member), 78
mynteyed::OpenParams::state_awb (C++ member), 78
mynteyed::OpenParams::stream_mode (C++ member), 78
mynteyed::PROC_IMU_ALL (C++ enumerator), 80
mynteyed::PROC_IMU_ASSEMBLY (C++ enumerator), 80
mynteyed::PROC_IMU_TEMP_DRIFT (C++ enumerator), 80
mynteyed::PROC_NONE (C++ enumerator), 80
mynteyed::ProcessMode (C++ enum), 80
mynteyed::RIGHT (C++ enumerator), 83
mynteyed::SENSOR_TYPE_AR0134 (C++ enumerator), 83
mynteyed::SENSOR_TYPE_AR0135 (C++ enumer-

ator), 83
mynteyed::SENSOR_TYPE_H22 (*C++ enumerator*), 83
mynteyed::SENSOR_TYPE_OV7740 (*C++ enumerator*), 83
mynteyed::SENSOR_TYPE_OV9714 (*C++ enumerator*), 83
mynteyed::SensorMode (*C++ enum*), 83
mynteyed::SensorType (*C++ enum*), 83
mynteyed::SERIAL_NUMBER (*C++ enumerator*), 80
mynteyed::SPEC_VERSION (*C++ enumerator*), 80
mynteyed::STREAM_1280x480 (*C++ enumerator*), 81
mynteyed::STREAM_1280x720 (*C++ enumerator*), 81
mynteyed::STREAM_2560x720 (*C++ enumerator*), 81
mynteyed::STREAM_640x480 (*C++ enumerator*), 81
mynteyed::STREAM_FORMAT_LAST (*C++ enumerator*), 82
mynteyed::STREAM_MJPG (*C++ enumerator*), 82
mynteyed::STREAM_MODE_LAST (*C++ enumerator*), 81
mynteyed::STREAM_YUYV (*C++ enumerator*), 82
mynteyed::StreamData (*C++ class*), 84
mynteyed::StreamData::img (*C++ member*), 84
mynteyed::StreamData::img_info (*C++ member*), 84
mynteyed::StreamFormat (*C++ enum*), 81
mynteyed::StreamInfo (*C++ class*), 78
mynteyed::StreamInfo::format (*C++ member*), 79
mynteyed::StreamInfo::height (*C++ member*), 79
mynteyed::StreamInfo::index (*C++ member*), 79
mynteyed::StreamInfo::width (*C++ member*), 79
mynteyed::StreamIntrinsics (*C++ class*), 85
mynteyed::StreamMode (*C++ enum*), 81
mynteyed::SUCCESS (*C++ enumerator*), 79
mynteyed::util::is_right_color_supported (*C++ function*), 87
mynteyed::util::print_stream_infos (*C++ function*), 87
mynteyed::util::select (*C++ function*), 87