MYNT® EYE S SDK Documentation

Release 2.5.0

MYNTAI

Jan 06, 2023

CONTENTS

CHAPTER

ONE

PRODUCT

1.1 Product Description

MYNT® EYE S Series includes MYNT EYE S, MYNT EYE SE and MYNT EYE SC. The "Binocular + IMU" inertial navigation solution for MYNT® EYE S Series provides accurate six-axis complementary data for vSLAM applications and is more accurate and robust than other single solutions.

Combined with self-developed camera synchronization, auto exposure, and white balance control camera technology, MYNT® EYE S Series provides a CUDA-based GPU real-time acceleration solution that outputs high-precision, synchronized image sources to help reduce the difficulty of algorithm development and speed up the efficiency of algorithm research and development. At the same time, MYNT EYE S is equipped with a six-axis sensor (IMU) and an infrared active light (IR). Among them, six-axis sensor (IMU) can provide data complementation and correction for the research of visual positioning algorithm, suitable for algorithm research of visual inertial odometer (VIO), help improve positioning accuracy; infrared active light (IR) can help solve the problem of identifying indoor white walls and non-textured objects, and improve the recognition accuracy of image sources. The difference between MYNT EYE SE and MYNT EYE S is that MYNT EYE SE does not include IR and offers customers with lower cost hardware. MYNT EYE SC provides 8cm/12cm optional baseline solution, super wide angle 146°FOV, providing a wider depth recognition range and accuracy level, with color image sensor, upgraded brand new BMI088 six-axis IMU, IR active light, I2C time synchronization Chip, global shutter, etc., with resolutions up to 2560x800@30fps and accuracy is up to centimeters. In addition, MYNT EYE S Series also provides a rich SDK interface and VSLAM open source project support, which can help customers quickly integrate solutions, accelerate the product development process, and achieve rapid productization and implementation.

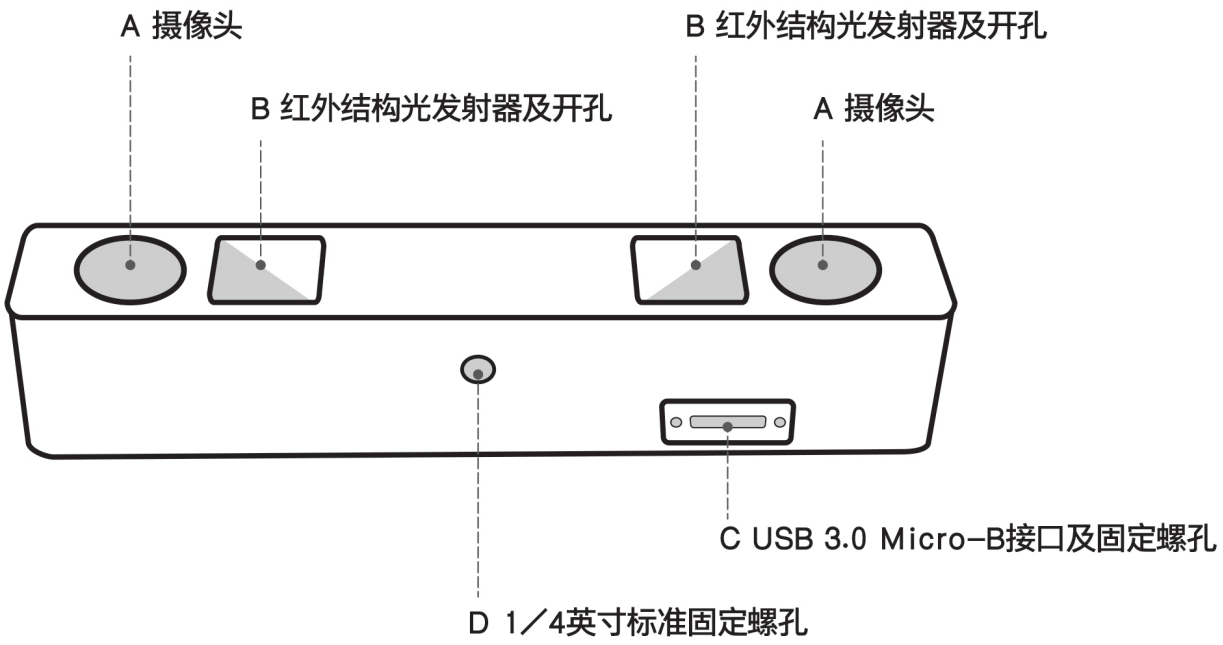
As a hardware product for research and development of stereo vision computing applications, MYNT® EYE S Series can be widely used in a field of visual positioning navigation (vSLAM), including visual real-time positioning navigation system of driverless vehicle and robots, visual positioning system of UAV, obstacle avoidance navigation system for driverless Vehicle, Augmented Reality (AR), Virtual Reality (VR), etc. At the same time, it can be used in a 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. At present, MYNTAI has carried out service and cooperation with more than 500 domestic and foreign enterprise clients.

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

1.2.1 S1030 Size and Structure

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.

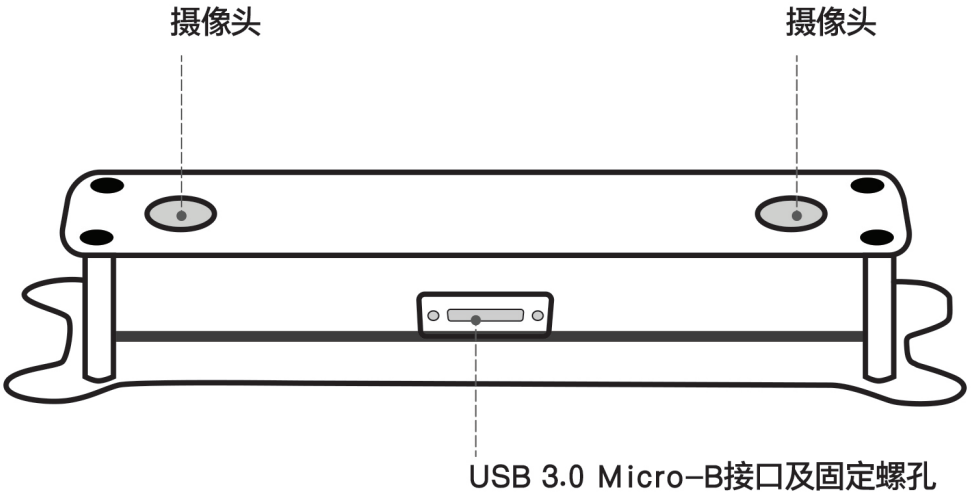
B. Infrared structured-light transmitter and outlet: the infrared structured-light can effectively solve the problem associated with the visual positioning calculations of white wall non-textured object(For non-IR version, the outlet is reserved but there is no internal structured-light emitter).

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.

1.2.2 S2100 Size and Structure

Shell(mm)	PCBA board(mm)
125x47x40	100x15

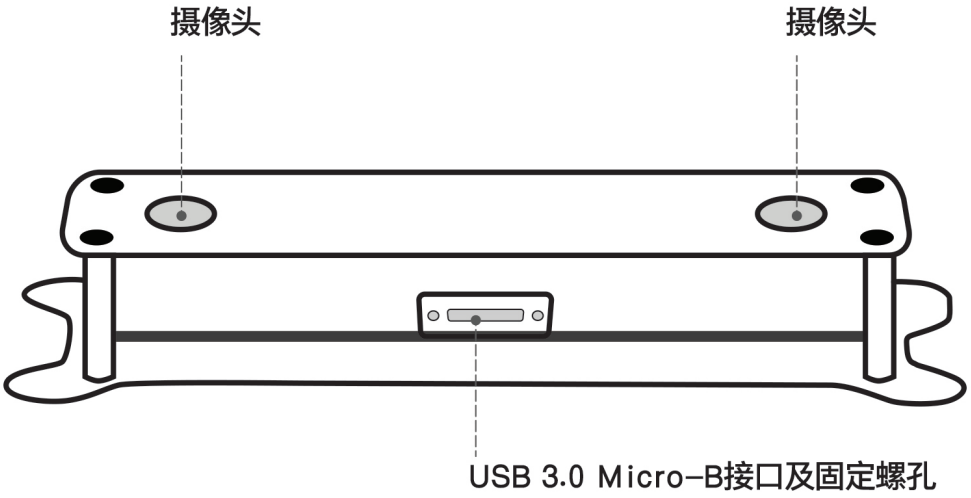


A. Camera:please pay attention to protect the camera sensor lenses, to avoid imaging quality degradation.

B. 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.

1.2.3 S2110 Size and Structure

Shell(mm)	PCBA board(mm)
125x47x40	100x15



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

B. 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.

1.3 Product Specification

1.3.1 S1030-120/Mono Product Specification

Product Specification

Model	S1030-120/Mono
Size	165x31.5x31.23mm
Frame Rate	10/15/20/25/30/35/40/45/50/55/60FPS
Resolution	752*480; 376*240
Depth Resolution	Based on CPU/GPU Up to 752*480@60FPS
Pixel Size	6.0*6.0m
Baseline	120.0mm
Visual Angle	D:146° H:122° V:76°
Focal Length	2.1mm
Filter	Dual Pass Filter
IR Support	No
IR detectable range	-
Color Mode	Monochrome
Depth Working Distance	0.8-5m+
Scanning Mode	Global Shutter
Power	1W@5V DC from USB
Synchronization Precision	<1ms (up to 0.05ms)
IMU Frequency	100/200/250/333/500Hz
Output data format	Raw data
Data transfer Interface	USB3.0
Weight	160g
UVC MODE	Yes

Software

Work Environment

Operating Temperature	10°C~50°C
Storage Temperature	-20°C~60°C
Humidity	10% to 90% non-condensing

Package

Package Contents	MYNT EYE x1 USB Micro-B Cable x1

Warranty

Product Warranty	12 Months Limited Manufacturer's Warranty

Accuracy

Depth Distance Deviation	Less than 4%

1.3.2 S1030-IR-120/Mono Product Specification

Product Specification

Model	S1030-IR-120/Mono
Size	165x31.5x31.23mm
Frame Rate	10/15/20/25/30/35/40/45/50/55/60FPS
Resolution	752*480; 376*240
Depth Resolution	Based on CPU/GPU Up to 752*480@60FPS
Pixel Size	6.0*6.0m
Baseline	120.0mm
Camera Lens	Replacable Standard M12
Visual Angle	D:146° H:122° V:76°
Focal Length	2.1mm
Filter	Dual Pass Filter
IR Support	Yes
IR detectable range	Up to 3m
Color Mode	Monochrome
Depth Working Distance	0.8-5m+
Scanning Mode	Global Shutter
Power	1~2.7W@5V DC from USB
Synchronization Precision	<1ms (up to 0.05ms)
IMU Frequency	100/200/250/333/500Hz
Output data format	Raw data
Data transfer Interface	USB3.0
Weight	184g
UVC MODE	Yes

Software

Work Environment

Operating Temperature	10°C~50°C
Storage Temperature	-20°C~60°C
Humidity	10% to 90% non-condensing

Package

Package Contents	MYNT EYE x1 USB Micro-B Cable x1

Warranty

Product Warranty	12 Months Limited Manufacturer's Warranty

Accuracy

Depth Distance Deviation	Less than 4%

1.3.3 S2100-146/Color Product Specification

Product Specification

Model	S2100-146/Color
Size	PCB dimension:15x100mm Total dimension:125x47x26.6mm
Frame Rate	1280x400@10/20/30/60fps 2560x800@10/20/30fps
Resolution	1280x400; 2560x800
Depth Resolution	Based on CPU/GPU Up to 1280*400@60FPS
Pixel Size	3.0*3.0m
Baseline	80.0mm
Visual Angle	D:141° H:124° V:87°
Focal Length	0.95mm
IR Support	No
Color Mode	Color
Depth Working Distance	0.26-3m+
Scanning Mode	Global Shutter
Power	1.1W@5V DC from USB
Synchronization Precision	<1ms (up to 0.02ms)
IMU Frequency	200Hz
Output data format	YUYV
Data transfer Interface	USB3.0
Time Sync interface	DF50A
Weight	62g
UVC MODE	Yes

Software

Support system	Windows 10Ubuntu 14.04/16.04/18.04ROS indigo/kinetic/melodicAndroid 7.0+
SDK	http://www.myntai.com/dev/mynteye
Support	ORB_SLAM2OKVISVins-MonoVins-FusionVIORB

Work Environment

Operating Temperature	-15°C~55°C
Storage Temperature	-20°C~75°C
Humidity	0% to 95% non-condensing

Package

Package Contents	MYNT EYE x1 USB Micro-B Cable x1

Warranty

Product Warranty	12 Months Limited Manufacturer's Warranty

Accuracy

Depth Distance Deviation	Less than 4%

1.3.4 S2110-95/Color Product Specification

Product Specification

Model	S2110-95/Color
Size	PCB dimension:17.74x100mm Total dimension:125x47x26.6mm
Frame Rate	1280x400@10/20/30/60fps 2560x800@10/20/30fps
Resolution	1280x400; 2560x800
Depth Resolution	Based on CPU/GPU Up to 1280*400@60FPS
Pixel Size	3.0*3.0m
Baseline	80.0mm
Visual Angle	D:112° H:95° V:50°
Focal Length	2.63mm
IR Support	No
Color Mode	Color
Depth Working Distance	0.60-7m+
Scanning Mode	Global Shutter
Power	1.1W@5V DC from USB
Synchronization Precision	<1ms (up to 0.02ms)
IMU Frequency	200Hz
Output data format	YUYV
Data transfer Interface	USB3.0
Time Sync interface	DF50A
Weight	100.8g
UVC MODE	Yes

Software

Support system	Windows 10Ubuntu 14.04/16.04/18.04ROS indigo/kinetic/melodicAndroid 7.0+
SDK	http://www.myntai.com/dev/mynteye
Support	ORB_SLAM2OKVISVins-MonoVins-FusionVIORB

Work Environment

Operating Temperature	-15°C~55°C
Storage Temperature	-20°C~75°C
Humidity	0% to 95% non-condensing

Package

Package Contents	MYNT EYE x1 USB Micro-B Cable x1

Warranty

Product Warranty	12 Months Limited Manufacturer's Warranty

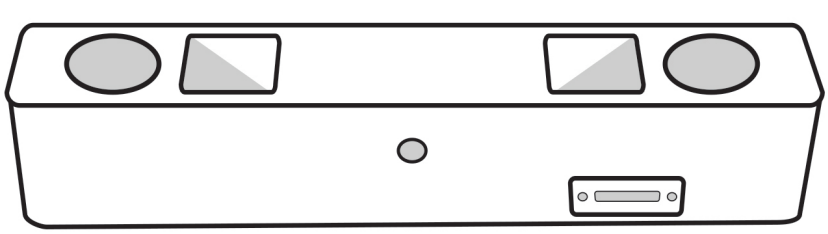
Accuracy

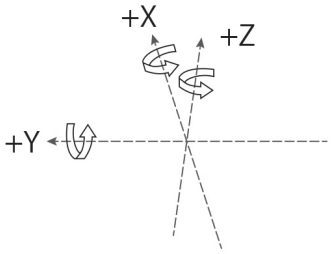
Depth Distance Deviation	Less than 4%

1.4 IMU Coordinate System

1.4.1 S1030 Coordinate System

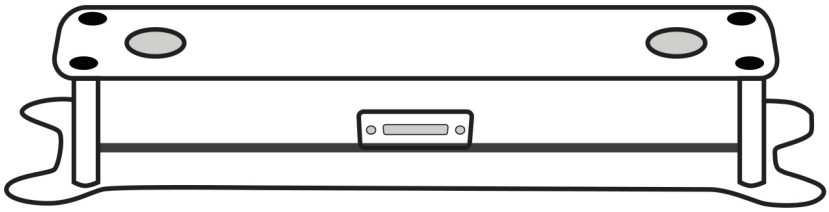
IMU coordinate system is right-handed,the axis directions are as follows:

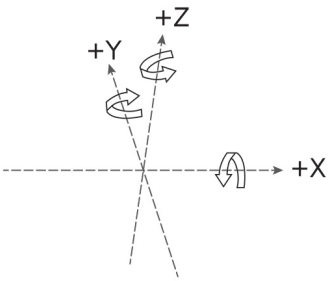




1.4.2 S2100 Coordinate System

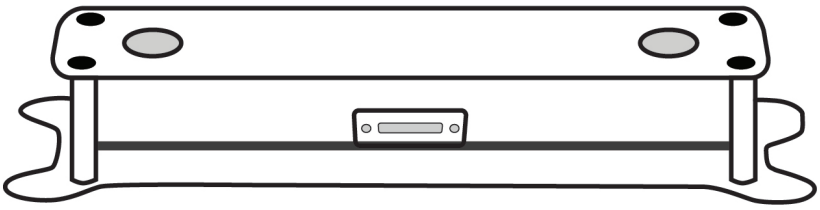
IMU coordinate system is right-handed,the axis directions are as follows:

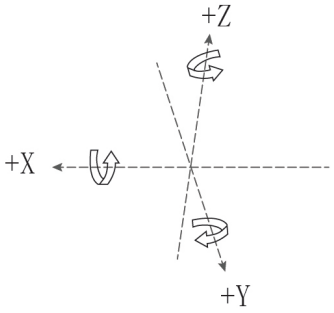




1.4.3 S2110 Coordinate System

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





CHAPTER

TWO

SDK

2.1 SDK Description

2.1.1 Supported Platforms

SDK is built on CMake and can be used cross multiple platforms such as “Linux, Windows, etc. We provides two installation: Download and install, and Compile and install from source code.

These are the platforms that can be used:

- Windows 10
- Ubuntu 18.04.1 / 16.04.6 / 14.04.5
- Jetson TX1/TX2 / Xavier
- firefly RK3399

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.

2.1.2 OpenCV Description

SDK provides a three-tier interface with OpenCV dependencies:

- `api`, upper interface, with OpenCV dependencies
- `device`, interlayer interface, without OpenCV dependencies
- `uvc`, bottom interface, without OpenCV dependencies

If you don’t want to use OpenCV, edit `<sdk>/cmake/Option.cmake` , set `WITH_API` to `OFF`. This will stop the compilation of the interface api:

```
option(WITH_API "Build with API layer, need OpenCV" ON)
```

For samples for the interface `device`, please refer to device/camera.cc .

2.2 SDK Installation

2.2.1 Ubuntu PPA Installation

Ubuntu 14.04	Ubuntu 16.04	Ubuntu 18.04
✓	✓	✓

x64 PPA installation

```
$ sudo add-apt-repository ppa:slightech/mynt-eye-s-sdk
$ sudo apt-get update
$ sudo apt-get install mynt-eye-s-sdk
```

armv8 PPA installation

```
$ sudo add-apt-repository ppa:slightech/mynt-eye-s-sdk-arm
$ sudo apt-get update
$ sudo apt-get install mynt-eye-s-sdk
```

Run samples

Tip: samples path: /opt/mynt-eye-s-sdk/samples

```
$ cd /opt/mynt-eye-s-sdk/samples
$ ./camera_with_junior_device_api
```

2.2.2 Ubuntu Source Installation

Ubuntu 14.04	Ubuntu 16.04	Ubuntu 18.04
✓	✓	✓

Tip: If you used any other Linux distributions without apt-get package manager, you need to install required packages manaully instead of using `make init`.

Linux	How to install required packages
Debian based	sudo apt-get install build-essential cmake git libv4l-dev
Red Hat based	sudo yum install make gcc gcc-c++ kernel-devel cmake git libv4l-devel
Arch Linux	sudo pacman -S base-devel cmake git v4l-utils

Getting Source Code

```
sudo apt-get install git
git clone https://github.com/slightech/MYNT-EYE-S-SDK.git
```

Required Packages

```
cd <sdk>
make init
```

- OpenCV

Tip: To build and install Opencv(Not support 4.0+), 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/3.4.1

$ mkdir _build
$ cd _build/

$ cmake \
-DCMAKE_BUILD_TYPE=RELEASE \
-DCMAKE_INSTALL_PREFIX=/usr/local \
\
-DWITH_CUDA=OFF \
\
-DBUILD_DOCS=OFF \
-DBUILD_EXAMPLES=OFF \
-DBUILD_TESTS=OFF \
-DBUILD_PERF_TESTS=OFF \
..

$ make -j4
$ sudo make install
```

Building code

Tip: If opencv is installed in custom directory or if you want to specify a version, you should set the path before building:

```
# OpenCV_DIR is the directory where your OpenCVConfig.cmake exists
export OpenCV_DIR=~/opencv
```

Otherwise, CMake will prompt cannot find OpenCV. If you need sdk without OpenCV, please read *OpenCV Description* .

Build and install:

```
cd <sdk>
make install
```

Finally, sdk will install in `/usr/local` by default.

Building samples

```
cd <sdk>
make samples
```

Run samples:

```
./samples/_output/bin/api/camera_a
```

Tutorial samples, please read *SDK Project Demos* and *SDK Control Samples* .

Building tools

```
cd <sdk>
make tools
```

Installation requirement:

```
cd <sdk>/tools/
sudo pip install -r requirements.txt
```

The usage of tools and scripts will be introduced later.

Conclusion

If your project will use SDK, you can refer to the settings in `samples/CMakeLists.txt` for CMake. Alternatively, import the head file and dynamic library in the installation directory.

2.2.3 Windows EXE Installation

Windows 10
✓

Download and install SDK

Tip: Download here: mynteye-s-x.x.x-win-x64-opencv-3.4.3.exe Google Drive Baidu Pan(key:rj4k) .

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

Goto the `<SDK_ROOT_DIR>\\bin\\samples\\tutorials` directory and click `get_stereo.exe` to run.

Generate samples project

First, you should install Visual Studio 2017 and CMake .

Second, goto the `<SDK_ROOT_DIR>\\samples` directory and click `generate.bat` to generate project.

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

The tutorials of samples are here: https://slightech.github.io/MYNT-EYE-S-SDK-Guide/src/data/contents.html.

Start using SDK with Visual Studio 2017

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

2.2.4 Windows Source Installation

Windows 10
✓

Tip: Windows does not provide Visual Studio `*.sln` file directly and requires CMake to build. Firstly, CMake can be used across multiple platforms, it is easy to configure and can be maintained in a sustainable way. Secondly, the third-party codes (Glog, OpenCV) are built using CMake.

Tip: There is currently no binary installer available, which requires you to compile from source. It is also the process of configuring the development environment.

Prerequisites

CMake (provide build

- CMakeused to build and compile (necessary).
- Git used to get code (optional).
- Doxygen used to generate documents (optional).

After you install the above tools, confirm that you can run this command in CMD (Command Prompt):

```
>cmake --version
cmake version 3.10.1

>git --version
git version 2.11.1.windows.1

>doxygen --version
1.8.13
```

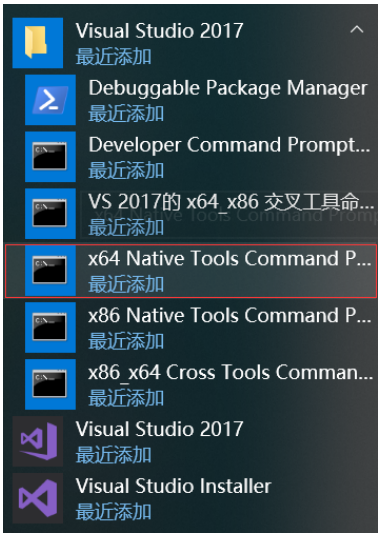
Visual Studio (provide compilation)

- Visual Studio
  - Visual Studio 2017
  - Visual Studio 2015
- Windows 10 SDK

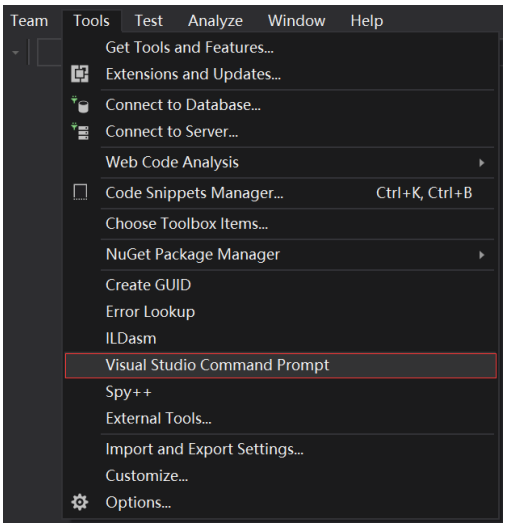
After installing Visual Studio, confirm that the following command can run in the Visual Studio Command Prompt:

```
>cl
Microsoft (R) C/C++ Optimizing Compiler Version 19.14.26429.4 for x86

>msbuild
Microsoft (R)  15.7.179.6572
```

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



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

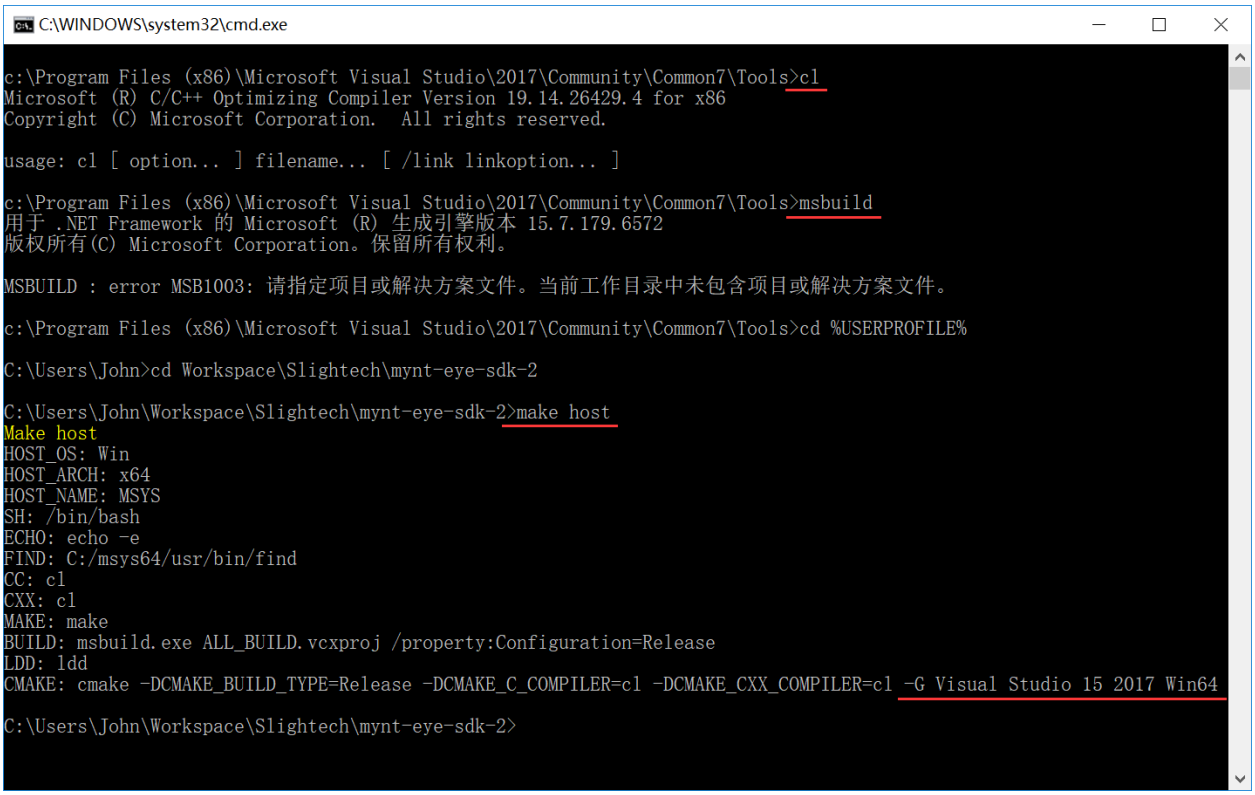


However, if you do not have the Visual Studio 2015 Tools menu, you can add one yourself.

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)`

In Visual Studio command Prompt, you can use the compile command `cl link lib msbuild`, etc.(need finish `MSYS2``and ``Getting Source Code` steps first)



MSYS2 (provide Linux command)

- MSYS2
  - mirror
  - pacman

After installation, verify that the following path has been added to the system environment variable PATH:

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

Then, open MSYS2 MSYS, perform the update and install `make`:

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

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

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

Getting Source Code

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

Required Packages

```
>cd <sdk>
>make init
Make init
Init deps
Install cmd: pacman -S
Install deps: git clang-format
pacman -S clang-format (not exists)
error: target not found: clang-format
pip install --upgrade autopep8 cpplint pylint requests
...
Init git hooks
ERROR: clang-format-diff is not installed!
Expect cmake version >= 3.0
cmake version 3.10.1
```

- OpenCV

Tip: The official OpenCV provides the `exe` for installation. If you want to compile from the source code, see the Official document Installation in Windows . or refer to the following command:

```
>git clone https://github.com/opencv/opencv.git
>cd opencv
>git checkout tags/3.4.1

>cd opencv
>mkdir _build
>cd _build

>cmake ^
-D CMAKE_BUILD_TYPE=RELEASE ^
-D CMAKE_INSTALL_PREFIX=C:/opencv ^
-D WITH_CUDA=OFF ^
-D BUILD_DOCS=OFF ^
-D BUILD_EXAMPLES=OFF ^
-D BUILD_TESTS=OFF ^
-D BUILD_PERF_TESTS=OFF ^
-G "Visual Studio 15 2017 Win64" ^
..

>msbuild ALL_BUILD.vcxproj /property:Configuration=Release
>msbuild INSTALL.vcxproj /property:Configuration=Release
```

Building Code

Tip: If OpenCV is installed in a custom directory or wants to specify a version, you can set the path as follows before compiling:

```
# OpenCV_DIR is hte path where OpenCVConfig.cmake in
set OpenCV_DIR=C:\opencv
```

Otherwise, CMake will prompt that OpenCV could not be found. If you don't want to rely on OpenCV, read *OpenCV Description* .

Build and install:

```
cd <sdk>
make install
```

Finally, the SDK will install in `<sdk>/_install` by default.

Building samples

```
cd <sdk>
make samples
```

Run samples:

```
.\samples\_output\bin\api\camera_a.bat
```

For tutorial samples, please read *SDK Project Demos* and *SDK Control Samples* .

Tip: All compiled sample programs `exe` will have a corresponding `bat`. `bat` will temporarily set system environment variables and then run `exe`. So it is recommended to run `bat`.

If you run``exe`` directly, it may prompt that cannot find `dll`. Then you should add `<sdk>\\_install\\bin` `%OPENCV_DIR%\\bin` to `PATH` in system environment variable.

How to set the environment variable for OpenCV, refer to the official document Set the OpenCV environment variable and add it to the systems path .

Building tools

```
cd <sdk>
make tools
```

The usage of tools and scripts will be introduced later.

Tip: The script is based on Python. You need to install Python and its package management tool pip first, and then install the dependencies as follows:

```
cd <sdk>\tools
pip install -r requirements.txt
```

Note: Python is also in MSYS2, but fail install Matplotlib in test.

Conclusion

If your project will use SDK, you can refer to the settings in `samples/CMakeLists.txt` for CMake. Or just import the head file and dynamic library in the installation directory.

2.2.5 ROS Wrapper Installation

ROS Kinetic	ROS Indigo
✓	✓

Prepare Environment

- ROS

ROS Melodic (Ubuntu 18.04)

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/
↪apt/sources.list.d/ros-latest.list'
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key␣
↪421C365BD9FF1F717815A3895523BAEEB01FA116
sudo apt update
sudo apt install ros-melodic-desktop-full
sudo rosdep init
rosdep update
```

ROS Kinetic (Ubuntu 16.04)

```
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 Indigo (Ubuntu 14.04)

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

Compiling Code

```
cd <sdk>
make ros
```

Running node

```
source wrappers/ros/devel/setup.bash
roslaunch mynt_eye_ros_wrapper mynteye.launch  # this node doesn't have preview
```

Run the node, and preview by RViz:

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

Testing Services

Run the node as follows, provide device information getting service, see follows:

```
$ source wrappers/ros/devel/setup.bash
$ rosrun mynt_eye_ros_wrapper get_device_info.py
LENS_TYPE: 0000
SPEC_VERSION: 1.0
NOMINAL_BASELINE: 120
HARDWARE_VERSION: 2.0
IMU_TYPE: 0000
SERIAL_NUMBER: 0610243700090720
FIRMWARE_VERSION: 2.0
DEVICE_NAME: MYNT-EYE-S1000
```

Common issues - ROS Indigo

Cannot find `libopencv` while `make ros`

```
make[3]: *** No rule to make target `/usr/lib/x86_64-linux-gnu/libopencv_videostab.so.2.
↪4.8', needed by `/home/john/Workspace/MYNT-EYE-S-SDK/wrappers/ros/devel/lib/libmynteye_
↪wrapper.so'.  Stop.
```

Solution 1) Install OpenCV 2:

```
sudo apt-get update
sudo apt-get install libcv-dev
```

Solution 2) Install OpenCV 3 & re-compiled `cv_bridge`:

```
sudo apt-get install ros-indigo-opencv3

git clone https://github.com/ros-perception/vision_opencv.git
mv vision_opencv/cv_bridge/ MYNT-EYE-S-SDK/wrappers/ros/src/
```

Then run `make ros` again

Conclusion

About more details, check the wrapper_ros .

2.3 SDK Data Samples

2.3.1 Get Device Information

Use `GetInfo()` function to get various current information values.

Reference code snippet:

```
auto &&api = API::Create(argc, argv);

LOG(INFO) << "Device name: " << api->GetInfo(Info::DEVICE_NAME);
LOG(INFO) << "Serial number: " << api->GetInfo(Info::SERIAL_NUMBER);
LOG(INFO) << "Firmware version: " << api->GetInfo(Info::FIRMWARE_VERSION);
LOG(INFO) << "Hardware version: " << api->GetInfo(Info::HARDWARE_VERSION);
LOG(INFO) << "Spec version: " << api->GetInfo(Info::SPEC_VERSION);
LOG(INFO) << "Lens type: " << api->GetInfo(Info::LENS_TYPE);
LOG(INFO) << "IMU type: " << api->GetInfo(Info::IMU_TYPE);
LOG(INFO) << "Nominal baseline: " << api->GetInfo(Info::NOMINAL_BASELINE);
```

Reference result on Linux:

```
$ ./samples/_output/bin/get_device_info
I0503 16:40:21.109391 32106 utils.cc:13] Detecting MYNT EYE devices
I0503 16:40:21.604116 32106 utils.cc:20] MYNT EYE devices:
I0503 16:40:21.604127 32106 utils.cc:24]   index: 0, name: MYNT-EYE-S1000
I0503 16:40:21.604142 32106 utils.cc:30] Only one MYNT EYE device, select index: 0
I0503 16:40:21.615054 32106 get_device_info.cc:10] Device name: MYNT-EYE-S1000
I0503 16:40:21.615113 32106 get_device_info.cc:11] Serial number: 0610243700090720
I0503 16:40:21.615129 32106 get_device_info.cc:12] Firmware version: 2.0
I0503 16:40:21.615139 32106 get_device_info.cc:13] Hardware version: 2.0
I0503 16:40:21.615146 32106 get_device_info.cc:14] Spec version: 1.0
I0503 16:40:21.615155 32106 get_device_info.cc:15] Lens type: 0000
I0503 16:40:21.615164 32106 get_device_info.cc:16] IMU type: 0000
I0503 16:40:21.615171 32106 get_device_info.cc:17] Nominal baseline: 120
```

Complete code examples, see get_device_info.cc .

2.3.2 Get Image Calibration Parameters

Use `GetIntrinsics()` & `GetExtrinsics()` to get image calibration parameters.

Tip: The detailed meaning of parameters can reference the files in `tools/writer/config` , of these the image calibration parameters of S21XX are in `tools/writer/config/S21XX` the image calibration parameters of S1030 are in `tools/writer/config/S1030`

Note

Camera Intrinsics/Extrinsics, please ref to: ros CameraInfo.

Reference code snippet:

```
auto &&api = API::Create(argc, argv);

LOG(INFO) << "Intrinsics left: {" << *api->GetIntrinsicsBase(Stream::LEFT)
          << "}";
LOG(INFO) << "Intrinsics right: {" << *api->GetIntrinsicsBase(Stream::RIGHT)
          << "}";
LOG(INFO) << "Extrinsics right to left: {"
          << api->GetExtrinsics(Stream::RIGHT, Stream::LEFT) << "}";
```

Reference result on Linux:

```
$ ./samples/_output/bin/get_img_params
I/utils.cc:48 MYNT EYE devices:
I/utils.cc:51   index: 0, name: MYNT-EYE-S1030, sn: 4B4C192400090712, firmware: 2.4
I/utils.cc:60 Only one MYNT EYE device, select index: 0
I/synthetic.cc:59 camera calib model: kannala_brandt
I/utils.cc:93 MYNT EYE requests:
I/utils.cc:96   index: 0, request: width: 752, height: 480, format: Format::YUYV, fps: 60
I/utils.cc:96   index: 1, request: width: 376, height: 240, format: Format::YUYV, fps: 60
I/utils.cc:107 There are 2 stream requests, select index:
0
I/get_img_params.cc:44 Intrinsics left: {equidistant, width: 752, height: 480, k2: 0.
↪00986113697985857, k3: -0.11937208025856659, k4: 0.19092250072175385, k5: -0.
↪10168315832257743, mu: 356.41566867259672335, mv: 356.31078130432149464, u0: 375.
↪76739787805968263, v0: 246.20025492033516912}
I/get_img_params.cc:45 Intrinsics right: {equidistant, width: 752, height: 480, k2: -0.
↪0224631217599986, k3: 0.01303393297719630, k4: -0.01735983686524734, k5: 0.
↪00675132874903371, mu: 357.96820061652590539, mv: 357.76889287108474491, u0: 397.
↪09281703352422710, v0: 258.93978588846073308}
I/get_img_params.cc:46 Extrinsics right to left: {rotation: [0.99997489222742053, 0.
↪00041828202737396, -0.00707389248605010, -0.00042920419615213, 0.99999871813992847, -0.
↪00154256353448567, 0.00707323819170721, 0.00154556094848940, 0.99997378992793495],␣
↪translation: [-120.01607586757218371, 0.34488126401045993, 0.64552185106557303]}
ROSMsgInfoPair:
left:
width: 752, height: 480
distortion_model: KANNALA_BRANDT
D: 0.00986114,-0.119372,0.190923,-0.101683,0,
K: 356.416,0,375.767,0,356.311,246.2,0,0,1,
```

(continues on next page)

(continued from previous page)

```
R: 0.999919,-0.00246361,-0.0124477,0.00245407,0.999997,-0.000781093,0.0124496,0.
↪000750482,0.999922,
P: 357.04,0,511.114,0,0,357.04,311.965,0,0,0,1,0,

right:
width: 752, height: 480
distortion_model: KANNALA_BRANDT
D: -0.0224631,0.0130339,-0.0173598,0.00675133,0,
K: 357.968,0,397.093,0,357.769,258.94,0,0,1,
R: 0.999981,-0.00287357,-0.00537853,0.00287782,0.999996,0.000781842,0.00537626,-0.
↪000797306,0.999985,
P: 357.04,0,511.114,-42851.3,0,357.04,311.965,0,0,0,1,0,
```

Complete code examples, see get_img_params.cc .

2.3.3 Get IMU Calibration Parameters

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

Reference commands:

```
auto &&api = API::Create(argc, argv);

LOG(INFO) << "Motion intrinsics: {" << api->GetMotionIntrinsics() << "}";
LOG(INFO) << "Motion extrinsics left to imu: {"
          << api->GetMotionExtrinsics(Stream::LEFT) << "}";
```

Complete code examples, see get_imu_params.cc .

2.3.4 Get Original Binocular Image

Use `Start()` or `Stop()` , to start or stop data capturing. If you only need the image data, use `Source::VIDEO_STREAMING` .

When data capturing starts, call `WaitForStreams()` function. Once data capturing begins, use `GetStreamData()` to get your data.

Reference commands:

```
auto &&api = API::Create(argc, argv);

api->Start(Source::VIDEO_STREAMING);

cv::namedWindow("frame");

while (true) {
  api->WaitForStreams();

  auto &&left_data = api->GetStreamData(Stream::LEFT);
  auto &&right_data = api->GetStreamData(Stream::RIGHT);

  cv::Mat img;
```

(continues on next page)

(continued from previous page)

```
  cv::hconcat(left_data.frame, right_data.frame, img);
  cv::imshow("frame", img);

  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') {  // ESC/Q
    break;
  }
}

api->Stop(Source::VIDEO_STREAMING);
```

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

Complete code examples, see get_stereo.cc .

2.3.5 Get Stereo Camera Correction Image

The `GetStreamData()` API provided can only get the raw data of the hardware, for example, the stereo camera raw image.

The stereo camera correction image belongs to the upper layer of synthetic data. For such data, you need to enable `EnableStreamData()` before you can get `GetStreamData()`.

In addition, `WaitForStreams()` waits for the key of the raw data. At the beginning when the synthetic data may still being processed, the value taken out will be null, so it needs to check not empty.

Tip: If you want the synthetic data once it is generated, see get_from_callbacks.

Reference code snippet:

```
auto &&api = API::Create(argc, argv);

api->EnableStreamData(Stream::LEFT_RECTIFIED);
api->EnableStreamData(Stream::RIGHT_RECTIFIED);

api->Start(Source::VIDEO_STREAMING);

cv::namedWindow("frame");

while (true) {
  api->WaitForStreams();

  auto &&left_data = api->GetStreamData(Stream::LEFT_RECTIFIED);
  auto &&right_data = api->GetStreamData(Stream::RIGHT_RECTIFIED);

  if (!left_data.frame.empty() && !right_data.frame.empty()) {
    cv::Mat img;
    cv::hconcat(left_data.frame, right_data.frame, img);
    cv::imshow("frame", img);
  }
```

(continues on next page)

(continued from previous page)

```
  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') {  // ESC/Q
    break;
  }
}

api->Stop(Source::VIDEO_STREAMING);
```

OpenCV is used to display the image above. Select the display window, press ESC/Q to exit the program.

Complete code examples, see get_stereo_rectified.cc .

2.3.6 Get Disparity Image

Disparity image belongs to the upper layer of synthetic data. You need to start the `EnableStreamData()` beforehand, to get it through `GetStreamData()`. In addition, it should be check not be empty before use.

For detailed process description, please see get_stereo get_stereo_rectified .

It is recommended to use plugin to calculate depth: the depth map will be better with a higher frame rate. Please see get_with_plugin .

Tip: The SetDisparityComputingMethodType method is used to change disparity computing method. Currently, BM and SGBM are available.

Reference code snippet:

```
auto &&api = API::Create(argc, argv);

// api->EnableStreamData(Stream::DISPARITY);
api->EnableStreamData(Stream::DISPARITY_NORMALIZED);

api->SetDisparityComputingMethodType(DisparityComputingMethod::BM);

api->Start(Source::VIDEO_STREAMING);

cv::namedWindow("frame");
// cv::namedWindow("disparity");
cv::namedWindow("disparity_normalized");

while (true) {
  api->WaitForStreams();

  auto &&left_data = api->GetStreamData(Stream::LEFT);
  auto &&right_data = api->GetStreamData(Stream::RIGHT);

  cv::Mat img;
  cv::hconcat(left_data.frame, right_data.frame, img);
  cv::imshow("frame", img);

  // auto &&disp_data = api->GetStreamData(Stream::DISPARITY);
  // if (!disp_data.frame.empty()) {
```

(continues on next page)

(continued from previous page)

```
  //   cv::imshow("disparity", disp_data.frame);
  // }

  auto &&disp_norm_data = api->GetStreamData(Stream::DISPARITY_NORMALIZED);
  if (!disp_norm_data.frame.empty()) {
    cv::imshow("disparity_normalized", disp_norm_data.frame);  // CV_8UC1
  }

  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') {  // ESC/Q
    break;
  }
}

api->Stop(Source::VIDEO_STREAMING);
```

The above code uses OpenCV to display the image. Select the display window, press ESC/Q to exit in the program.

Complete code examples, see get_disparity.cc .

2.3.7 Get Depth Image

Depth images belongs to the upper layer of synthetic data. You need to start the `EnableStreamData()` beforehand, to get it through `GetStreamData()`. The depth image type is CV_16UC1. In addition, it should be check not be empty before use.

For detailed process description, please see get_stereo get_stereo_rectified.

In addition, it is recommended to use plugins to calculate depth: Depth images work better and operate faster. Please refer to get_with_plugin.

Reference code snippet:

```
auto &&api = API::Create(argc, argv);

api->EnableStreamData(Stream::DEPTH);

api->Start(Source::VIDEO_STREAMING);

cv::namedWindow("frame");
cv::namedWindow("depth");

while (true) {
  api->WaitForStreams();

  auto &&left_data = api->GetStreamData(Stream::LEFT);
  auto &&right_data = api->GetStreamData(Stream::RIGHT);

  cv::Mat img;
  cv::hconcat(left_data.frame, right_data.frame, img);
  cv::imshow("frame", img);

  auto &&depth_data = api->GetStreamData(Stream::DEPTH);
```

(continues on next page)

(continued from previous page)

```
  if (!depth_data.frame.empty()) {
    cv::imshow("depth", depth_data.frame);  // CV_16UC1
  }

  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') {  // ESC/Q
    break;
  }
}

api->Stop(Source::VIDEO_STREAMING);
```

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

Complete code examples, see get_depth.cc .

Preview the value of a region of the depth image, see get_depth_with_region.cc .

2.3.8 Get Point Image

Point images belongs to upper layer of synthetic data. To get this kind of data through `GetStreamData()`, you need to start the `EnableStreamData()` beforehand. It should be check not empty before use.

For detail process description, please see get_stereo get_stereo_rectified .

Sample code snippet:

```
auto &&api = API::Create(argc, argv);

api->EnableStreamData(Stream::POINTS);

api->Start(Source::VIDEO_STREAMING);

cv::namedWindow("frame");
PCViewer pcviewer;

while (true) {
  api->WaitForStreams();

  auto &&left_data = api->GetStreamData(Stream::LEFT);
  auto &&right_data = api->GetStreamData(Stream::RIGHT);

  cv::Mat img;
  cv::hconcat(left_data.frame, right_data.frame, img);
  cv::imshow("frame", img);

  auto &&points_data = api->GetStreamData(Stream::POINTS);
  if (!points_data.frame.empty()) {
    pcviewer.Update(points_data.frame);
  }

  char key = static_cast<char>(cv::waitKey(1));
```

(continues on next page)

(continued from previous page)

```
  if (key == 27 || key == 'q' || key == 'Q') {  // ESC/Q
    break;
  }
  if (pcviewer.WasStopped()) {
    break;
  }
}

api->Stop(Source::VIDEO_STREAMING);
```

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

Complete code examples, see get_depth_and_points.cc .

Attention: Sample code only compiles when PCL is ready. If your PCL was installed in a different directory, please set `CMAKE_PREFIX_PATH` in CMakeLists.txt to the path of `PCLConfig.cmake` . You can find `CMAKE_PREFIX_PATH` near `find_package(PCL)` .

2.3.9 Get IMU Data

The API offers `Start()` / `Stop()` function to start/stop capturing data. You can set the argument to``Source::MOTION_TRACKING`` to capture IMU data only, or set it to `Source::ALL` to capture both image and IMU data.

During capturing 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_from_callbacks .

Sample code snippet:

```
auto &&api = API::Create(argc, argv);

// Enable this will cache the motion datas until you get them.
api->EnableMotionDatas();

api->Start(Source::ALL);

CVPainter painter;

cv::namedWindow("frame");

while (true) {
  api->WaitForStreams();

  auto &&left_data = api->GetStreamData(Stream::LEFT);
  auto &&right_data = api->GetStreamData(Stream::RIGHT);

  cv::Mat img;
  cv::hconcat(left_data.frame, right_data.frame, img);

  auto &&motion_datas = api->GetMotionDatas();
  /*
```

(continues on next page)

(continued from previous page)

```
  for (auto &&data : motion_datas) {
    LOG(INFO) << "Imu frame_id: " << data.imu->frame_id
              << ", timestamp: " << data.imu->timestamp
              << ", accel_x: " << data.imu->accel[0]
              << ", accel_y: " << data.imu->accel[1]
              << ", accel_z: " << data.imu->accel[2]
              << ", gyro_x: " << data.imu->gyro[0]
              << ", gyro_y: " << data.imu->gyro[1]
              << ", gyro_z: " << data.imu->gyro[2]
              << ", temperature: " << data.imu->temperature;
  }
  */

  painter.DrawImgData(img, *left_data.img);
  static std::vector<api::MotionData> motion_datas_s = motion_datas;

  if (!motion_datas.empty() && motion_datas.size() > 0) {
    motion_datas_s = motion_datas;
  }
  if (!motion_datas_s.empty() && motion_datas_s.size() > 0) {
    painter.DrawImuData(img, *motion_datas_s[0].imu);
  }

  cv::imshow("frame", img);

  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') {  // ESC/Q
    break;
  }
}

api->Stop(Source::ALL);
```

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 .

2.3.10 Get IMU Data With Timestamp Correspondence

If wanna get image with timestamp in the middle of IMU datas, you could call *EnableTimestampCorrespondence()* ` to enable this feature.

Reference code snippet:

```
auto &&api = API::Create(argc, argv);

// Enable motion datas with timestamp correspondence of some stream
api->EnableTimestampCorrespondence(Stream::LEFT);

api->Start(Source::ALL);

cv::namedWindow("frame");
```

(continues on next page)

(continued from previous page)

```
while (true) {
  api->WaitForStreams();

  auto &&left_data = api->GetStreamData(Stream::LEFT);
  auto &&right_data = api->GetStreamData(Stream::RIGHT);

  auto img_stamp = left_data.img->timestamp;
  LOG(INFO) << "Img timestamp: " << img_stamp
      << ", diff_prev=" << (img_stamp - prev_img_stamp);
  prev_img_stamp = img_stamp;

  cv::Mat img;
  cv::hconcat(left_data.frame, right_data.frame, img);

  auto &&motion_datas = api->GetMotionDatas();
  LOG(INFO) << "Imu count: " << motion_datas.size();
  for (auto &&data : motion_datas) {
    auto imu_stamp = data.imu->timestamp;
    LOG(INFO) << "Imu timestamp: " << imu_stamp
        << ", diff_prev=" << (imu_stamp - prev_imu_stamp)
        << ", diff_img=" << (1.f + imu_stamp - img_stamp);
    prev_imu_stamp = imu_stamp;
  }
  LOG(INFO);

  cv::imshow("frame", img);

  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') {  // ESC/Q
    break;
  }
}

api->Stop(Source::ALL);
```

Reference result on Linux:

```
$ ./samples/_output/bin/get_imu_correspondence
I/utils.cc:30 Detecting MYNT EYE devices
I/utils.cc:40 MYNT EYE devices:
I/utils.cc:43   index: 0, name: MYNT-EYE-S1030, sn: 0281351000090807
I/utils.cc:51 Only one MYNT EYE device, select index: 0
I/synthetic.cc:126 camera calib model: kannala_brandt
I/utils.cc:79 MYNT EYE devices:
I/utils.cc:82   index: 0, request: width: 752, height: 480, format: Format::YUYV, fps: 60
I/utils.cc:87 Only one stream request, select index: 0
I/get_imu_correspondence.cc:50 Img timestamp: 171323050, diff_prev=39990
I/get_imu_correspondence.cc:58 Imu count: 13
I/get_imu_correspondence.cc:61 Imu timestamp: 171318710, diff_prev=171318710, diff_img=-
↪4352
I/get_imu_correspondence.cc:61 Imu timestamp: 171320730, diff_prev=2020, diff_img=-2320
```

(continues on next page)

(continued from previous page)

```
I/get_imu_correspondence.cc:61 Imu timestamp: 171322750, diff_prev=2020, diff_img=-304
I/get_imu_correspondence.cc:61 Imu timestamp: 171324770, diff_prev=2020, diff_img=1712
I/get_imu_correspondence.cc:61 Imu timestamp: 171326790, diff_prev=2020, diff_img=3728
I/get_imu_correspondence.cc:61 Imu timestamp: 171328800, diff_prev=2010, diff_img=5744
I/get_imu_correspondence.cc:61 Imu timestamp: 171330810, diff_prev=2010, diff_img=7760
I/get_imu_correspondence.cc:61 Imu timestamp: 171332840, diff_prev=2030, diff_img=9776
I/get_imu_correspondence.cc:61 Imu timestamp: 171334860, diff_prev=2020, diff_img=11808
I/get_imu_correspondence.cc:61 Imu timestamp: 171336880, diff_prev=2020, diff_img=13824
I/get_imu_correspondence.cc:61 Imu timestamp: 171338900, diff_prev=2020, diff_img=15840
I/get_imu_correspondence.cc:61 Imu timestamp: 171340920, diff_prev=2020, diff_img=17872
I/get_imu_correspondence.cc:61 Imu timestamp: 171342930, diff_prev=2010, diff_img=19872
I/get_imu_correspondence.cc:66
I/get_imu_correspondence.cc:50 Img timestamp: 171403040, diff_prev=79990
I/get_imu_correspondence.cc:58 Imu count: 20
I/get_imu_correspondence.cc:61 Imu timestamp: 171383310, diff_prev=40380, diff_img=-19728
I/get_imu_correspondence.cc:61 Imu timestamp: 171385330, diff_prev=2020, diff_img=-17712
I/get_imu_correspondence.cc:61 Imu timestamp: 171387350, diff_prev=2020, diff_img=-15696
I/get_imu_correspondence.cc:61 Imu timestamp: 171389370, diff_prev=2020, diff_img=-13664
I/get_imu_correspondence.cc:61 Imu timestamp: 171391380, diff_prev=2010, diff_img=-11664
I/get_imu_correspondence.cc:61 Imu timestamp: 171393390, diff_prev=2010, diff_img=-9648
I/get_imu_correspondence.cc:61 Imu timestamp: 171395420, diff_prev=2030, diff_img=-7616
I/get_imu_correspondence.cc:61 Imu timestamp: 171397440, diff_prev=2020, diff_img=-5600
I/get_imu_correspondence.cc:61 Imu timestamp: 171399460, diff_prev=2020, diff_img=-3584
I/get_imu_correspondence.cc:61 Imu timestamp: 171401480, diff_prev=2020, diff_img=-1568
I/get_imu_correspondence.cc:61 Imu timestamp: 171403500, diff_prev=2020, diff_img=464
I/get_imu_correspondence.cc:61 Imu timestamp: 171405510, diff_prev=2010, diff_img=2464
I/get_imu_correspondence.cc:61 Imu timestamp: 171407520, diff_prev=2010, diff_img=4480
I/get_imu_correspondence.cc:61 Imu timestamp: 171409540, diff_prev=2020, diff_img=6496
I/get_imu_correspondence.cc:61 Imu timestamp: 171411570, diff_prev=2030, diff_img=8528
I/get_imu_correspondence.cc:61 Imu timestamp: 171413590, diff_prev=2020, diff_img=10544
I/get_imu_correspondence.cc:61 Imu timestamp: 171415610, diff_prev=2020, diff_img=12576
I/get_imu_correspondence.cc:61 Imu timestamp: 171417630, diff_prev=2020, diff_img=14592
I/get_imu_correspondence.cc:61 Imu timestamp: 171419650, diff_prev=2020, diff_img=16608
I/get_imu_correspondence.cc:61 Imu timestamp: 171421660, diff_prev=2010, diff_img=18624
```

Complete code examples, see get_imu_correspondence.cc .

2.3.11 Get Data From Callbacks

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

> **Attention:** Make sure to not block callback. If the data processing time is too long, use the callback as a data producer.

Reference code snippet:

```
auto &&api = API::Create(argc, argv);

// Attention: must not block the callbacks.

```

(continues on next page)

(continued from previous page)

```
// Get left image from callback
std::atomic_uint left_count(0);
api->SetStreamCallback(
    Stream::LEFT, [&left_count](const api::StreamData &data) {
      CHECK_NOTNULL(data.img);
      ++left_count;
    });

// Get depth image from callback
api->EnableStreamData(Stream::DEPTH);
std::atomic_uint depth_count(0);
cv::Mat depth;
std::mutex depth_mtx;
api->SetStreamCallback(
    Stream::DEPTH,
    [&depth_count, &depth, &depth_mtx](const api::StreamData &data) {
      UNUSED(data)
      ++depth_count;
      {
        std::lock_guard<std::mutex> _(depth_mtx);
        depth = data.frame;
      }
    });

// Get motion data from callback
std::atomic_uint imu_count(0);
std::shared_ptr<mynteye::ImuData> imu;
std::mutex imu_mtx;
api->SetMotionCallback(
    [&imu_count, &imu, &imu_mtx](const api::MotionData &data) {
      CHECK_NOTNULL(data.imu);
      ++imu_count;
      {
        std::lock_guard<std::mutex> _(imu_mtx);
        imu = data.imu;
      }
    });

api->Start(Source::ALL);

CVPainter painter;

cv::namedWindow("frame");
cv::namedWindow("depth");

unsigned int depth_num = 0;
while (true) {
  api->WaitForStreams();

  auto &&left_data = api->GetStreamData(Stream::LEFT);
  auto &&right_data = api->GetStreamData(Stream::RIGHT);
```

(continues on next page)

(continued from previous page)

```
  // Concat left and right as img
  cv::Mat img;
  cv::hconcat(left_data.frame, right_data.frame, img);

  // Draw img data and size
  painter.DrawImgData(img, *left_data.img);

  // Draw imu data
  if (imu) {
    std::lock_guard<std::mutex> _(imu_mtx);
    painter.DrawImuData(img, *imu);
  }

  // Draw counts
  std::ostringstream ss;
  ss << "left: " << left_count << ", depth: " << depth_count
     << ", imu: " << imu_count;
  painter.DrawText(img, ss.str(), CVPainter::BOTTOM_RIGHT);

  // Show img
  cv::imshow("frame", img);

  // Show depth
  if (!depth.empty()) {
    // Is the depth a new one?
    if (depth_num != depth_count || depth_num == 0) {
      std::lock_guard<std::mutex> _(depth_mtx);
      depth_num = depth_count;
      // LOG(INFO) << "depth_num: " << depth_num;
      ss.str("");
      ss.clear();
      ss << "depth: " << depth_count;
      painter.DrawText(depth, ss.str());
      cv::imshow("depth", depth);  // CV_16UC1
    }
  }

  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') {  // ESC/Q
    break;
  }
}

api->Stop(Source::ALL);
```

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 .

2.3.12 Using The Plugin To Get Data

API provides a `EnablePlugin()` function to enable plugins under a path.

Official provided plugins for calculating binocular parallax are now available in the MYNTEYE_BOX located in the `Plugins` directory.

```
Plugins/
├─linux-x86_64/
│   ├─libplugin_b_ocl1.2_opencv3.4.0.so
│   ├─libplugin_g_cuda9.1_opencv2.4.13.5.so
│   ├─libplugin_g_cuda9.1_opencv3.3.1.so
│   └─libplugin_g_cuda9.1_opencv3.4.0.so
├─tegra-armv8/
└─win-x86_64/
```

- The `linux-x86_64` directory shows the system and architecture.
  - You can find your CPU architecture from system information or `uname -a`.
- The library name `libplugin_*` shows the plugin identity and the third party dependency.
  - `b g` is a plugin identifier, indicating that different algorithms are used.
  - `ocl1.2` shows it dependence on `OpenCL 1.2`, if it exists.
  - `cuda9.1` shows it dependence on `CUDA 9.1`, if it exists.
  - `opencv3.4.0` shows it dependence on `OpenCV 3.4.0`, if it exists.
  - `mynteye2.0.0` shows it dependency on `MYNT EYE SDK 2.0.0`, if it exists.

First, select the plugins that you are going to use depending on your situation. If you relying on third parties, please install a corresponding version.

Then, enable the plugin with the following code:

```
auto &&api = API::Create(argc, argv);

api->EnablePlugin("plugins/linux-x86_64/libplugin_g_cuda10.1_opencv3.4.1.so");
```

The path can be an absolute path or a relative path (relative to the current working directory).

Finally, just call the API to get the data as before.

Tip: If the plugin is not enabled, `api->Start(Source::VIDEO_STREAMING);` will automatically find the appropriate plug-in in the `<sdk>/plugins/<platform>` directory to load.

In other words, you can move the plug-in directory of the current platform into the `< SDK > / plugins` directory. To automatically load the official plugin, install the corresponding `CUDA OpenCV` plugin dependency, recompiling and then run `API` layer interface program.

Before running, please execute the following commands to ensure that the plugin's dependency library can be searched:

```
# Linux
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
# /usr/local/lib means the path of dependency library

# macOS
```

(continues on next page)

(continued from previous page)

```
export DYLD_LIBRARY_PATH=/usr/local/lib:$DYLD_LIBRARY_PATH
# /usr/local/lib means the path of dependency library

# Windows
set PATH=C:\opencv\x64\vc14\bin;%PATH%
# add to PATH of system environment
```

In addition, the following command can be executed to check whether the dependency Library of the plug-in can be searched:

```
# Linux
ldd *.so
# *.so means plugin path

# macOS
otool -L *.dylib
# *.dylib means plugin path

# Windows
# please download Dependency Walker open DLL .
```

If the plugin's dependent library is not found, it will report an error "Open plugin failed" when loading.

Complete code sample, see get_with_plugin.cc .

Tip: Linux can also add a dependency library path to the system environment, so that the compiled program can run directly. (does not require `export LD_LIBRARY_PATH` in the terminal then run again).

- Create a `/etc/ld.so.conf.d/libmynteye.conf` file and write the dependent library path.
- Execute the `sudo /sbin/ldconfig` command in the terminal and refresh the cache.

2.3.13 Save Device Infomation And Parameters

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

Reference commands:

```
./samples/_output/bin/save_all_infos

# Windows
.\samples\_output\bin\save_all_infos.bat
```

Reference result on Linux:

```
$ ./samples/_output/bin/save_all_infos
I0512 21:40:08.687088  4092 utils.cc:26] Detecting MYNT EYE devices
I0512 21:40:09.366693  4092 utils.cc:33] MYNT EYE devices:
I0512 21:40:09.366734  4092 utils.cc:37]   index: 0, name: MYNT-EYE-S1000
I0512 21:40:09.366757  4092 utils.cc:43] Only one MYNT EYE device, select index: 0
I0512 21:40:09.367609  4092 save_all_infos.cc:38] Save all infos to "config/
↪SN0610243700090720"
```

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
      ├─img.params
      └─imu.params
```

2.3.14 Save Single Image

Press "Space" "s" "S" to save image.

Reference commands:

```
api->Start(Source::VIDEO_STREAMING);

cv::namedWindow("frame");

std::int32_t count = 0;
std::cout << "Press 'Space' 's' 'S' to save image." << std::endl;
while (true) {
  api->WaitForStreams();

  auto &&left_data = api->GetStreamData(Stream::LEFT);
  auto &&right_data = api->GetStreamData(Stream::RIGHT);
  if (!left_data.frame.empty() && !right_data.frame.empty()) {
    cv::Mat img;
    cv::hconcat(left_data.frame, right_data.frame, img);
    cv::imshow("frame", img);
  }

  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') {  // ESC/Q
    break;
  } else if (key == 32 || key == 's' || key == 'S') {
    if (!left_data.frame.empty() && !right_data.frame.empty()) {
      char l_name[20];
      char r_name[20];
      ++count;
      snprintf(l_name, sizeof(l_name), "left_%d.jpg", count);
      snprintf(r_name, sizeof(r_name), "right_%d.jpg", count);

      cv::imwrite(l_name, left_data.frame);
      cv::imwrite(r_name, right_data.frame);

      std::cout << "Saved " << l_name << " " << r_name << " to current directory" <<␣
↪std::endl;
    }
  }
}
```

(continues on next page)

(continued from previous page)

```
api->Stop(Source::VIDEO_STREAMING);
```

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

Complete code examples, see save_single_image.cc .

2.3.15 Write Image Parameters

The SDK provides a tool `write_img_params` for writing image parameters.

For getting image parameters, please read get_img_params. This is used to calculate the deviation.

Reference commands:

```
./samples/_output/bin/write_img_params samples/config/img.params

# Windows
.\samples\_output\bin\write_img_params.bat samples\config\img.params
```

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

And, samples/config/S1030/img.params.pinhole is the path of S1030 pihole parameters file. If you calibrated parameters yourself, you can edit it and run previous commands to write them into the devices.

Tip: The image calibration parameters of S21XX are in `samples/config/S21XX` The image calibration parameters of S1030 are in `samples/config/S1030`

Tip: You can also write into devices with `SN*.conf` provided by old SDK.

2.3.16 Write IMU Parameters

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

Information about how to get IMU parameters, please read get_imu_params .

Reference commands:

```
./samples/_output/bin/write_imu_params samples/config/imu.params

# Windows
.\samples\_output\bin\write_imu_params.bat samples\config\imu.params
```

The path of parameters folder can be found in samples/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.

2.4 SDK Control Samples

2.4.1 Set the frame rate of image & IMU frequency

Using the `SetOptionValue()` function in the API, you can set various control valuesfor the current device.

For mynteye s1030, to set the image frame rate and IMU frequency, set `Option::FRAME_RATE` and `Option::IMU_FREQUENCY`.

Attention:

- The effective fps of the image: 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60.
- The effective frequency of IMU: 100, 200, 250, 333, 500.

For mynteye s21XX, the image frame rate should be selected when running the sample, and the frame rate and resolution are combined as follows:

```
index: 0, request: width: 1280, height: 400, format: Format::BGR888, fps: 10
index: 1, request: width: 1280, height: 400, format: Format::BGR888, fps: 20
index: 2, request: width: 1280, height: 400, format: Format::BGR888, fps: 30
index: 3, request: width: 1280, height: 400, format: Format::BGR888, fps: 60
index: 4, request: width: 2560, height: 800, format: Format::BGR888, fps: 10
index: 5, request: width: 2560, height: 800, format: Format::BGR888, fps: 20
index: 6, request: width: 2560, height: 800, format: Format::BGR888, fps: 30
```

Reference Code:

s1030

```
auto &&api = API::Create(argc, argv);

// Attention: must set FRAME_RATE and IMU_FREQUENCY together, otherwise won't
// succeed.

// FRAME_RATE values: 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60
api->SetOptionValue(Option::FRAME_RATE, 25);
// IMU_FREQUENCY values: 100, 200, 250, 333, 500
api->SetOptionValue(Option::IMU_FREQUENCY, 500);

LOG(INFO) << "Set FRAME_RATE to " << api->GetOptionValue(Option::FRAME_RATE);
LOG(INFO) << "Set IMU_FREQUENCY to "
          << api->GetOptionValue(Option::IMU_FREQUENCY);
```

s21XX

```
auto &&api = API::Create(argc, argv);
if (!api) return 1;
```

(continues on next page)

(continued from previous page)

```
bool ok;
auto &&request = api->SelectStreamRequest(&ok);
if (!ok) return 1;
api->ConfigStreamRequest(request);

LOG(INFO) << "Please set frame rate by 'SelectStreamRequest()'";
```

Reference running results on Linux:

s1030

```
$ ./samples/_output/bin/ctrl_framerate
I0513 14:05:57.218222 31813 utils.cc:26] Detecting MYNT EYE devices
I0513 14:05:57.899404 31813 utils.cc:33] MYNT EYE devices:
I0513 14:05:57.899430 31813 utils.cc:37]   index: 0, name: MYNT-EYE-S1000
I0513 14:05:57.899435 31813 utils.cc:43] Only one MYNT EYE device, select index: 0
I0513 14:05:58.076257 31813 framerate.cc:36] Set FRAME_RATE to 25
I0513 14:05:58.076836 31813 framerate.cc:37] Set IMU_FREQUENCY to 500
I0513 14:06:21.702361 31813 framerate.cc:82] Time beg: 2018-05-13 14:05:58.384967, end:␣
↪2018-05-13 14:06:21.666115, cost: 23281.1ms
I0513 14:06:21.702388 31813 framerate.cc:85] Img count: 573, fps: 24.6122
I0513 14:06:21.702404 31813 framerate.cc:87] Imu count: 11509, hz: 494.348
```

s21XX

```
$ ./samples/_output/bin/ctrl_framerate
I/utils.cc:30 Detecting MYNT EYE devices
I/utils.cc:40 MYNT EYE devices:
I/utils.cc:43   index: 0, name: MYNT-EYE-S210A, sn: 07C41A190009071F
I/utils.cc:51 Only one MYNT EYE device, select index: 0
I/utils.cc:79 MYNT EYE devices:
I/utils.cc:82   index: 0, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 10
I/utils.cc:82   index: 1, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 20
I/utils.cc:82   index: 2, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 30
I/utils.cc:82   index: 3, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 60
I/utils.cc:82   index: 4, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 10
I/utils.cc:82   index: 5, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 20
I/utils.cc:82   index: 6, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 30
I/utils.cc:93 There are 7 stream requests, select index:
2
I/framerate.cc:54 Please set frame rate by 'SelectStreamRequest()'
I/framerate.cc:99 Time beg: 2018-12-29 10:05:08.203095, end: 2018-12-29 10:08:20.074969,␣
↪cost: 191872ms
I/framerate.cc:102 Img count: 5759, fps: 30.0148
I/framerate.cc:104 Imu count: 77163, hz: 402.159
```

After the sample program finishes running with ESC/Q, it will output the calculated value of the frame rate of image

& IMU frequency.

Complete code samplesplease see ctrl_framerate.cc .

2.4.2 Set the range of accelerometer & gyroscope

Using the `SetOptionValue()` function in the API, you can set various control values for the current device.

To set the range of accelerometer and gyroscope, set `Option::ACCELEROMETER_RANGE` and `Option::GYROSCOPE_RANGE`.

Attention: For mynteye s1030, the available settings are:

- The effective range of accelerometer(unit:g): 4, 8, 16, 32.
- Gyroscope Range Valid value (unit: DEG/S): 500, 1000, 2000, 4000.

For mynteye s21XX, the available settings are:

- The effective range of accelerometer(unit:g): 6, 12, 24, 48.
- The effective range of gyroscope(unit:deg/s): 250, 500, 1000, 2000, 4000.

Reference Code:

s1030

```
auto &&api = API::Create(argc, argv);
if (!api)
  return 1;

// ACCELEROMETER_RANGE values: 4, 8, 16, 32
api->SetOptionValue(Option::ACCELEROMETER_RANGE, 8);
// GYROSCOPE_RANGE values: 500, 1000, 2000, 4000
api->SetOptionValue(Option::GYROSCOPE_RANGE, 1000);

LOG(INFO) << "Set ACCELEROMETER_RANGE to "
          << api->GetOptionValue(Option::ACCELEROMETER_RANGE);
LOG(INFO) << "Set GYROSCOPE_RANGE to "
          << api->GetOptionValue(Option::GYROSCOPE_RANGE);
```

s21XX

```
auto &&api = API::Create(argc, argv);
if (!api) return 1;

bool ok;
auto &&request = api->SelectStreamRequest(&ok);
if (!ok) return 1;
api->ConfigStreamRequest(request);

// ACCELEROMETER_RANGE values: 6, 12, 24, 48
api->SetOptionValue(Option::ACCELEROMETER_RANGE, 6);
// GYROSCOPE_RANGE values: 250, 500, 1000, 2000, 4000
api->SetOptionValue(Option::GYROSCOPE_RANGE, 1000);
```

(continues on next page)

(continued from previous page)

```
LOG(INFO) << "Set ACCELEROMETER_RANGE to "
          << api->GetOptionValue(Option::ACCELEROMETER_RANGE);
LOG(INFO) << "Set GYROSCOPE_RANGE to "
          << api->GetOptionValue(Option::GYROSCOPE_RANGE);
```

Reference running results on Linux:

s1030

```
$ ./samples/_output/bin/ctrl_imu_range
I/utils.cc:28 Detecting MYNT EYE devices
I/utils.cc:38 MYNT EYE devices:
I/utils.cc:41   index: 0, name: MYNT-EYE-S1030, sn: 4B4C1F1100090712
I/utils.cc:49 Only one MYNT EYE device, select index: 0
I/imu_range.cc:34 Set ACCELEROMETER_RANGE to 8
I/imu_range.cc:36 Set GYROSCOPE_RANGE to 1000
I/imu_range.cc:81 Time beg: 2018-11-21 15:34:57.726428, end: 2018-11-21 15:35:12.190478,␣
↪cost: 14464ms
I/imu_range.cc:84 Img count: 363, fps: 25.0967
I/imu_range.cc:86 Imu count: 2825, hz: 195.312
```

s21XX

```
$ ./samples/_output/bin/ctrl_imu_range
I/utils.cc:30 Detecting MYNT EYE devices
I/utils.cc:40 MYNT EYE devices:
I/utils.cc:43   index: 0, name: MYNT-EYE-S210A, sn: 07C41A190009071F
I/utils.cc:51 Only one MYNT EYE device, select index: 0
I/utils.cc:79 MYNT EYE devices:
I/utils.cc:82   index: 0, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 10
I/utils.cc:82   index: 1, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 20
I/utils.cc:82   index: 2, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 30
I/utils.cc:82   index: 3, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 60
I/utils.cc:82   index: 4, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 10
I/utils.cc:82   index: 5, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 20
I/utils.cc:82   index: 6, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 30
I/utils.cc:93 There are 7 stream requests, select index:
3
I/imu_range.cc:51 Set ACCELEROMETER_RANGE to 6
I/imu_range.cc:53 Set GYROSCOPE_RANGE to 1000
I/imu_range.cc:98 Time beg: 2018-12-29 10:03:10.706211, end: 2018-12-29 10:04:12.497427,␣
↪cost: 61791.2ms
I/imu_range.cc:101 Img count: 3706, fps: 59.9762
I/imu_range.cc:103 Imu count: 24873, hz: 402.533
```

After the sample program finishes running with ESC/Q, the ranges of imu setting is complete. The ranges will be kept

inside the hardware and not affected by power off.

Complete code samplesplease see ctrl_imu_range.cc.

2.4.3 Enable auto exposure and its adjustment function

Using the `SetOptionValue()` function of the API, you can set various control values of the current open device.

To enable auto exposure, set `Option::EXPOSURE_MODE` to `0` .

For mynteye s1030, the settings available for adjustment during auto exposure are:

- `Option::MAX_GAIN` Maximum gain.
- `Option::MAX_EXPOSURE_TIME` Maximum exposure time.
- `Option::DESIRED_BRIGHTNESS` Expected brightness.

For mynteye s21XX, the settings available for adjustment during auto exposure are:

- `Option::MAX_GAIN` Maximum gain.
- `Option::MAX_EXPOSURE_TIME` Maximum exposure time.
- `Option::DESIRED_BRIGHTNESS` Expected brightness.
- `Option::MIN_EXPOSURE_TIME` Minimum exposure time.

Reference Code:

s1030

```
auto &&api = API::Create(argc, argv);

// auto-exposure: 0
api->SetOptionValue(Option::EXPOSURE_MODE, 0);

// max_gain: range [0,48], default 48
api->SetOptionValue(Option::MAX_GAIN, 48);
// max_exposure_time: range [0,240], default 240
api->SetOptionValue(Option::MAX_EXPOSURE_TIME, 240);
// desired_brightness: range [0,255], default 192
api->SetOptionValue(Option::DESIRED_BRIGHTNESS, 192);

LOG(INFO) << "Enable auto-exposure";
LOG(INFO) << "Set MAX_GAIN to " << api->GetOptionValue(Option::MAX_GAIN);
LOG(INFO) << "Set MAX_EXPOSURE_TIME to "
          << api->GetOptionValue(Option::MAX_EXPOSURE_TIME);
LOG(INFO) << "Set DESIRED_BRIGHTNESS to "
          << api->GetOptionValue(Option::DESIRED_BRIGHTNESS);
```

s21XX

```
auto &&api = API::Create(argc, argv);

bool ok;
auto &&request = api->SelectStreamRequest(&ok);
if (!ok) return 1;
api->ConfigStreamRequest(request);
```

(continues on next page)

(continued from previous page)

```
// auto-exposure: 0
api->SetOptionValue(Option::EXPOSURE_MODE, 0);

// max_gain: range [0,255], default 8
api->SetOptionValue(Option::MAX_GAIN, 8);
// max_exposure_time: range [0,1000], default 333
api->SetOptionValue(Option::MAX_EXPOSURE_TIME, 333);
// desired_brightness: range [1,255], default 122
api->SetOptionValue(Option::DESIRED_BRIGHTNESS, 122);
// min_exposure_time: range [0,1000], default 0
api->SetOptionValue(Option::MIN_EXPOSURE_TIME, 0);

LOG(INFO) << "Enable auto-exposure";
LOG(INFO) << "Set EXPOSURE_MODE to "
          << api->GetOptionValue(Option::EXPOSURE_MODE);
LOG(INFO) << "Set MAX_GAIN to " << api->GetOptionValue(Option::MAX_GAIN);
LOG(INFO) << "Set MAX_EXPOSURE_TIME to "
          << api->GetOptionValue(Option::MAX_EXPOSURE_TIME);
LOG(INFO) << "Set DESIRED_BRIGHTNESS to "
          << api->GetOptionValue(Option::DESIRED_BRIGHTNESS);
LOG(INFO) << "Set MIN_EXPOSURE_TIME to "
          << api->GetOptionValue(Option::MIN_EXPOSURE_TIME);
```

Reference running results on Linux:

s1030

```
$ ./samples/_output/bin/ctrl_auto_exposure
I0513 14:07:57.963943 31845 utils.cc:26] Detecting MYNT EYE devices
I0513 14:07:58.457536 31845 utils.cc:33] MYNT EYE devices:
I0513 14:07:58.457563 31845 utils.cc:37]   index: 0, name: MYNT-EYE-S1000
I0513 14:07:58.457567 31845 utils.cc:43] Only one MYNT EYE device, select index: 0
I0513 14:07:58.474916 31845 auto_exposure.cc:37] Enable auto-exposure
I0513 14:07:58.491058 31845 auto_exposure.cc:38] Set MAX_GAIN to 48
I0513 14:07:58.505131 31845 auto_exposure.cc:39] Set MAX_EXPOSURE_TIME to 240
I0513 14:07:58.521375 31845 auto_exposure.cc:41] Set DESIRED_BRIGHTNESS to 192
```

s21XX

```
$ ./samples/_output/bin/ctrl_auto_exposure
I/utils.cc:30 Detecting MYNT EYE devices
I/utils.cc:40 MYNT EYE devices:
I/utils.cc:43   index: 0, name: MYNT-EYE-S210A, sn: 07C41A190009071F
I/utils.cc:51 Only one MYNT EYE device, select index: 0
I/utils.cc:79 MYNT EYE devices:
I/utils.cc:82   index: 0, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 10
I/utils.cc:82   index: 1, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 20
I/utils.cc:82   index: 2, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 30
I/utils.cc:82   index: 3, request: width: 1280, height: 400, format: Format::BGR888,␣
```

(continues on next page)

(continued from previous page)

```
↪fps: 60
I/utils.cc:82   index: 4, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 10
I/utils.cc:82   index: 5, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 20
I/utils.cc:82   index: 6, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 30
I/utils.cc:93 There are 7 stream requests, select index:
3
I/auto_exposure.cc:72 Enable auto-exposure
I/auto_exposure.cc:73 Set EXPOSURE_MODE to 0
I/auto_exposure.cc:75 Set MAX_GAIN to 8
I/auto_exposure.cc:76 Set MAX_EXPOSURE_TIME to 333
I/auto_exposure.cc:78 Set DESIRED_BRIGHTNESS to 122
I/auto_exposure.cc:80 Set MIN_EXPOSURE_TIME to 0
```

The sample program displays an image with a real exposure time in the upper left corner, in milliseconds.

Complete code examples, see ctrl_auto_exposure.cc .

2.4.4 Enable manual exposure and its adjustment function

Using the `SetOptionValue()` function of the API, you can set various control values for the current open device.

To enabling manual exposure, set `Option::EXPOSURE_MODE` to 1.

For mynteye s1030, during manual exposure, the settings available for adjustment are:

- `Option::GAIN` Gain.
- `Option::BRIGHTNESS` Brightness (Exposure time).
- `Option::CONTRAST` Contrast (Black level calibration).

For mynteye s21XX, during manual exposure, the settings available for adjustment are:

- `Option::BRIGHTNESS` Brightness (Exposure time).

Reference Code:

s1030

```
auto &&api = API::Create(argc, argv);

// manual-exposure: 1
api->SetOptionValue(Option::EXPOSURE_MODE, 1);

// gain: range [0,48], default 24
api->SetOptionValue(Option::GAIN, 24);
// brightness/exposure_time: range [0,240], default 120
api->SetOptionValue(Option::BRIGHTNESS, 120);
// contrast/black_level_calibration: range [0,254], default 116
api->SetOptionValue(Option::CONTRAST, 116);

LOG(INFO) << "Enable manual-exposure";
LOG(INFO) << "Set GAIN to " << api->GetOptionValue(Option::GAIN);
```

(continues on next page)

(continued from previous page)

```
LOG(INFO) << "Set BRIGHTNESS to " << api->GetOptionValue(Option::BRIGHTNESS);
LOG(INFO) << "Set CONTRAST to " << api->GetOptionValue(Option::CONTRAST);
```

s21XX

```
auto &&api = API::Create(argc, argv);

bool ok;
auto &&request = api->SelectStreamRequest(&ok);
if (!ok) return 1;
api->ConfigStreamRequest(request);

// manual-exposure: 1
api->SetOptionValue(Option::EXPOSURE_MODE, 1);

// brightness/exposure_time: range [1,255], default 70
api->SetOptionValue(Option::BRIGHTNESS, 70);

LOG(INFO) << "Enable manual-exposure";
LOG(INFO) << "Set EXPOSURE_MODE to "
          << api->GetOptionValue(Option::EXPOSURE_MODE);
LOG(INFO) << "Set BRIGHTNESS to "
          << api->GetOptionValue(Option::BRIGHTNESS);
```

Reference running results on Linux:

s1030

```
$ ./samples/_output/bin/ctrl_manual_exposure
I0513 14:09:17.104431 31908 utils.cc:26] Detecting MYNT EYE devices
I0513 14:09:17.501519 31908 utils.cc:33] MYNT EYE devices:
I0513 14:09:17.501551 31908 utils.cc:37]   index: 0, name: MYNT-EYE-S1000
I0513 14:09:17.501562 31908 utils.cc:43] Only one MYNT EYE device, select index: 0
I0513 14:09:17.552918 31908 manual_exposure.cc:37] Enable manual-exposure
I0513 14:09:17.552953 31908 manual_exposure.cc:38] Set GAIN to 24
I0513 14:09:17.552958 31908 manual_exposure.cc:39] Set BRIGHTNESS to 120
I0513 14:09:17.552963 31908 manual_exposure.cc:40] Set CONTRAST to 116
```

s21XX

```
$ ./samples/_output/bin/ctrl_manual_exposure
I/utils.cc:30 Detecting MYNT EYE devices
I/utils.cc:40 MYNT EYE devices:
I/utils.cc:43   index: 0, name: MYNT-EYE-S210A, sn: 07C41A190009071F
I/utils.cc:51 Only one MYNT EYE device, select index: 0
I/utils.cc:79 MYNT EYE devices:
I/utils.cc:82   index: 0, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 10
I/utils.cc:82   index: 1, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 20
I/utils.cc:82   index: 2, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 30
I/utils.cc:82   index: 3, request: width: 1280, height: 400, format: Format::BGR888,␣
```

(continues on next page)

(continued from previous page)

```
↪fps: 60
I/utils.cc:82   index: 4, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 10
I/utils.cc:82   index: 5, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 20
I/utils.cc:82   index: 6, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 30
I/utils.cc:93 There are 7 stream requests, select index:
3
I/manual_exposure.cc:62 Enable manual-exposure
I/manual_exposure.cc:63 Set EXPOSURE_MODE to 1
I/manual_exposure.cc:65 Set BRIGHTNESS to 70
```

The sample program displays an image with a real exposure time in the upper left corner, in milliseconds.

Complete code samplessee ctrl_manual_exposure.cc .

2.4.5 Enable IR and its adjustments function

Using the `SetOptionValue()` function of the API, you can set various control values for the current open device.

Enabling IR is setting `Option::IR_CONTROL` greater than 0. The greater the value, the greater the IR's intensity.

Attention:

- mynteye s21XX doesn't support this feature.

Reference Code:

```
auto &&api = API::Create(argc, argv);

// Detect infrared add-ons
LOG(INFO) << "Support infrared: " << std::boolalpha
          << api->Supports(AddOns::INFRARED);
LOG(INFO) << "Support infrared2: " << std::boolalpha
          << api->Supports(AddOns::INFRARED2);

// Get infrared intensity range
auto &&info = api->GetOptionInfo(Option::IR_CONTROL);
LOG(INFO) << Option::IR_CONTROL << ": {" << info << "}";

// Set infrared intensity value
api->SetOptionValue(Option::IR_CONTROL, 80);
```

Reference running results on Linux:

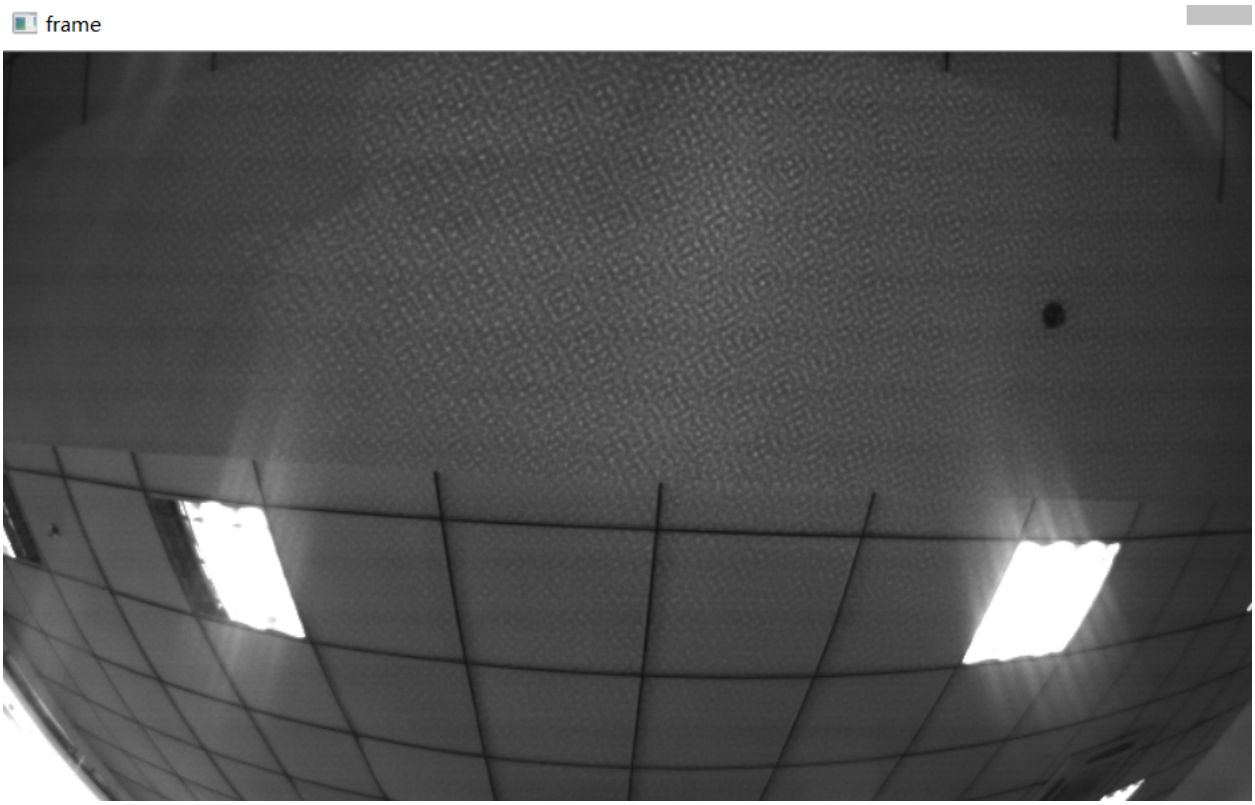
```
$ ./samples/_output/bin/ctrl_infrared
I0504 16:16:28.016624 25848 utils.cc:13] Detecting MYNT EYE devices
I0504 16:16:28.512462 25848 utils.cc:20] MYNT EYE devices:
I0504 16:16:28.512473 25848 utils.cc:24]   index: 0, name: MYNT-EYE-S1000
I0504 16:16:28.512477 25848 utils.cc:30] Only one MYNT EYE device, select index: 0
I0504 16:16:28.520848 25848 infrared.cc:13] Support infrared: true
```

(continues on next page)

(continued from previous page)

```
I0504 16:16:28.520869 25848 infrared.cc:15] Support infrared2: true
I0504 16:16:28.520889 25848 infrared.cc:20] Option::IR_CONTROL: {min: 0, max: 160, def:␣
↪0}
```

At this point, if the image is displayed, you can see IR speckle on the image, as below:



Attention: The hardware will not record the IR value after being turned off and will reset to 0. In order to keep IR enabled, you must set the IR value after turning on the device.

Complete code samplessee ctrl_infrared.cc .

2.4.6 Low-pass Filter

Using the `SetOptionValue()` function in the API, you can set various control values for the current device.

To set the value of accelerometer low-pass filter and gyroscope low-pass filter, set `Option::ACCELEROMETER_LOW_PASS_FILTER` and `Option::GYROSCOPE_LOW_PASS_FILTER`.

Attention:

- s1030 doesn't support this feature

Reference Code:

```
auto &&api = API::Create(argc, argv);
if (!api) return 1;

bool ok;
auto &&request = api->SelectStreamRequest(&ok);
if (!ok) return 1;
api->ConfigStreamRequest(request);

// ACCELEROMETER_RANGE values: 0, 1, 2
api->SetOptionValue(Option::ACCELEROMETER_LOW_PASS_FILTER, 2);
// GYROSCOPE_RANGE values: 23, 64
api->SetOptionValue(Option::GYROSCOPE_LOW_PASS_FILTER, 64);

LOG(INFO) << "Set ACCELEROMETER_LOW_PASS_FILTER to "
          << api->GetOptionValue(Option::ACCELEROMETER_LOW_PASS_FILTER);
LOG(INFO) << "Set GYROSCOPE_LOW_PASS_FILTER to "
          << api->GetOptionValue(Option::GYROSCOPE_LOW_PASS_FILTER);
```

Reference running results on Linux:

```
$ ./samples/_output/bin/ctrl_imu_low_pass_filter
I/utils.cc:30 Detecting MYNT EYE devices
I/utils.cc:40 MYNT EYE devices:
I/utils.cc:43   index: 0, name: MYNT-EYE-S210A, sn: 07C41A190009071F
I/utils.cc:51 Only one MYNT EYE device, select index: 0
I/utils.cc:79 MYNT EYE devices:
I/utils.cc:82   index: 0, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 10
I/utils.cc:82   index: 1, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 20
I/utils.cc:82   index: 2, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 30
I/utils.cc:82   index: 3, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 60
I/utils.cc:82   index: 4, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 10
I/utils.cc:82   index: 5, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 20
I/utils.cc:82   index: 6, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 30
I/utils.cc:93 There are 7 stream requests, select index:
1
I/imu_low_pass_filter.cc:48 Set ACCELEROMETER_LOW_PASS_FILTER to 2
I/imu_low_pass_filter.cc:50 Set GYROSCOPE_LOW_PASS_FILTER to 64
I/imu_low_pass_filter.cc:96 Time beg: 2018-12-29 13:53:42.296299, end: 2018-12-29␣
↪14:06:33.295960, cost: 771000ms
I/imu_low_pass_filter.cc:99 Img count: 15412, fps: 19.9896
I/imu_low_pass_filter.cc:101 Imu count: 309891, hz: 401.934
```

After the sample program finishes running with ESC/Q, the low-pass filter of imu setting is complete. The ranges will be kept inside the hardware and not affected by power off.

Complete code samplesplease see ctrl_imu_low_pass_filter.cc

2.4.7 Set IIC Address

Using the `SetOptionValue()` function in the API, you can set various control values for the current device.

To set the IIC address, set `Option::IIC_ADDRESS_SETTING`.

> **Attention:** Only support S21XX

Reference Code:

s2XX

```
auto &&api = API::Create(argc, argv);
if (!api) return 1;
bool ok;
auto &&request = api->SelectStreamRequest(&ok);
if (!ok) return 1;
api->ConfigStreamRequest(request);
Model model = api->GetModel();
if (model == Model::STANDARD210A || model == Model::STANDARD2) {
  api->SetOptionValue(Option::IIC_ADDRESS_SETTING, 0x31);
  LOG(INFO) << "Set iic address to " << std::hex << "0x"
            << api->GetOptionValue(Option::IIC_ADDRESS_SETTING);
}
```

Reference running results on Linux:

s21XX

```
$ ./samples/_output/bin/ctrl_iic_adress
I/utils.cc:30 Detecting MYNT EYE devices
I/utils.cc:40 MYNT EYE devices:
I/utils.cc:43   index: 0, name: MYNT-EYE-S210A, sn: 07C41A190009071F
I/utils.cc:51 Only one MYNT EYE device, select index: 0
I/utils.cc:79 MYNT EYE devices:
I/utils.cc:82   index: 0, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 10
I/utils.cc:82   index: 1, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 20
I/utils.cc:82   index: 2, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 30
I/utils.cc:82   index: 3, request: width: 1280, height: 400, format: Format::BGR888,␣
↪fps: 60
I/utils.cc:82   index: 4, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 10
I/utils.cc:82   index: 5, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 20
I/utils.cc:82   index: 6, request: width: 2560, height: 800, format: Format::BGR888,␣
↪fps: 30
I/utils.cc:93 There are 7 stream requests, select index:
3
I/imu_range.cc:51 Set iic address to 0x31
```

After the sample program finishes running with ESC/Q. Complete code samplesplease see ctrl_iic_address.cc .

2.5 SDK Project Demos

2.5.1 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>/samples/simple_demo/project_cmake directory .

Preparation

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

Create Project

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

```
cmake_minimum_required(VERSION 3.0)

project(mynteyed_demo VERSION 1.0.0 LANGUAGES C CXX)

# 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")

## mynteye_demo

add_executable(mynteye_demo mynteye_demo.cc)
target_link_libraries(mynteye_demo mynteye ${OpenCV_LIBS})
```

Config Project

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

```
# packages

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

(continues on next page)

(continued from previous page)

```
  endif()
endif()

## mynteye

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

# When SDK build with OpenCV, we can add WITH_OPENCV macro to enable some
# features depending on OpenCV, such as ToMat().
if(mynteye_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 `mynteye_demo` target,

```
# targets

include_directories(
  ${OpenCV_INCLUDE_DIRS}
)

## mynteye_demo

add_executable(mynteye_demo mynteye_demo.cc)
target_link_libraries(mynteye_demo mynteye ${OpenCV_LIBS})
```

Start using SDK

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

Windows

Reference to “Install Build Tools” in install_windows_exe .

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" ..
```

(continues on next page)

(continued from previous page)

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

.\Release\mynteye_demo.exe
```

Linux

Open `Terminal` to build and run.

```
mkdir _build
cd _build/

cmake ..

make

./mynteye_demo
```

2.5.2 How to use SDK with Visual Studio 2017

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

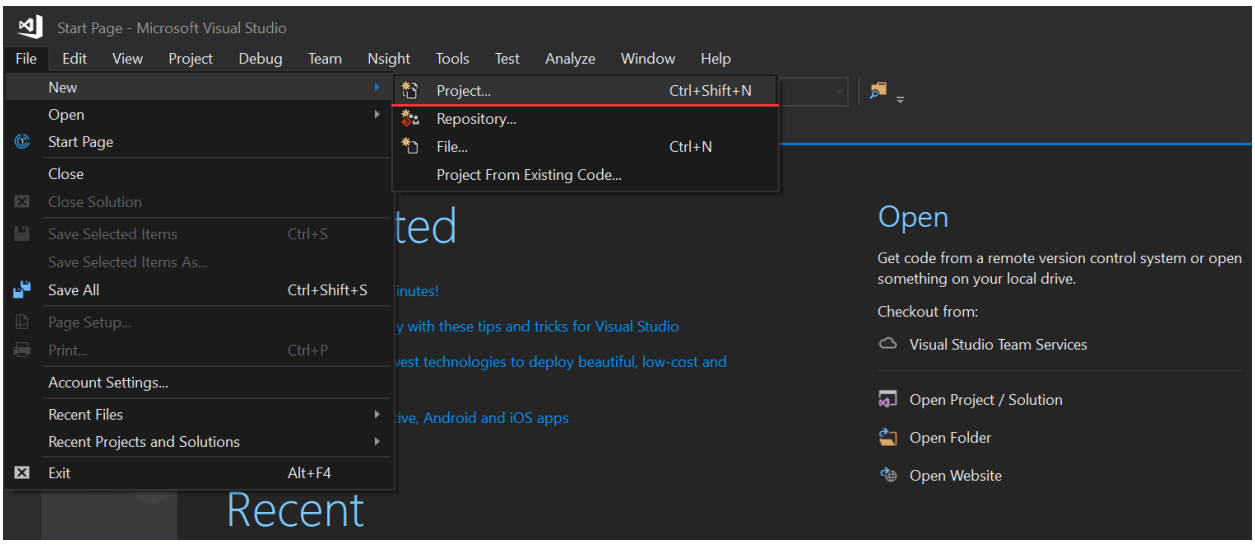
> You could find the project demo in `<sdk>/samples/simple_demo/project_vs2017` directory.

Preparation

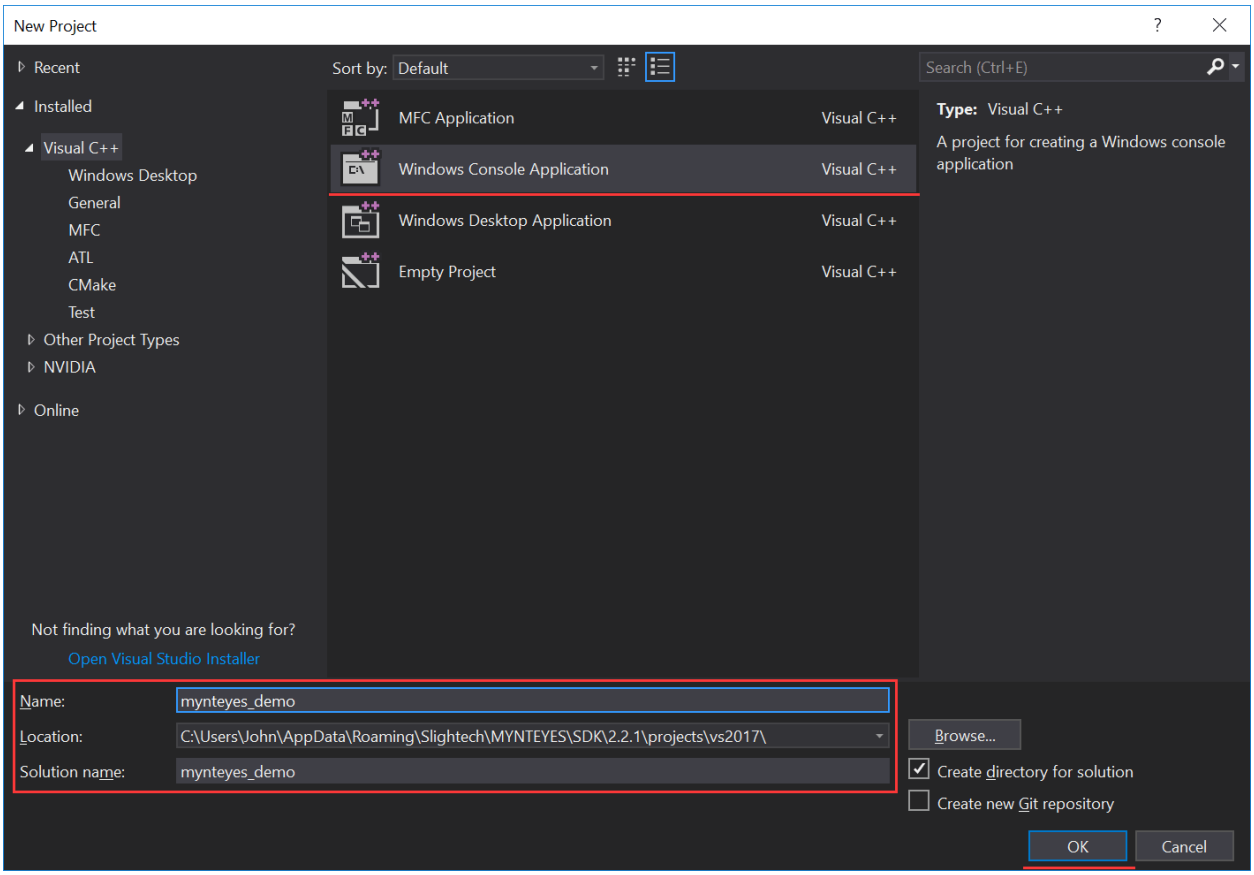
- Windows: install the win pack of SDK

Create Project

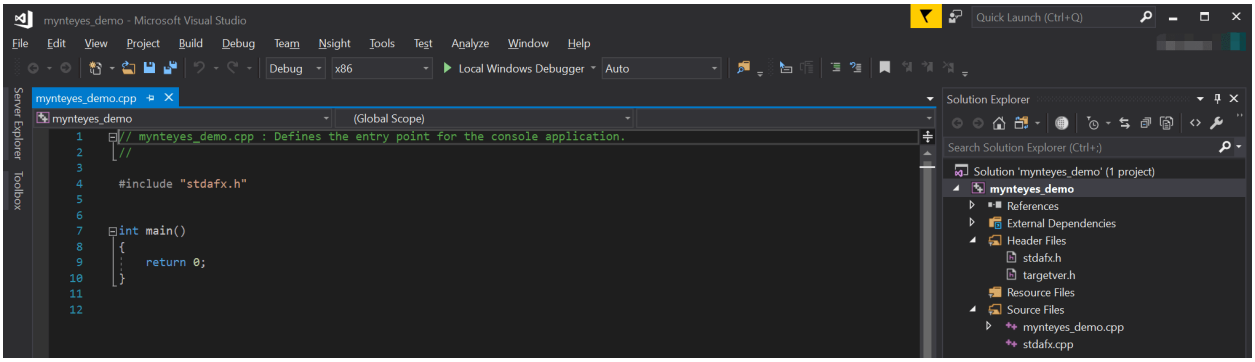
Open Visual Studio 2017, then `File > New > Project`,



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

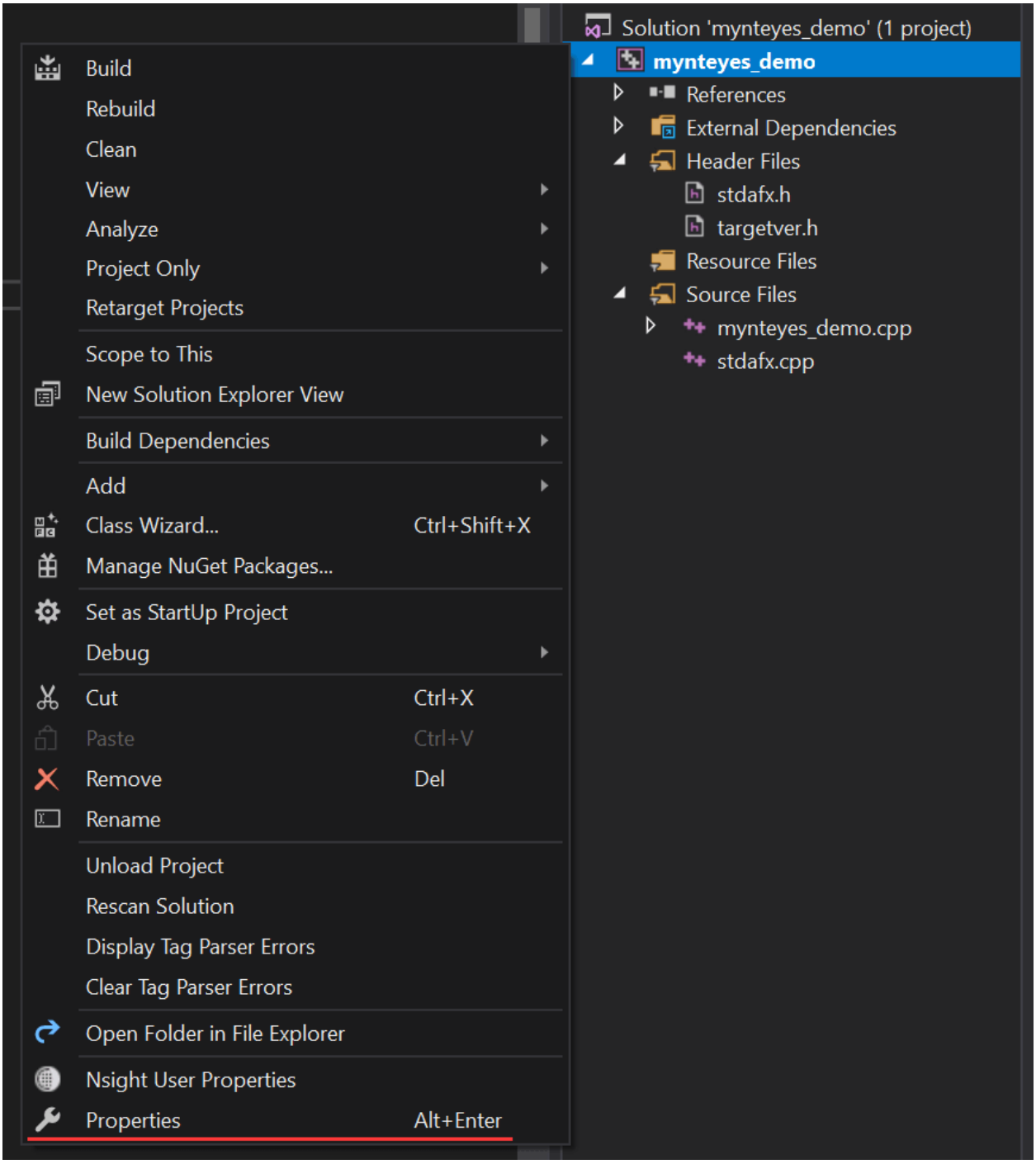


Finally, you will see the new project like this,



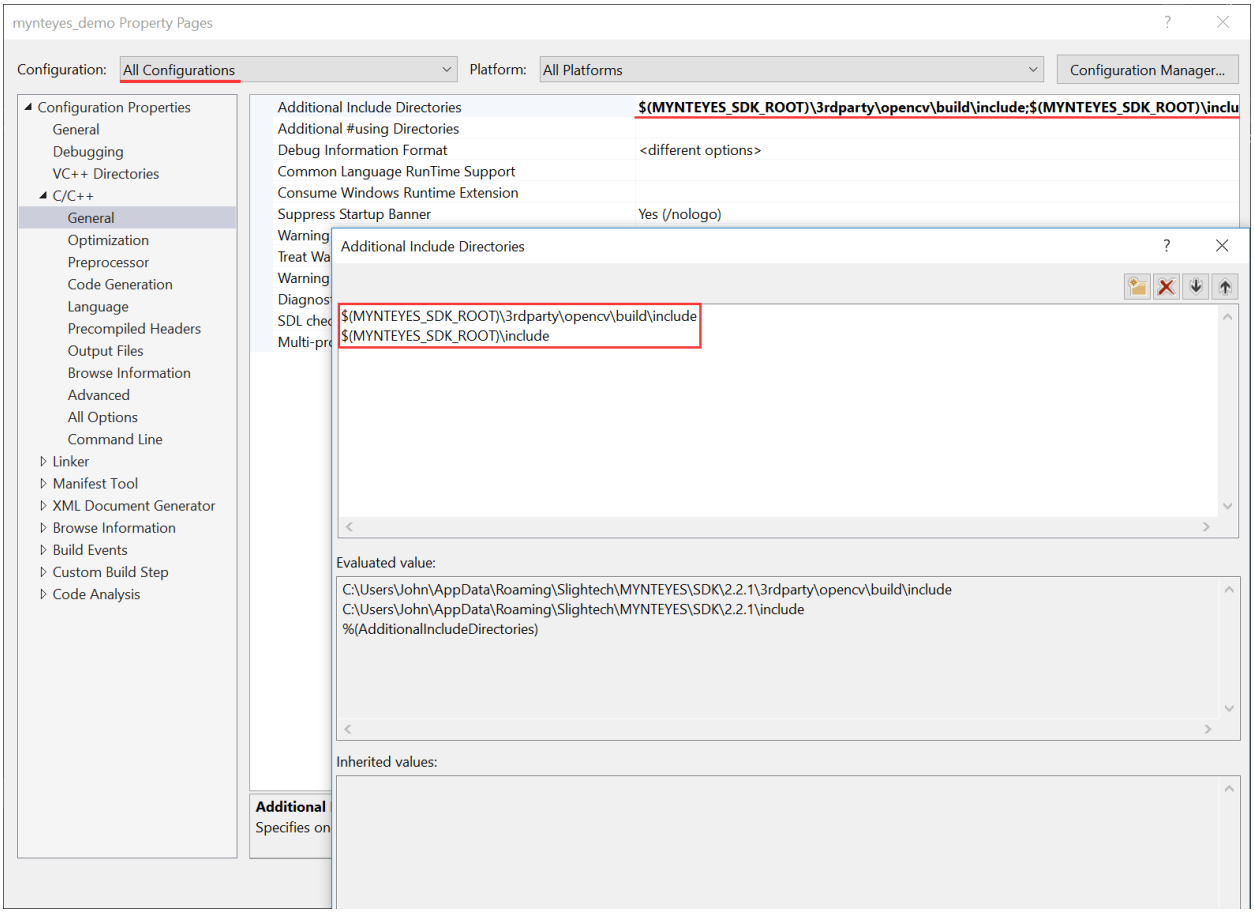
Config Properties

Right click the project, and open its `Properties` window,



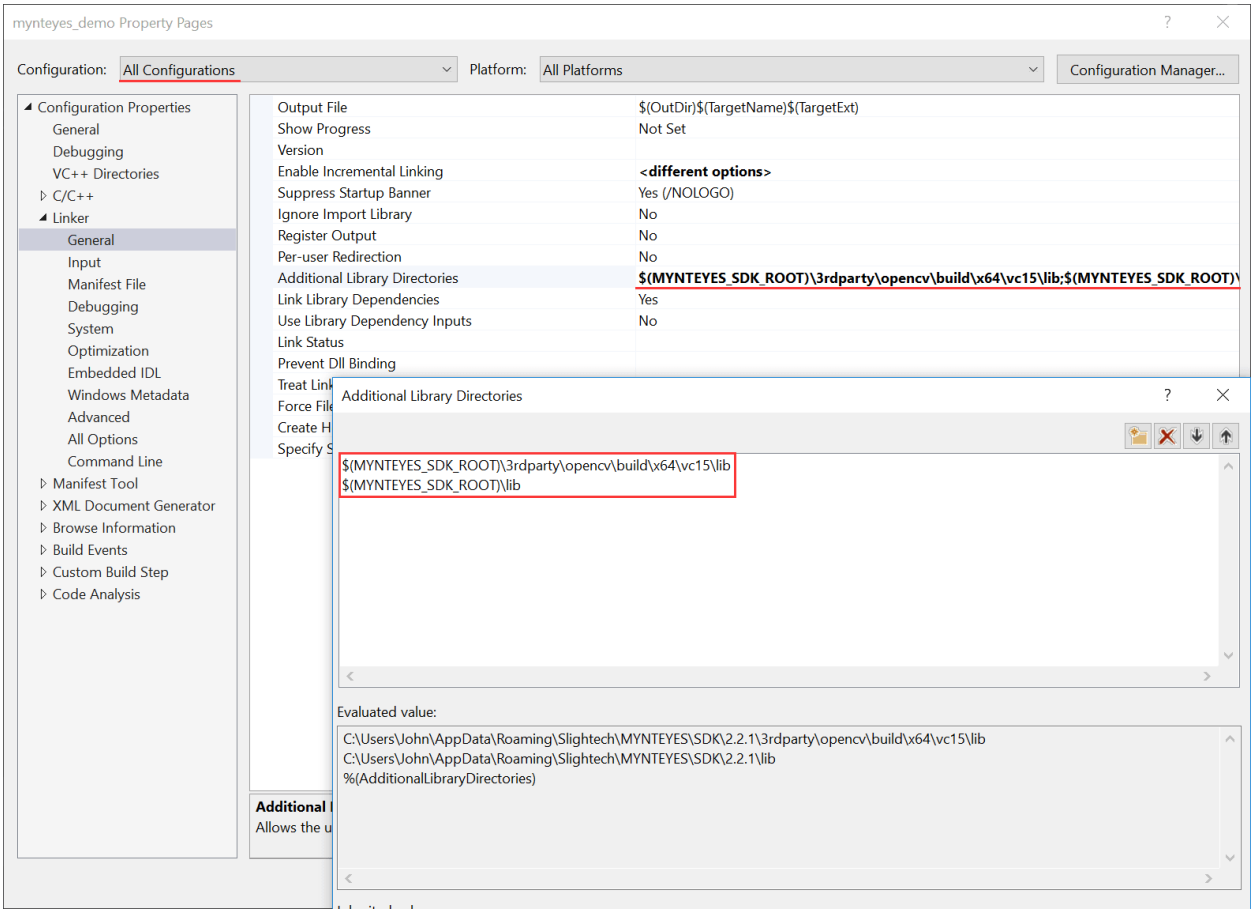
Change `Configuration` to `All Configurations` , then add the following paths to `Additional Include Directories` ,

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



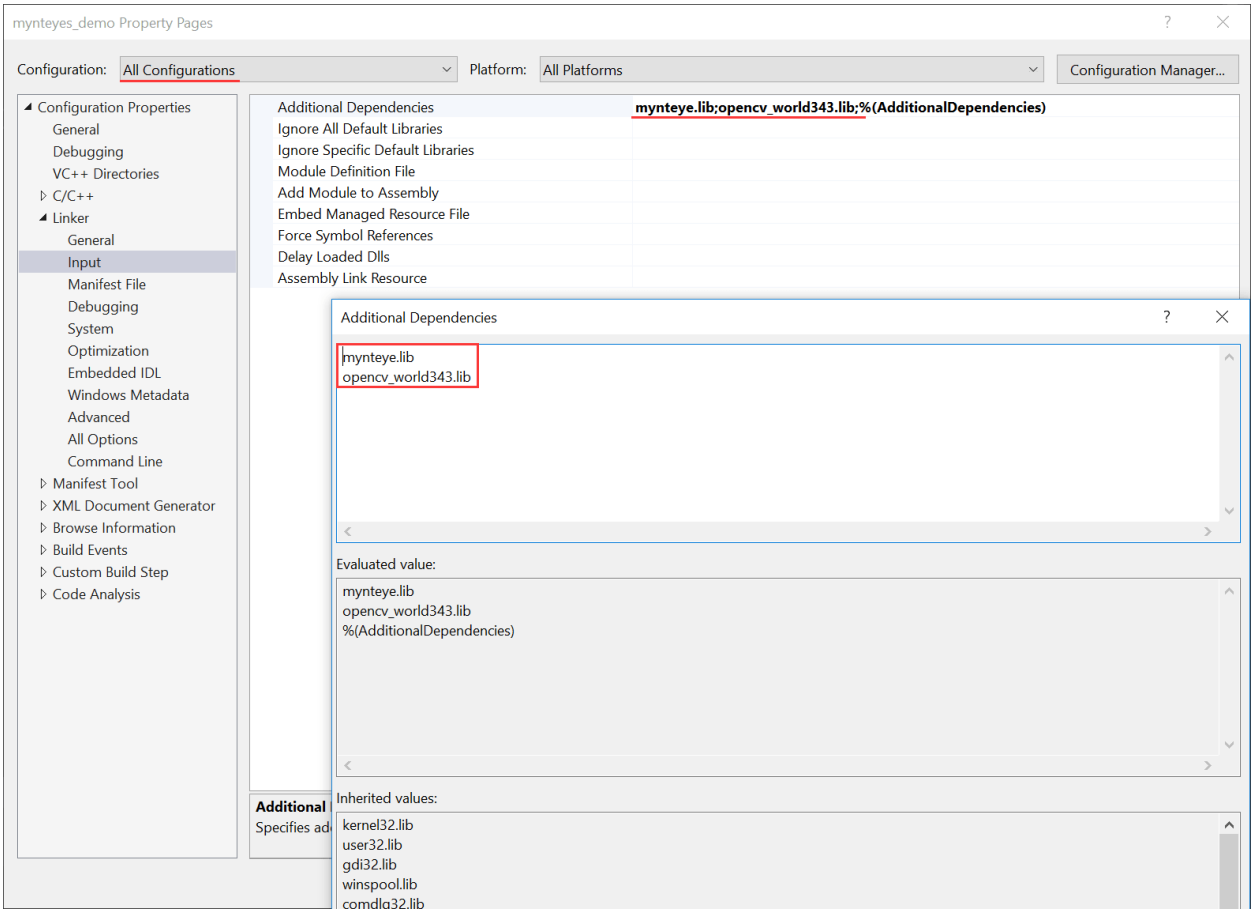
Add the following definitions to `Preprocessor Definitions`,

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



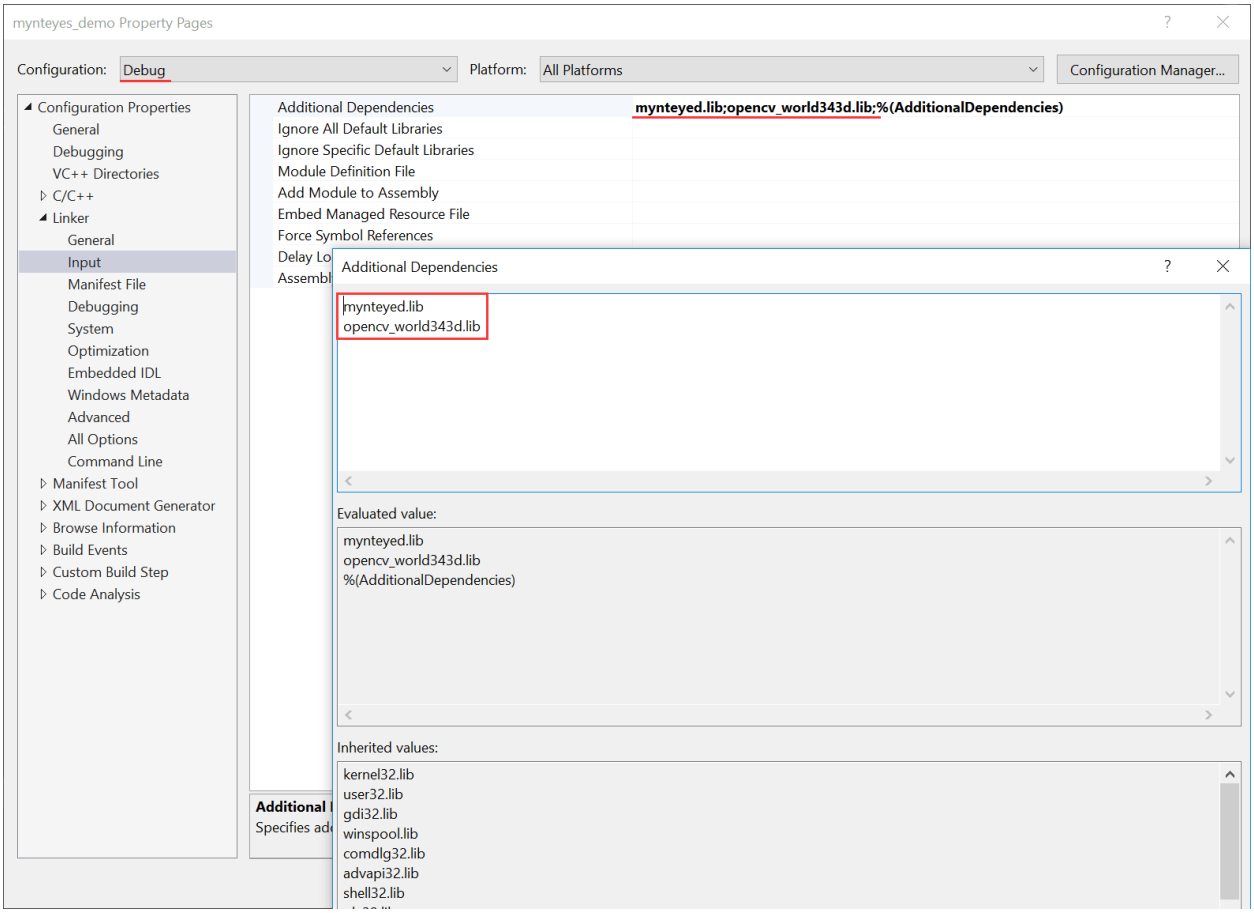
Add the following paths to `Additional Dependencies`

```
mynteye.lib
opencv_world343.lib
```



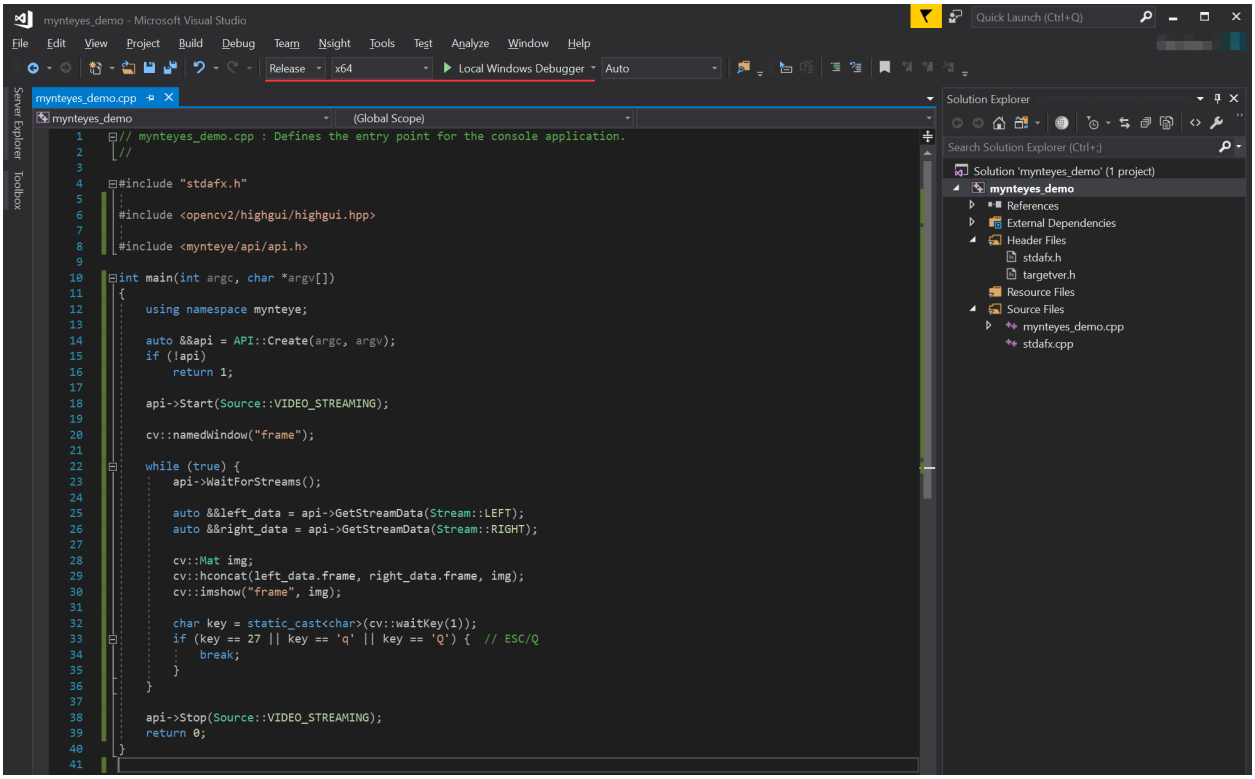
If want to use debug mode, could change `Configuration` to `Debug` and add following debug libs:

```
mynteyed.lib
opencv_world343d.lib
```

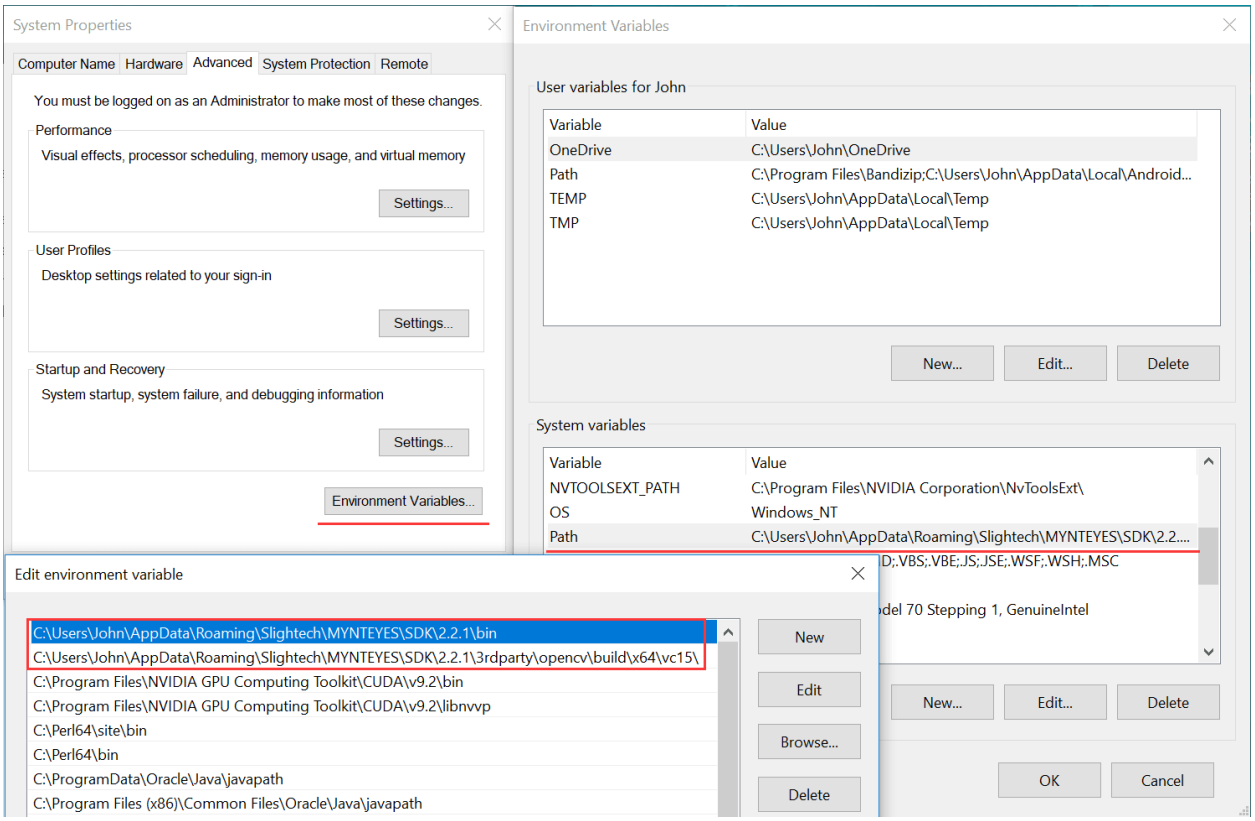


Start using SDK

Include the headers of SDK and start using its APIs,



Select `Release x64` or `Debug x64` to run the project.



2.6 Change Log

2.6.1 2019-10-21(v2.5.0)

1. Change project directory structure.
2. Lifting dependency and improving code quality.
3. Support xy demo code in sample get_depth_with_region.
4. Fix some bugs.

2.6.2 2019-09-4(v2.4.2)

1. Part of the sample examples were sorted out, and useless tools and engineering code were removed.
2. Optimize disparity calculation.
3. Improving the logic of model display.
4. Fix some bugs.

2.6.3 2019-08-17(v2.4.1)

1. Optimize disparity calculation

2.6.4 2019-08-09(v2.4.0)

1. Optimize the synchronization of images and imu
2. Add 4.16+ kernel support on Ubuntu
3. Fix missinf frame_id issue in image information
4. Fix S1030 device not work issue in mynteye_multiple.launch
5. Add save single picture sample save_single_image

2.6.5 2019-07-03(v2.3.9)

1. Fix ros timestamp issue
2. Add calibration tool doc

2.6.6 2019-05-20(v2.3.8)

1. Improve VINS-Fusion supporting
2. Improve VINS-MONO supporting
3. Fix left/right rect image order error

2.6.7 2019-04-19(v2.3.7)

1. Improve VINS-Fusion supporting
2. Improve ORB-SLAM2 supporting

2.6.8 2019-04-15(v2.3.6)

1. Fix imu align bug of ros wrapper
2. Fix 14.04 complie error of ros wrapper
3. Support set iic address for s2100

2.6.9 2019-04-01(v2.3.5)

1. Improve camera info
2. Modify image algorithm parameters by yaml file
3. Add opening multi devices launch file in ROS
4. Add setting IIC address API of S210A
5. Add image/imu flag of external time source
6. Add LaserScan sample for S1030
7. Modify default orientation of point in ROS

2.6.10 2019-03-18(v2.3.4)

1. Add API to get auxiliary chip&ISP's version (Depend on S2100/S210A 1.1 firmware & 1.0 subsidiary chip firmware)
2. Fix point fragment issue in image algorithm
3. Add 376*240 resolution support to S1030 (Depend on 2.4.0 firmware of S1030)
4. Add API to handle imu temperature drift (Depend on imu calibration)
5. Add version check feature
6. Fix depth image crash issue when use CUDA plugin
7. Documents update

CHAPTER

THREE

FIRMWARE

3.1 Firmware Description

3.1.1 Firmware and SDK compatibility

S1030 Firmwares	SDK Version
MYNTEYE-S1030-2.7.0.img	2.5.0 (2.5.0 ~ latest)
MYNTEYE-S1030-2.5.0.img	2.4.0 (2.4.0 ~ 2.5.0)
MYNTEYE_S_2.4.0.img	2.3.4 (2.3.4 ~ 2.3.9)
MYNTEYE_S_2.3.0.img	2.3.0 (2.2.2-rc1 ~ 2.3.3)
MYNTEYE_S_2.2.2.img	2.3.0 (2.2.2-rc1 ~ 2.3.0)
MYNTEYE_S_2.0.0_rc.img	2.0.0-rc (2.0.0-rc ~ 2.0.0-rc2)
MYNTEYE_S_2.0.0_rc2.img	2.0.0-rc2 (2.0.0-rc ~ 2.0.0-rc2)
MYNTEYE_S_2.0.0_rc1.img	2.0.0-rc1
MYNTEYE_S_2.0.0_rc0.img	2.0.0-rc0 (2.0.0-rc1 ~ 2.0.0-alpha1)
MYNTEYE_S_2.0.0_alpha1.1.img	2.0.0-alpha1 (2.0.0-rc1 ~ 2.0.0-alpha1)
MYNTEYE_S_2.0.0_alpha1.img	2.0.0-alpha1 (2.0.0-rc1 ~ 2.0.0-alpha1)
MYNTEYE_S_2.0.0_alpha0.img	2.0.0-alpha0

S21X0 Firmwares	SDK Version
MYNTEYE-S21X0-2.1.0.img	2.5.0 (2.5.0 ~ latest)
MYNTEYE-S21X0-1.4.0.img	2.4.2(2.4.2 ~ 2.5.0)
MYNTEYE-S2100-1.3.2.img	2.4.0(2.4.0 ~ 2.4.1)
MYNTEYE_S2100_1.2.img	2.3.5(2.3.5 ~ 2.3.9)
MYNTEYE_S2100_1.1.img	2.3.4

Attention: Please CONFIRM your device model and use CORRECT firmware.

`Firmwares` indicates the firmware file name. It's in MYNTEYE_BOX(Download Link) in the `Firmwares` directory.

`SDK Version` indicates the version of the SDK that the firmware is adapted to, and the range of available versions are indicated in parentheses.

3.2 Firmware Update

3.2.1 Update Main Chip Firmware

Please use the MYNT EYE TOOL to update main processing chip.

You can download the firmware and MYNT EYE TOOL installation package in the `Firmwares` folder of MYNT-EYE_BOX(Download Link) . The file structure is as follows:

```
Firmwares/
├─Checksum.txt              # File checksum
├─MYNTEYE-S1030-2.7.0.img   # S1030 firmware
├─MYNTEYE-S21x0-2.1.0.img   # S21X0 firmware
├─...
└─mynt-eye-tool-setup.zip   # MYNT EYE TOOL zip
```

The firmware upgrade program currently only supports Windows, so you need to operate under Windows. Proceed as follows:

Download preparation

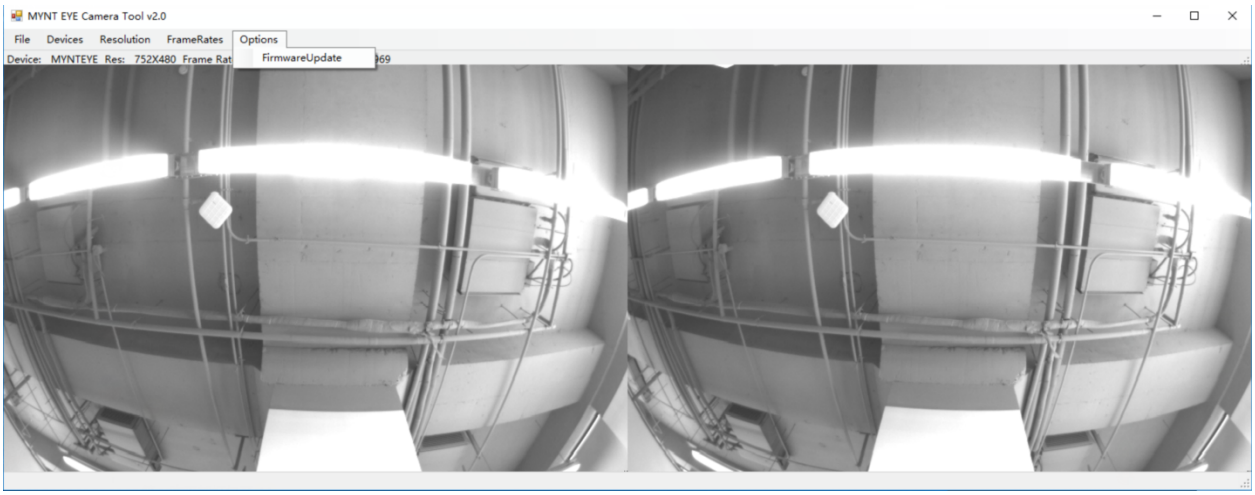
- Download and unzip `mynt-eye-tool-setup.zip`
- Find firmware, such as `MYNTEYE-S1030-2.7.0.img`
  - Please refer to *Firmware and SDK compatibility* to select the firmware suitable for the SDK version

Install MYNT EYE TOOL

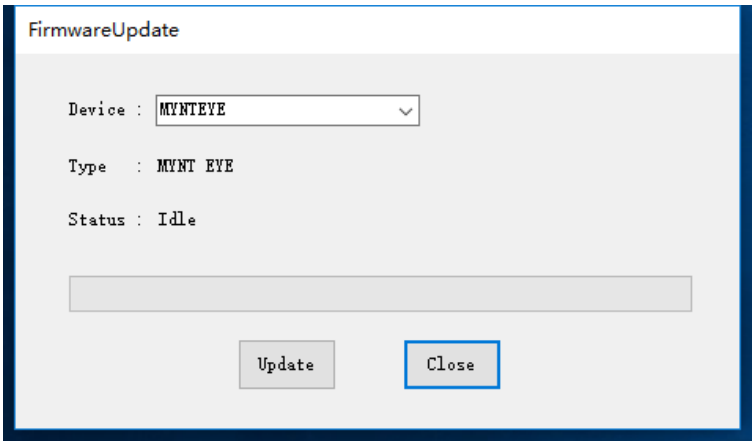
- Double click on `setup.msi` and install the application.

Update Firmware

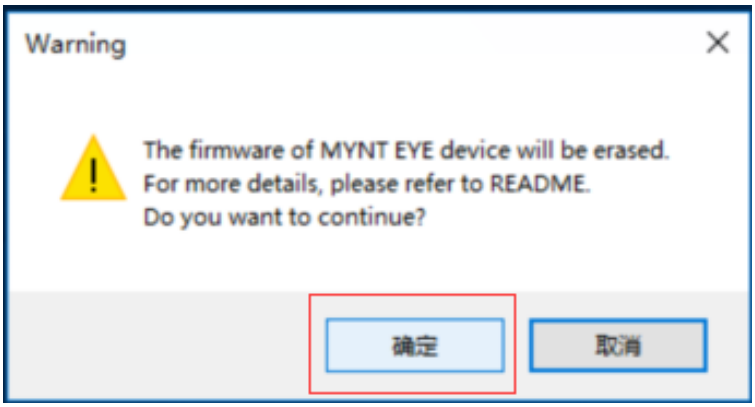
- Plug in the MYNT® EYE camera into a USB3.0 port
- Open MYNT EYE TOOL and select `Options/FirmwareUpdate` .

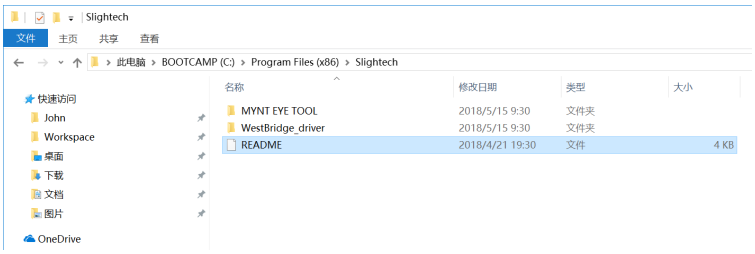


- Click `Update` .

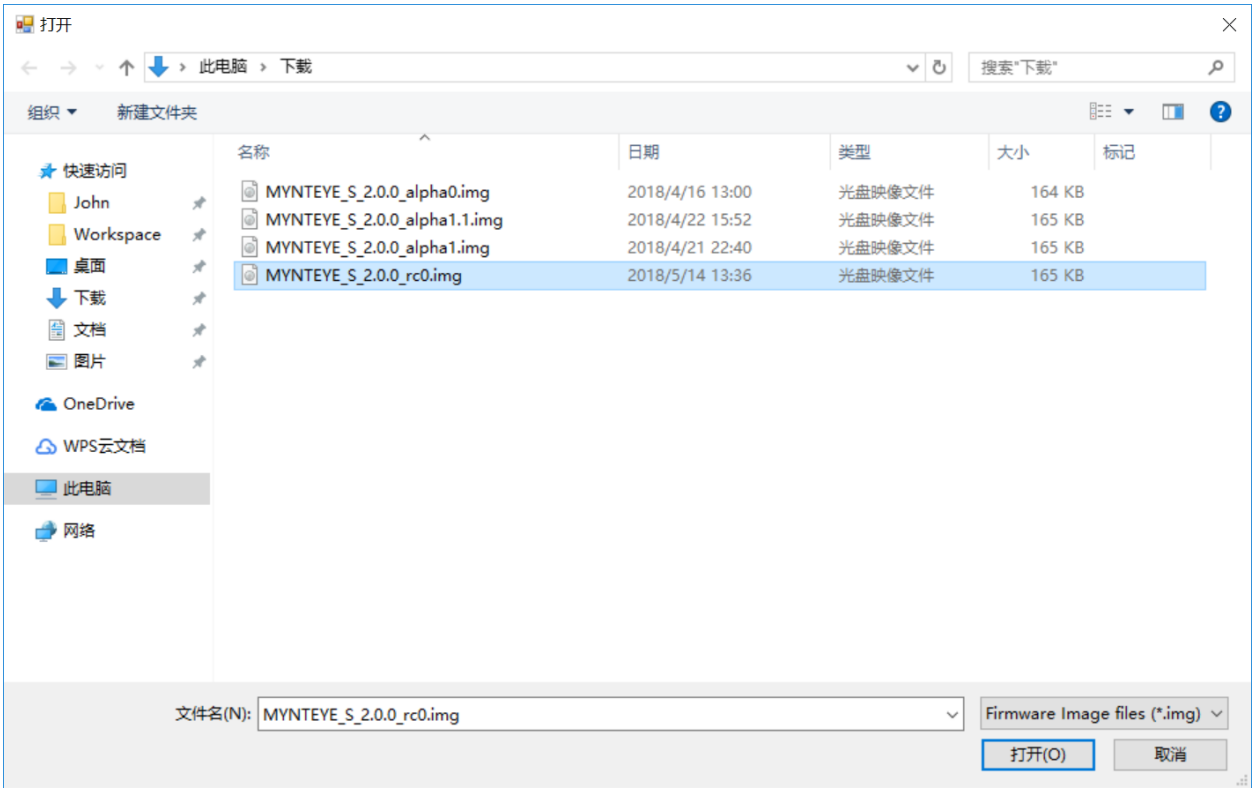


- A warning dialog box will pop up, click `yes` .
  - This operation will erase the firmware, for details see README.
    - Usually, the MYNT EYE TOOL automatically installs the driver during the upgrade process.
    - If the upgrade fails, refer to README.

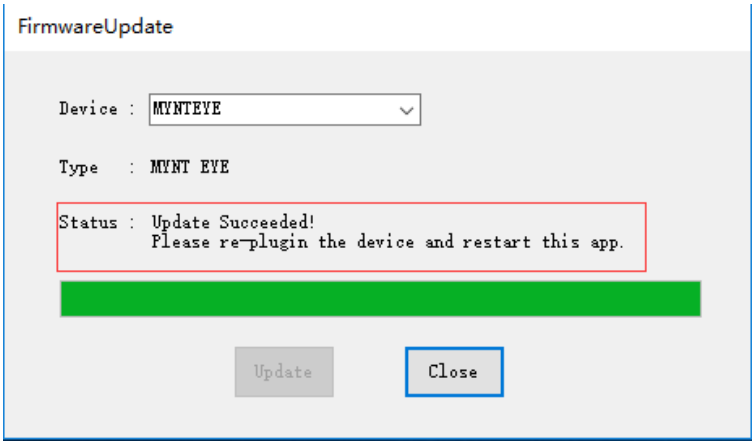




- In the open file selection box, select the firmware you want to upgrade and start upgrading.



- Once the upgrade is complete, the status will changes to `Succeeded`.

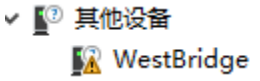


- Close the MYNT EYE TOOLfinish.

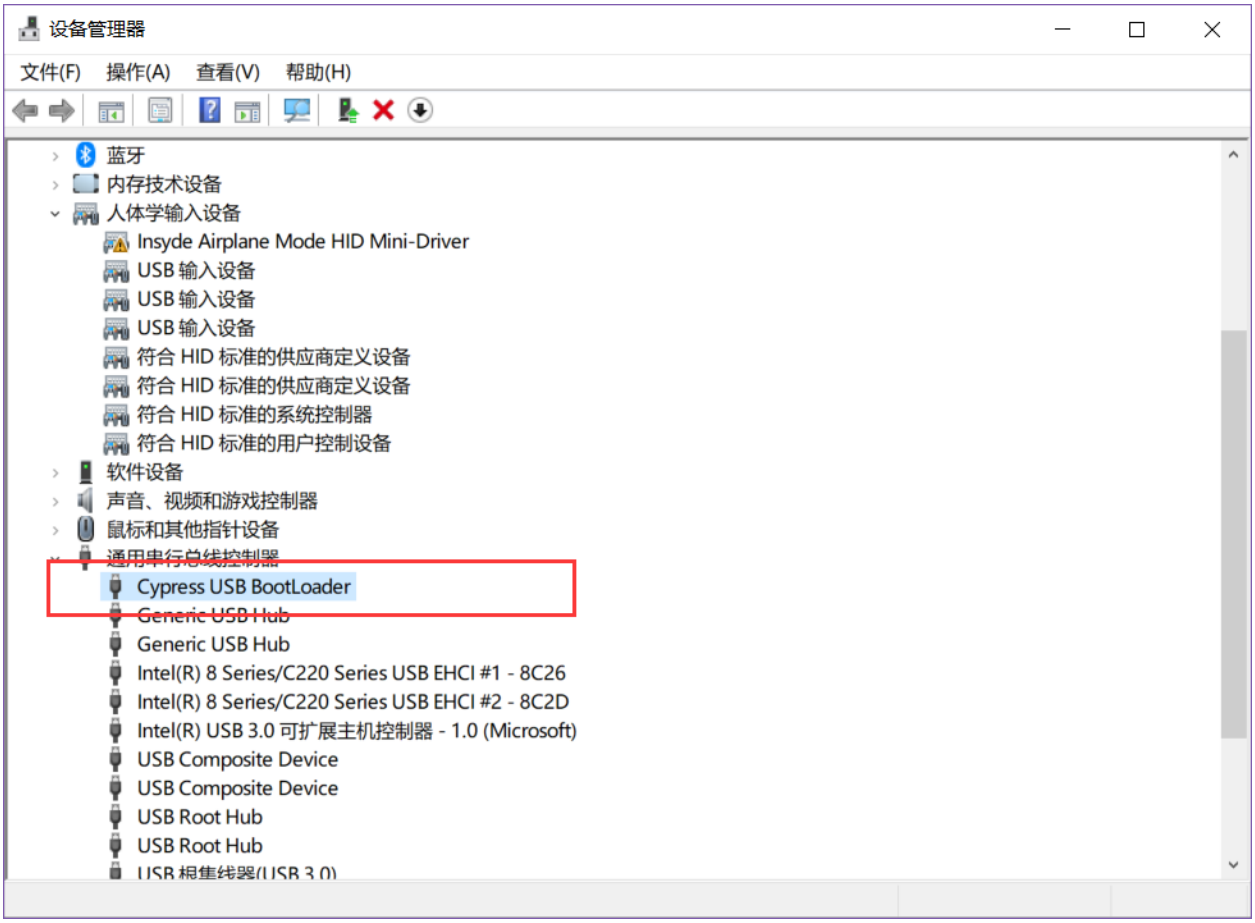
Attention: If you can't find MYNT image device, `WestBridge_driver`, and `Cypress USB BootLoader` at the same time in the device manager, try another computer to perform the above operation. If you can not upgrade successfully, please contact us in time.

Manually update drivers

- If the application indicates that you failed to update, you may fail to install the driver automatically. You can try to install the driver manually and then update it. The following is the manual installation of the driver.
- Open device manager, locate `WestBridge_driver` device, and right click Update Driver,select `[application directory]WestBridge_driver\\[corresponding system folders](If it is more than win7, choose wlh)\\[system bits]` .



- For example,if it is the win10 64 bit system computer,and the application is installed under the default path,you should select `C:\Program Files (x86)\slightech\MYNT EYE TOOL 2.0\WestBridge_driver\wlh\x64`.
- After the installation driver is successful, you can find the `Cypress USB BootLoader` device in the device manager.



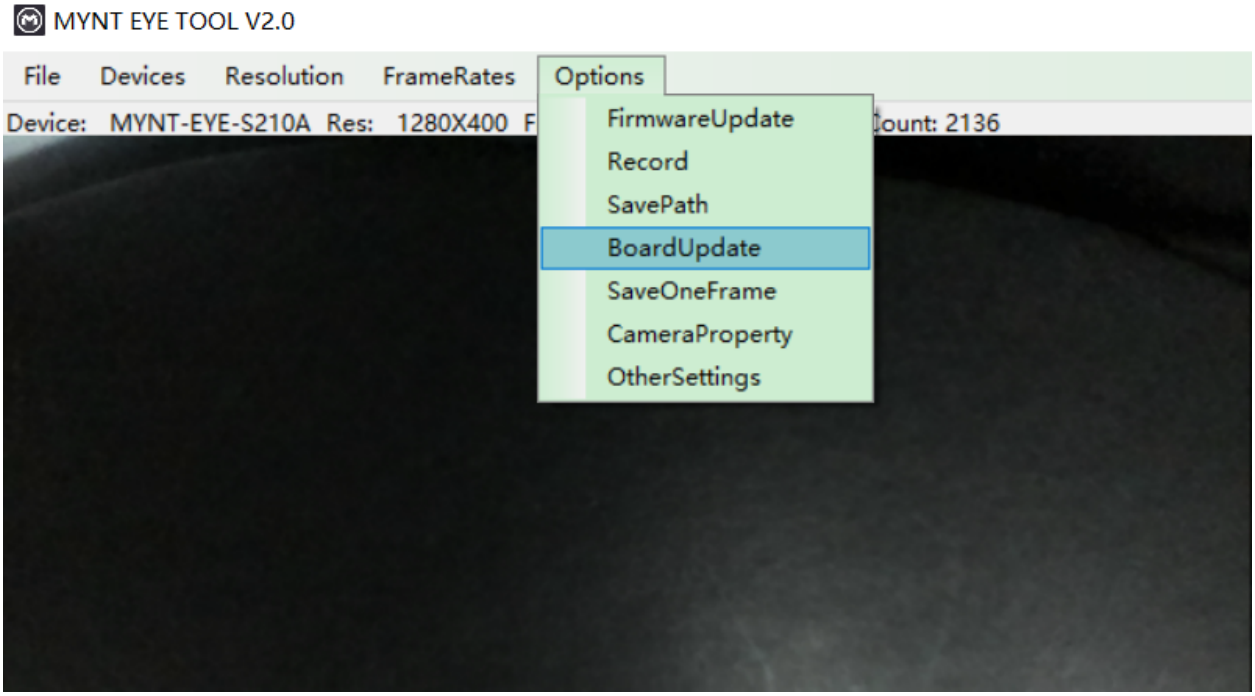
- Then plug in the camera and open the application again to update.

Warning: During the first time you open the MYNT® EYE camera after a firmware update, please hold the camera steadily for 3 seconds, for a zero drift compensation process. You can also call the API `RunOptionAction(Option::ZERO_DRIFT_CALIBRATION)` for zero drift correction.

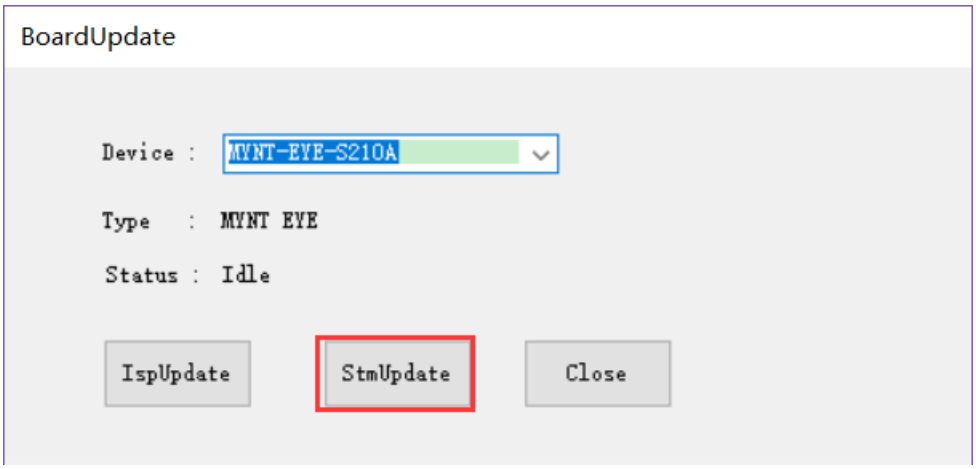
3.2.2 Update Auxiliary Chip Firmware

Update auxiliary chip (Only Support S21XX)

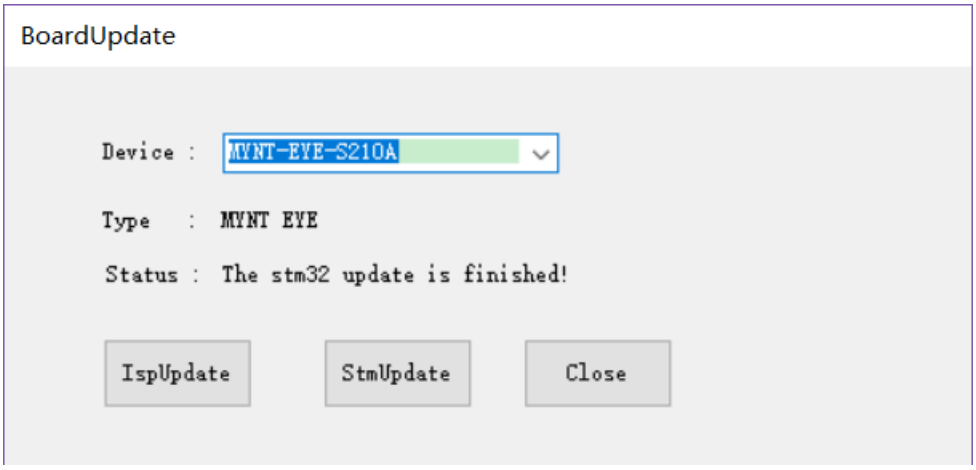
- Plug in the MYNT® EYE camera into a USB3.0 port
- Open MYNT EYE TOOL and select `Options/BoardUpdate` .



- Click `StmUpdate` .



- In the open file selection box, select the firmware `MYNTEYE-S21X0-auxiliary-chip-1.4.2.bin` and start upgrading.
- Once the upgrade is complete, it will display update finished.



3.3 Change Log

3.3.1 2019-11-25

S1030 Main Chip Firmware: MYNTEYE-S1030-2.7.0.img

1. Fix some issues

S21X0 Main Chip Firmware: MYNTEYE-S21X0-2.1.0.img

1. Update protocol version
2. Fix and optimize some other issues

3.3.2 2019-09-09

S21X0 Main Chip Firmware: MYNTEYE-S21X0-1.4.0.img

1. Support 2110 device.

3.3.3 2019-08-09

S1030 Main Chip Firmware: MYNTEYE-S1030-2.5.0.img

1. Optimize the synchronization of image and imu
2. Not save the camera control parameters
3. Fix the overexplosion problem at low resolution
4. Fix USB 2.0 first open failure problem
5. Add automatic recovery function when updating wrong firmware

S2100 Main Chip Firmware: MYNTEYE-S2100-1.3.2.img

1. Optimize the synchronization of image and imu
2. Not save the camera control parameters
3. Optimize IMU low-pass filter default values

4. Optimize the exposure time calculation method, the maximum exposure time is limited to 66.5ms
5. Add automatic recovery function when updating wrong firmware
6. Fix and optimize some other issues

S2100 Auxiliary Chip Firmware: MYNTEYE-S2100-auxiliary-chip-1.4.2.bin

1. Time synchronization adds uart interface, io interruption judgement
2. Time synchronization i2c interface adds whoami, read timestamp and median filter open state interface
3. Fix and optimize some other issues

CHAPTER

FOUR

TOOLS SUPPORT

4.1 Calibration Tool Manual

4.1.1 Introduction

4.1.2 1.1 Support Platform

Currently the calibration tool only supports Ubuntu 16.04 LTS, but support the official, ROS multiple version of OpenCV dependencies.

Platform	Architecture	Different dependence
Ubuntu 16.04 LTS	x64(amd64)	libopencv-dev
Ubuntu 16.04 LTS	x64(amd64)	ros-kinetic-opencv3

4.1.3 1.2 Tools description

Deb/ppa installation package is available on Ubuntu. The architecture, dependencies, and versions will be distinguished from the name:

- mynteye-s-calibrator-opencv-official-1.0.0_amd64.deb
- mynteye-s-calibrator-opencv-ros-kinetic-1.0.0_amd64.deb

Dependency identifier	Dependency package	Detailed description
opencv-official	libopencv-dev	https://packages.ubuntu.com/xenial/libopencv-dev
opencv-ros-kinetic	ros-kinetic-opencv3	http://wiki.ros.org/opencv3

4.1.4 1.3 Deb Toolkit Get

Method of Obtaining	Get address
Baidu Cloud	https://pan.baidu.com/s/19rW0fPKUlQj6eldZpZFoAA Extraction code: a6ps
Google Drive	https://drive.google.com/open?id=1RsV2WEKAsfxbn-Z5nGjk5g3ml1UDEsDc

4.1.5 Installation

4.1.6 2.1 Installation Preparation

- Ubuntu 16.04 LTS environment, x64 architecture
- Deb package for the calibration tool, select OpenCV dependencies as needed (this step is not required for PPA installation)

4.1.7 2.2 Install ppa Package

```
$ sudo add-apt-repository ppa:slightech/mynt-eye-s-sdk
$ sudo apt-get update
$ sudo apt-get install mynteye-s-calibrator
$ sudo ln -sf /opt/myntai/mynteye-s-calibrator/mynteye-s-calibrator /usr/local/bin/␣
↪mynteye-s-calibrator
```

4.1.8 2.3 Install deb Package

Install the deb package with udo dpkg -i:

```
$ sudo dpkg -i mynteye-s-calibrator-opencv-official-1.0.0_amd64.deb
...
(Reading database ... 359020 files and directories currently installed.)
Preparing to unpack mynteye-s-calibrator-opencv-official-1.0.0_amd64.deb ...
Unpacking mynteye-s-calibrator (1.0.0) over (1.0.0) ...
Setting up mynteye-s-calibrator (1.0.0) ...
```

If you encounter an error that the dependency package is not installed, for example:

```
$ sudo dpkg -i mynteye-s-calibrator-opencv-official-1.0.0_amd64.deb
Selecting previously unselected package mynteye-s-calibrator.
(Reading database ... 358987 files and directories currently installed.)
Preparing to unpack mynteye-s-calibrator-opencv-official-1.0.0_amd64.deb ...
Unpacking mynteye-s-calibrator (1.0.0) ...
dpkg: dependency problems prevent configuration of mynteye-s-calibrator:
mynteye-s-calibrator depends on libatlas-base-dev; however:
Package libatlas-base-dev is not installed.
dpkg: error processing package mynteye-s-calibrator (--install):
dependency problems - leaving unconfigured
Errors were encountered while processing:
mynteye-s-calibrator
```

You can continue use sudo apt-get -f install to finished install

```
$ sudo apt-get -f install
Reading package lists... Done
Building dependency tree
Reading state information... Done
Correcting dependencies... Done
The following additional packages will be installed:
```

(continues on next page)

(continued from previous page)

```
libatlas-base-dev
Suggested packages:
libblas-doc liblapack-doc
The following NEW packages will be installed:
libatlas-base-dev
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
1 not fully installed or removed.
Need to get 3,596 kB of archives.
After this operation, 30.8 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://cn.archive.ubuntu.com/ubuntu xenial/universe amd64 libatlas-base-dev amd64␣
↪3.10.2-9 [3,596 kB]
Fetched 3,596 kB in 3s (1,013 kB/s)
Selecting previously unselected package libatlas-base-dev.
(Reading database ... 358993 files and directories currently installed.)
Preparing to unpack .../libatlas-base-dev_3.10.2-9_amd64.deb ...
Unpacking libatlas-base-dev (3.10.2-9) ...
Setting up libatlas-base-dev (3.10.2-9) ...
update-alternatives: using /usr/lib/atlas-base/atlas/libblas.so to provide /usr/lib/
↪libblas.so (libblas.so) in auto mode
update-alternatives: using /usr/lib/atlas-base/atlas/liblapack.so to provide /usr/lib/
↪liblapack.so (liblapack.so) in auto mode
Setting up mynteye-s-calibrator (1.0.0) ...
```

4.1.9 How To Use

4.1.10 3.1 Preparation For Use

- MYNT EYE S Camera
- Checkerboard
- Evenly illuminated scene

4.1.11 3.2 Use Command

- After installing the calibration tool, you can run the *mynteye-s-calibrator* command directly on the terminal to calibrate. -h can see its options:

```
$ mynteye-s-calibrator -h
Usage: mynteye-s-calibrator [options]
help: mynteye-s-calibrator -h
calibrate: mynteye-s-calibrator -x 11 -y 7 -s 0.036
Calibrate MYNT EYE S device.
```

Options:

-h, --help show this help message and exit

-x WIDTH, --width=WIDTH The chessboard width, default: 11

-y HEIGHT, --height=HEIGHT The chessboard height, default: 7

-s METERS, --square=METERS The chessboard square size in meters, default: 0.036

-n NUMBER, --number=NUMBER The number of imagestools to use for calibration, default: 11

-p PATH, --path=PATH The path to save the result, default: folder name using device's SN

- -x -y -s Used to set the width, height, and grid size of the calibration plate. Width and height refer to the number of black and white intersections in the horizontal and vertical directions of the checkerboard. Square size in meters.

4.1.12 3.3 Steps For Usage

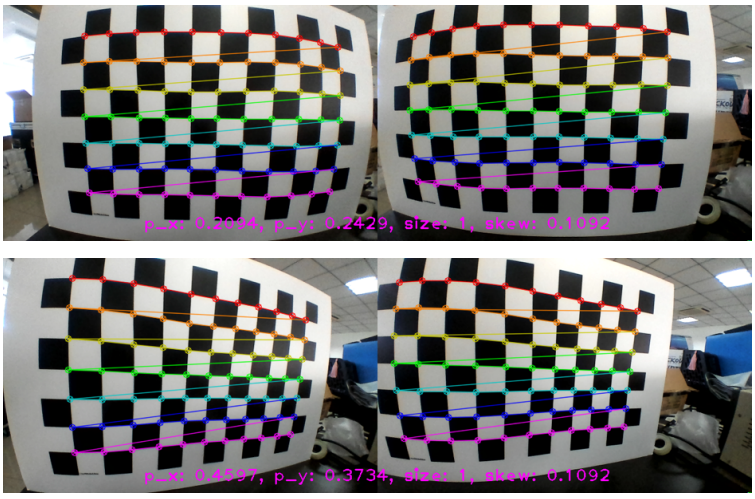
- First, connect the MYNT EYE S camera.
- Then, run the mynteye-s-calibrator <calibration board parameter> command in the terminal.

```
john@john-ubuntu:~$ mynteye-s-calibrator
I/main.cc:68 Calibrate MYNT EYE S device with chessboard 11x7, size: 0.036m, num: 11
I/utils.cc:35 Detecting MYNT EYE devices
I/utils.cc:48 MYNT EYE devices:
I/utils.cc:51   index: 0, name: MYNT-EYE-S2100, sn: 030E1D2C0009072B, firmware: 1.2
I/utils.cc:60 Only one MYNT EYE device, select index: 0
I/synthetic.cc:59 camera calib model: kannala_brandt
E/disparity_processor.cc:51 BM not supported in opencv 2.x, use sgbm
I/utils.cc:93 MYNT EYE requests:
I/utils.cc:96   index: 0, request: width: 1280, height: 400, format: Format::YUYV, fps: 10
I/utils.cc:96   index: 1, request: width: 1280, height: 400, format: Format::YUYV, fps: 20
I/utils.cc:96   index: 2, request: width: 1280, height: 400, format: Format::YUYV, fps: 30
I/utils.cc:96   index: 3, request: width: 1280, height: 400, format: Format::YUYV, fps: 60
I/utils.cc:96   index: 4, request: width: 2560, height: 800, format: Format::YUYV, fps: 10
I/utils.cc:96   index: 5, request: width: 2560, height: 800, format: Format::YUYV, fps: 20
I/utils.cc:96   index: 6, request: width: 2560, height: 800, format: Format::YUYV, fps: 30
I/utils.cc:107 There are 7 stream requests, select index:
2
I/calibrator.cc:76 Had selected 1 imgs
I/calibrator.cc:76 Had selected 2 imgs
```

- Follow the prompts to select an index for the camera's resolution, perform image calibration at this resolution
- The S1030 camera only need calibrate 752*480 resolution. The S2100 camera need calibrate 2560*800 and 1280*400 resolutions.
- As far as possible, let the calibration plate cover the left and right eye images of the camera,

and take care of the surroundings (maximum distortion). The calibration tool will automatically evaluate the qualified image for the calibration calculation and will indicate on the terminal how many have been selected.

Reference acquisition image, as follows:

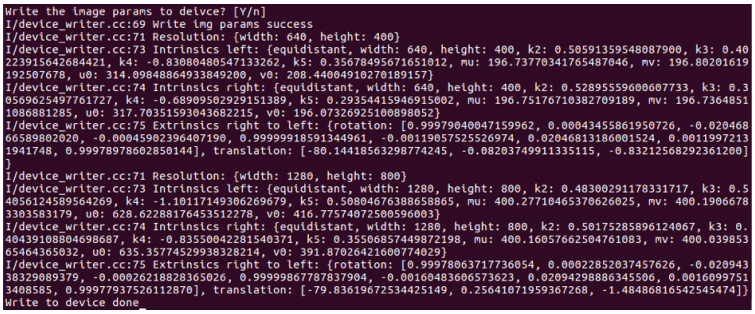


- Note: p_x, p_y, size, skew respectively indicate the scale of the calibration plate on the x-axis, y-axis, zoom, and tilt when the image is acquired. Make a point for reference.
- Once the number of images acquired by the calibration needs is reached, the calibration calculation will be performed. The output is as follows:

```
[stereo] INFO: Final extrinsics:
r: 0.001  p: -0.020  yaw: -0.000
x: -0.080  y: -0.000  z: -0.001
[left] INFO: Final reprojection error: 0.201 pixels
[left] INFO: Camera Parameters:
    model_type KANNALA_BRANDT
   camera_name left
   image_width 640
  image_height 400
Projection Parameters
            k2 0.50591359548087900
            k3 0.40223915642684421
            k4 -0.83080480547133262
            k5 0.35678495671651012
            mu 196.73770341765487046
            mv 196.80201619192507678
            u0 314.09848864933849200
            v0 208.44004910270189157

[right] INFO: Final reprojection error: 0.199 pixels
[right] INFO: Camera Parameters:
    model_type KANNALA_BRANDT
   camera_name right
   image_width 640
  image_height 400
Projection Parameters
            k2 0.52895559600607733
            k3 0.30569625497761727
            k4 -0.68909502929151389
            k5 0.29354415946915002
            mu 196.75176710382709189
            mv 196.73648511086881285
            u0 317.70351593043682215
            v0 196.07326925100898052

I/calibrator.cc:91 Complete rectify
I/calibrator.cc:93 Calibration took a total time of 0.483 sec
I/calibrator.cc:96 Wrote calibration files to SN030E1D2C0009072B-190603101122

Write the image params to deivce? [Y/n] 
```

- 1. The terminal will print out the left and right purpose calibration results.

- 2. The calibration results will be written into the files in <SN number> directory.
  a) camera_left.yaml: Left eye parameter
  b) camera_right.yaml: Right eye parameter
  c) extrinsics.yaml: Binocular external parameter
  d) img.params.equidistant: Camera parameters, which can be used for S SDK writing
  e) stereo_reprojection_error.yaml: Reprojection error
- Finally, you will also be asked if you want to write to the camera device. Enter or *y* to confirm



- After writing to the device, you will be prompted with "Write to device done".

4.1.13 3.4 Calibration result

Calibration result, It is desirable to have a reprojection error of 0.2 or less. If exceeds 1, it needs to be recalibrated.

Reprojection error, visible output after calibration completion "Final reprojection error: 0.201 Pixels", or see the calibration result file "stereo_reprojection_error.yaml".

CHAPTER

FIVE

OPEN SOURCE SUPPORT

5.1 How To Use In VINS-Mono

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

1. Download MYNT-EYE-S-SDK and install mynt_eye_ros_wrapper.
2. Follow the normal procedure to install VINS-Mono.
3. Run mynt_eye_ros_wrapper and VINS-Mono.

5.1.2 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
```

5.1.3 Install Docker

```
sudo apt-get update
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Then add your account to `docker` group by `sudo usermod -aG docker $YOUR_USER_NAME` . Relaunch the terminal or logout and re-login if you get `Permission denied` error.

To complie with docker,we recommend that you should use more than 16G RAM, or ensure that the RAM and virtual memory space is greater than 16G.

5.1.4 Install MYNT-EYE-VINS-Sample

```
git clone -b docker_feat https://github.com/slightech/MYNT-EYE-VINS-Sample.git
cd MYNT-EYE-VINS-Sample/docker
make build
```

Note that the docker building process may take a while depends on your network and machine. After VINS-Mono successfully started, open another terminal and play your bag file, then you should be able to see the result. If you need modify the code, simply run `./run.sh LAUNCH_FILE_NAME` after your changes.

5.1.5 Run VINS-Mono with MYNT® EYE

1. Launch mynteye node

```
cd (local path of MYNT-EYE-S-SDK)
source ./wrappers/ros/devel/setup.bash
roslaunch mynt_eye_ros_wrapper vins_mono.launch
```

2. Open another terminal and run vins

```
cd path/to/MYNT-EYE-VINS-Sample/docker
./run.sh mynteye_s.launch
# ./run.sh mynteye_s2100.launch  # mono with s2100
```

5.2 How To Use In VINS-Fusion

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

1. Download MYNT-EYE-S-SDK and install mynt_eye_ros_wrapper.
2. Follow the normal procedure to install VINS-Fusion.
3. Run mynt_eye_ros_wrapper and VINS-Fusion.

5.2.2 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
```

5.2.3 Install Docker

```
sudo apt-get update
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Then add your account to `docker` group by `sudo usermod -aG docker $YOUR_USER_NAME` . Relaunch the terminal or logout and re-login if you get `Permission denied` error.

To complie with docker,we recommend that you should use more than 16G RAM, or ensure that the RAM and virtual memory space is greater than 16G.

5.2.4 Install MYNT-EYE-VINS-FUSION-Samples

```
git clone https://github.com/slightech/MYNT-EYE-VINS-FUSION-Samples.git
cd MYNT-EYE-VINS-FUSION-Samples/docker
make build
```

Note that the docker building process may take a while depends on your network and machine. After VINS-Mono successfully started, open another terminal and play your bag file, then you should be able to see the result. If you need modify the code, simply run `./run.sh LAUNCH_FILE_NAME` after your changes.

5.2.5 Run VINS-FUSION with MYNT® EYE

1. Launch mynteye node

```
cd (local path of MYNT-EYE-S-SDK)
source ./wrappers/ros/devel/setup.bash
roslaunch mynt_eye_ros_wrapper vins_fusion.launch
```

2. Open another terminal and run vins

```
cd path/to/MYNT-EYE-VINS-FUSION-Samples/docker
./run.sh mynteye-s/mynt_stereo_imu_config.yaml  # Stereo fusion
# ./run.sh mynteye-s2100/mynt_stereo_config.yaml # Stereo fusion with mynteye-s2100
# ./run.sh mynteye-s2100/mynt_stereo_imu_config.yaml # Stereo+imu fusion with mynteye-
↪s2100
```

5.3 How To Use In ORB_SLAM2

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

1. Download MYNT-EYE-S-SDK and follow steps to install.
2. Follow the normal procedure to install ORB_SLAM2.
3. Run examples by MYNT® EYE.

5.3.2 Prerequisites

```
sudo apt-get -y install libglew-dev cmake libgoogle-glog-dev
cd ~
git clone https://github.com/stevenlovegrove/Pangolin.git
cd Pangolin
mkdir build
cd build
cmake ..
cmake --build .
sudo make install
```

5.3.3 Building the nodes for mono and 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.

```
export ROS_PACKAGE_PATH=${ROS_PACKAGE_PATH}:~/catkin_ws/src/MYNT-EYE-ORB-SLAM2-Sample
```

- Execute *build_ros.sh*:

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

Stereo_ROS Example

- Launch ORB_SLAM2 `Stereo_ROS`

1. Launch mynteye node

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

2. Open another terminal and run ORB_SLAM2

```
rosrun ORB_SLAM2 mynteye_s_stereo ./Vocabulary/ORBvoc.txt ./config/mynteye_s_stereo.yaml␣
↪false /mynteye/left_rect/image_rect /mynteye/right_rect/image_rect
```

5.4 How To Use In OKVIS

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

1. Download MYNT-EYE-S-SDK and install it.
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.

Tip: OKVIS doesn't support ARM right now

5.4.2 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
```

5.4.3 Get camera calibration parameters

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

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

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

Tip: You can get the camera model of device when get camera calibration parameters, if model is equidistant you need calibrate pinhole model by yourself or reference write_img_params to write a default pinhole config file to your device.

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

(continues on next page)

(continued from previous page)

```
principal_point: [cx, cy]
distortion_type: radialtangential
```

5.4.4 Run MYNTEYE OKVIS

Go to `MYNT-EYE-OKVIS-Sample/build` folder and Run the application `okvis_app_mynteye_s` :

```
cd MYNT-EYE-OKVIS-Sample/build
./okvis_app_mynteye_s ../config/config_mynteye_s.yaml
```

5.5 How To Use In VIORB

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

1. Download MYNT-EYE-S-SDK and install mynt_eye_ros_wrapper.
2. Follow the normal procedure to install VIORB.
3. Update camera parameters to <VIO>`/config/mynteye_s.yaml`.
4. Run mynt_eye_ros_wrapper and VIORB.

5.5.2 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
```

5.5.3 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-S-SDK API, you can get the image calibration parameters of the currently open device:

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

Tip: You can get the camera model of device when get camera calibration parameters, if model is equidistant you need calibrate pinhole model by yourself or reference write_img_params to write a default pinhole config file to your device.

5.5.4 Run VIORB and mynt_eye_ros_wrapper

1. Launch mynteye node

```
roslaunch mynt_eye_ros_wrapper mynteye.launch
```

2. Open another terminal and run viorb

```
roslaunch ORB_VIO testmynteye_s.launch
```

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

5.6 How To Use In Maplab x

CHAPTER

SIX

API DOCS

6.1 API

6.1.1 API

class **API**

The *API* class to communicate with MYNT® EYE device.

Public Types

using **stream_callback_t** = std::function<void(const api::*StreamData* &data)>

The *api::StreamData* callback.

using **motion_callback_t** = std::function<void(const api::*MotionData* &data)>

The *api::MotionData* callback.

using **stream_switch_callback_t** = std::function<void(const *Stream* &stream)>

The enable/disable switch callback.

Public Functions

Model **GetModel**() const

Get the model.

bool **Supports**(const *Stream* &stream) const

Supports the stream or not.

bool **Supports**(const *Capabilities* &capability) const

Supports the capability or not.

bool **Supports**(const *Option* &option) const

Supports the option or not.

bool **Supports**(const *AddOns* &addon) const

Supports the addon or not.

StreamRequest **SelectStreamRequest**(bool *ok) const

Log all stream requests and prompt user to select one.

const std::vector<*StreamRequest*> &**GetStreamRequests**(const *Capabilities* &capability) const

Get all stream requests of the capability.

void **ConfigStreamRequest**(const *Capabilities* &capability, const *StreamRequest* &request)

Config the stream request to the capability.

const *StreamRequest* &**GetStreamRequest**(const *Capabilities* &capability) const

Get the config stream requests of the capability.

const std::vector<*StreamRequest*> &**GetStreamRequests**() const

Get all stream requests of the key stream capability.

void **ConfigStreamRequest**(const *StreamRequest* &request)

Config the stream request to the key stream capability.

const *StreamRequest* &**GetStreamRequest**() const

Get the config stream requests of the key stream capability.

std::shared_ptr<DeviceInfo> **GetInfo**() const

Get the device info.

std::string **GetInfo**(const *Info* &info) const

Get the device info.

std::string **GetSDKVersion**() const

Get the sdk version.

IntrinsicsPinhole **GetIntrinsics**(const *Stream* &stream) const

Deprecated:

Get the intrinsics (pinhole) of stream.

template<typename **T**>
T **GetIntrinsics**(const *Stream* &stream) const

Get the intrinsics of stream.

std::shared_ptr<IntrinsicsBase> **GetIntrinsicsBase**(const *Stream* &stream) const

Get the intrinsics base of stream.

Extrinsics **GetExtrinsics**(const *Stream* &from, const *Stream* &to) const

Get the extrinsics from one stream to another.

MotionIntrinsics **GetMotionIntrinsics**() const

Get the intrinsics of motion.

Extrinsics **GetMotionExtrinsics**(const *Stream* &from) const

Get the extrinsics from one stream to motion.

void **LogOptionInfos**() const

Log all option infos.

OptionInfo **GetOptionInfo**(const *Option* &option) const

Get the option info.

std::int32_t **GetOptionValue**(const *Option* &option) const

Get the option value.

void **SetDisparityComputingMethodType**(const *DisparityComputingMethod* &MethodType)

Set the disparity computing method.

void **SetRectifyAlpha**(const double &alpha)

Set the rectify bord cut alpha.

void **setDuplicate**(bool isEnable)

Set if the duplicate frames is enable.

void **SetOptionValue**(const *Option* &option, std::int32_t value)

Set the option value.

bool **SetOptionValue**(const *Option* &option, std::uint64_t value)

Set the option value.

bool **RunOptionAction**(const *Option* &option) const

Run the option action.

void **SetStreamCallback**(const *Stream* &stream, *stream_callback_t* callback)

Set the callback of stream.

void **SetMotionCallback**(*motion_callback_t* callback)

Set the callback of motion.

bool **HasStreamCallback**(const *Stream* &stream) const

Has the callback of stream.

bool **HasMotionCallback**() const

Has the callback of motion.

void **Start**(const *Source* &source)

Start capturing the source.

void **Stop**(const *Source* &source)

Stop capturing the source.

void **WaitForStreams**()

Wait the streams are ready.

void **EnableStreamData**(const *Stream* &stream)

Enable the data of stream.

Note: must enable the stream if it's a synthetic one. This means the stream in not native, the device has the capability to provide this stream, but still support this stream.

---

void **EnableStreamData**(const *Stream* &stream, *stream_switch_callback_t* callback, bool try_tag = false)

Enable the data of stream.

callback function will call before the father processor enable. when try_tag is true, the function will do nothing except callback.

void **DisableStreamData**(const *Stream* &stream)

Disable the data of stream.

void **DisableStreamData**(const *Stream* &stream, *stream_switch_callback_t* callback, bool try_tag = false)

Disable the data of stream.

callback function will call before the children processor disable. when try_tag is true, the function will do nothing except callback.

api::*StreamData* **GetStreamData**(const *Stream* &stream)

Get the latest data of stream.

std::vector<api::*StreamData*> **GetStreamDatas**(const *Stream* &stream)

Get the datas of stream.

Note: default cache 4 datas at most.

void **EnableMotionDatas**(std::size_t max_size = std::numeric_limits<std::size_t>::max())

Enable cache motion datas.

std::vector<api::*MotionData*> **GetMotionDatas**()

Get the motion datas.

void **EnableImuTimestampCorrespondence**(bool is_enable)

enable motion datas timestamp correspondence in device.

void **EnableTimestampCorrespondence**(const *Stream* &stream, bool keep_accel_then_gyro = true)

Enable motion datas with timestamp correspondence of some stream in api.

void **EnablePlugin**(const std::string &path)

Enable the plugin.

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

std::shared_ptr<struct CameraROSMsgInfoPair> **GetCameraROSMsgInfoPair**()

Get ROS need camera info struct.

bool **ConfigDisparityFromFile**(const std::string &config_file)

Load disparity config from file.

Public Static Functions

static std::shared_ptr<*API*> **Create**(int argc, char *argv[])

Create the *API* instance.

Note: This will init glog with args and call *device::select()* to select a device.

Parameters

- **argc** – the arg count.

- **argv** – the arg values.

Returns
the *API* instance.

static std::shared_ptr<*API*> **Create**(int argc, char *argv[], const std::shared_ptr<*Device*> &device)

Create the *API* instance.

Note: This will init glog with args.

Parameters

- **argc** – the arg count.
- **argv** – the arg values.
- **device** – the selected device.

Returns
the *API* instance.

static std::shared_ptr<*API*> **Create**(const std::shared_ptr<*Device*> &device)

Create the *API* instance.

Parameters
device – the selected device.

Returns
the *API* instance.

6.1.2 api::StreamData

struct **StreamData**

API stream data.

Public Members

std::shared_ptr<*ImgData*> **img**

ImgData.

cv::Mat **frame**

Frame.

std::shared_ptr<device::*Frame*> **frame_raw**

Raw frame.

std::uint16_t **frame_id**

Frame ID.

6.1.3 api::MotionData

struct **MotionData**

API motion data.

Public Members

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

ImuData.

6.2 Device

6.2.1 Device

class **Device**

The *Device* class to communicate with MYNT® EYE device.

Public Types

using **stream_callback_t** = device::StreamCallback

The *device::StreamData* callback.

using **motion_callback_t** = device::MotionCallback

The *device::MotionData* callback.

Public Functions

inline *Model* **GetModel**() const

Get the model.

bool **Supports**(const *Stream* &stream) const

Supports the stream or not.

bool **Supports**(const *Capabilities* &capability) const

Supports the capability or not.

bool **Supports**(const *Option* &option) const

Supports the option or not.

bool **Supports**(const *AddOns* &addon) const

Supports the addon or not.

const std::vector<*StreamRequest*> &**GetStreamRequests**(const *Capabilities* &capability) const

Get all stream requests of the capability.

void **ConfigStreamRequest**(const *Capabilities* &capability, const *StreamRequest* &request)

Config the stream request to the capability.

const *StreamRequest* &**GetStreamRequest**(const *Capabilities* &capability) const

Get the config stream requests of the capability.

const std::vector<*StreamRequest*> &**GetStreamRequests**() const

Get all stream requests of the key stream capability.

void **ConfigStreamRequest**(const *StreamRequest* &request)

Config the stream request to the key stream capability.

const *StreamRequest* &**GetStreamRequest**() const

Get the config stream requests of the key stream capability.

std::shared_ptr<DeviceInfo> **GetInfo**() const

Get the device info.

std::string **GetInfo**(const *Info* &info) const

Get the device info of a field.

std::shared_ptr<IntrinsicsBase> **GetIntrinsics**(const *Stream* &stream) const

Get the intrinsics of stream.

Extrinsics **GetExtrinsics**(const *Stream* &from, const *Stream* &to) const

Get the extrinsics from one stream to another.

MotionIntrinsics **GetMotionIntrinsics**() const

Get the intrinsics of motion.

Extrinsics **GetMotionExtrinsics**(const *Stream* &from) const

Get the extrinsics from one stream to motion.

std::shared_ptr<IntrinsicsBase> **GetIntrinsics**(const *Stream* &stream, bool *ok) const

Get the intrinsics of stream.

Extrinsics **GetExtrinsics**(const *Stream* &from, const *Stream* &to, bool *ok) const

Get the extrinsics from one stream to another.

MotionIntrinsics **GetMotionIntrinsics**(bool *ok) const

Get the intrinsics of motion.

Extrinsics **GetMotionExtrinsics**(const *Stream* &from, bool *ok) const

Get the extrinsics from one stream to motion.

void **SetIntrinsics**(const *Stream* &stream, const std::shared_ptr<IntrinsicsBase> &in)

Set the intrinsics of stream.

void **SetExtrinsics**(const *Stream* &from, const *Stream* &to, const *Extrinsics* &ex)

Set the extrinsics from one stream to another.

void **SetMotionIntrinsics**(const *MotionIntrinsics* &in)

Set the intrinsics of motion.

void **SetMotionExtrinsics**(const *Stream* &from, const *Extrinsics* &ex)

Set the extrinsics from one stream to motion.

void **LogOptionInfos**() const

Log all option infos.

OptionInfo **GetOptionInfo**(const *Option* &option) const

Get the option info.

std::int32_t **GetOptionValue**(const *Option* &option) const

Get the option value.

void **SetOptionValue**(const *Option* &option, std::int32_t value)

Set the option value.

bool **SetOptionValue**(const *Option* &option, std::uint64_t value)

Set the option value.

bool **RunOptionAction**(const *Option* &option) const

Run the option action.

void **SetStreamCallback**(const *Stream* &stream, *stream_callback_t* callback, bool async = false)

Set the callback of stream.

void **SetMotionCallback**(*motion_callback_t* callback, bool async = false)

Set the callback of motion.

bool **HasStreamCallback**(const *Stream* &stream) const

Has the callback of stream.

bool **HasMotionCallback**() const

Has the callback of motion.

virtual void **Start**(const *Source* &source)

Start capturing the source.

virtual void **Stop**(const *Source* &source)

Stop capturing the source.

void **WaitForStreams**()

Wait the streams are ready.

device::*StreamData* **GetStreamData**(const *Stream* &stream)

Get the latest data of stream.

device::*StreamData* **GetLatestStreamData**(const *Stream* &stream)

Deprecated:

Replaced by *GetStreamData(const Stream &stream)*

std::vector<device::*StreamData*> **GetStreamDatas**(const *Stream* &stream)

Get the datas of stream.

Note: default cache 4 datas at most.

---

void **DisableMotionDatas**()

Disable cache motion datas.

void **EnableMotionDatas**()

Enable cache motion datas.

void **EnableImuCorrespondence**(bool is_enable)

Enable motion datas timestamp correspondence.

void **EnableMotionDatas**(std::size_t max_size)

Enable cache motion datas.

std::vector<device::*MotionData*> **GetMotionDatas**()

Get the motion datas.

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

Public Static Functions

static std::shared_ptr<*Device*> **Create**(const std::string &name, std::shared_ptr<uvc::device> device)

Create the *Device* instance.

Parameters

- **name** – the device name.
- **device** – the device from uvc.

Returns

the *Device* instance.

6.2.2 device::Frame

class **Frame**

Frame with raw data.

Public Functions

inline **Frame**(const *StreamRequest* &request, const void *data)

Construct the frame with *StreamRequest* and raw data.

inline **Frame**(std::uint16_t width, std::uint16_t height, *Format* format, const void *data)

Construct the frame with stream info and raw data.

inline std::uint16_t **width**() const

Get the width.

inline std::uint16_t **height**() const

Get the height.

inline *Format* **format**() const

Get the format.

inline std::uint8_t *data()
Get the data.

inline const std::uint8_t *data() const
Get the const data.

inline std::size_t size() const
Get the size of data.

inline *Frame* clone() const
Clone a new frame.

6.2.3 device::StreamData

struct StreamData
Device stream data.

Public Members

std::shared_ptr<*ImgData*> img
ImgData.

std::shared_ptr<*Frame*> frame
Frame.

std::uint16_t frame_id
Frame ID.

6.2.4 device::MotionData

struct MotionData
Device motion data.

Public Members

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

6.3 Enums

6.3.1 Model

enum `mynteye::Model`

Device model.

Values:

enumerator **STANDARD**

Standard.

enumerator **STANDARD2**

Standard 2.

enumerator **STANDARD210A**

Standard 210a.

enumerator **STANDARD200B**

Standard 200b.

6.3.2 Stream

enum `mynteye::Stream`

Streams define different type of data.

Values:

enumerator **LEFT**

Left stream.

enumerator **RIGHT**

Right stream.

enumerator **LEFT_RECTIFIED**

Left stream, rectified.

enumerator **RIGHT_RECTIFIED**

Right stream, rectified.

enumerator **DISPARITY**

Disparity stream.

enumerator **DISPARITY_NORMALIZED**

Disparity stream, normalized.

enumerator **DEPTH**

Depth stream.

enumerator **POINTS**

Point cloud stream.

6.3.3 Capabilities

enum `mynteye::Capabilities`

Capabilities define the full set of functionality that the device might provide.

Values:

enumerator **STEREO**

Provides stereo stream.

enumerator **STEREO_COLOR**

Provide stereo color stream.

enumerator **COLOR**

Provides color stream.

enumerator **DEPTH**

Provides depth stream.

enumerator **POINTS**

Provides point cloud stream.

enumerator **FISHEYE**

Provides fisheye stream.

enumerator **INFRARED**

Provides infrared stream.

enumerator **INFRARED2**

Provides second infrared stream.

enumerator **IMU**

Provides IMU (accelerometer, gyroscope) data.

6.3.4 Info

enum `mynteye::Info`

Camera info fields are read-only strings that can be queried from the device.

Values:

enumerator **DEVICE_NAME**

Device name.

enumerator **SERIAL_NUMBER**

Serial number.

enumerator **FIRMWARE_VERSION**

Firmware version.

enumerator **HARDWARE_VERSION**

Hardware version.

enumerator **SPEC_VERSION**

Spec version.

enumerator **LENS_TYPE**

Lens type.

enumerator **IMU_TYPE**

IMU type.

enumerator **NOMINAL_BASELINE**

Nominal baseline.

enumerator **AUXILIARY_CHIP_VERSION**

Auxiliary chip version.

enumerator **ISP_VERSION**

Isp version.

6.3.5 Option

enum `mynteye::Option`

Camera control options define general configuration controls.

Values:

enumerator **GAIN**

Image gain, valid if manual-exposure.

range: [0,48], default: 24

enumerator **BRIGHTNESS**

Image brightness, valid if manual-exposure.

range: [0,240], default: 120

enumerator **CONTRAST**

Image contrast, valid if manual-exposure.

range: [0,254], default: 116

enumerator **FRAME_RATE**

Image frame rate, must set IMU_FREQUENCY together.

values: {10,15,20,25,30,35,40,45,50,55,60}, default: 25

enumerator **IMU_FREQUENCY**

IMU frequency, must set FRAME_RATE together.

values: {100,200,250,333,500}, default: 200

enumerator **EXPOSURE_MODE**

Exposure mode.

0: enable auto-exposure

1: disable auto-exposure (manual-exposure)

enumerator **MAX_GAIN**

Max gain, valid if auto-exposure.

range of standard 1: [0,48], default: 48

range of standard 2: [0,255], default: 8

enumerator **MAX_EXPOSURE_TIME**

Max exposure time, valid if auto-exposure.

range of standard 1: [0,240], default: 240

range of standard 2: [0,655], default: 333

enumerator **MIN_EXPOSURE_TIME**

min exposure time, valid if auto-exposure

range: [0,655], default: 0

enumerator **DESIRED_BRIGHTNESS**

Desired brightness, valid if auto-exposure.

range of standard 1: [0,255], default: 192

range of standard 2: [1,255], default: 122

enumerator **IR_CONTROL**

IR control.

range: [0,160], default: 0

enumerator **HDR_MODE**

HDR mode.

0: normal

1: WDR

enumerator **ACCELEROMETER_RANGE**

The range of accelerometer.

value of standard 1: {4,8,16,32}, default: 8

value of standard 2: {6,12,24,48}, default: 12

enumerator **GYROSCOPE_RANGE**

The range of gyroscope.

value of standard 1: {500,1000,2000,4000}, default: 1000

value of standard 2: {250,500,1000,2000,4000}, default: 1000

enumerator **ACCELEROMETER_LOW_PASS_FILTER**

The parameter of accelerometer low pass filter.

values: {0,1,2}, default: 2

enumerator **GYROSCOPE_LOW_PASS_FILTER**

The parameter of gyroscope low pass filter.

values: {23,64}, default: 64

enumerator **IIC_ADDRESS_SETTING**

The setting of IIC address.

range: [0,127], default: 0

enumerator **ZERO_DRIFT_CALIBRATION**

Zero drift calibration.

enumerator **ERASE_CHIP**

Erase chip.

enumerator **SYNC_TIMESTAMP**

Sync timestamp.

6.3.6 Source

enum `mynteye::Source`

Source allows the user to choose which data to be captured.

Values:

enumerator **VIDEO_STREAMING**

Video streaming of stereo, color, depth, etc.

enumerator **MOTION_TRACKING**

Motion tracking of IMU (accelerometer, gyroscope)

enumerator **ALL**

Enable everything together.

6.3.7 AddOns

enum `mynteye::AddOns`

Add-Ons are peripheral modules of our hardware.

Values:

enumerator **INFRARED**

Infrared.

enumerator **INFRARED2**

Second infrared.

6.3.8 Format

enum `mynteye::Format`

Formats define how each stream can be encoded.

Values:

enumerator **GREY**

Greyscale, 8 bits per pixel.

enumerator **YUYV**

YUV 4:2:2, 16 bits per pixel.

enumerator **BGR888**

BGR 8:8:8, 24 bits per pixel.

enumerator **RGB888**

RGB 8:8:8, 24 bits per pixel.

6.3.9 CalibrationModel

enum `mynteye::CalibrationModel`

Camera calibration model.

Values:

enumerator **PINHOLE**

Pinhole.

enumerator **KANNALA_BRANDT**

Equidistant: KANNALA_BRANDT.

enumerator **UNKNOW**

Unknow.

6.3.10 DisparityComputingMethod

enum `mynteye::DisparityComputingMethod`

Camera disparity computing method type.

Values:

enumerator **SGBM**

bm

enumerator **BM**

sgbm

enumerator **UNKNOW**

unknow

6.4 Types

6.4.1 OptionInfo

struct **OptionInfo**

Option info.

Public Members

std::int32_t **min**

Minimum value.

std::int32_t **max**

Maximum value.

std::int32_t **def**

Default value.

6.4.2 Resolution

struct **Resolution**

Resolution.

Public Members

std::uint16_t **width**

Width.

std::uint16_t **height**

Height.

6.4.3 StreamRequest

struct **StreamRequest**

Stream request.

Public Members

std::uint16_t **width**

Stream width in pixels.

std::uint16_t **height**

Stream height in pixels.

Format **format**

Stream pixel format.

std::uint16_t **fps**

Stream frames per second.

6.4.4 Intrinsics

IntrinsicsPinhole

struct **IntrinsicsPinhole** : public mynteye::IntrinsicsBase

Stream intrinsics (Pinhole)

Public Members

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.

std::uint8_t **model**

Deprecated:

Replaced by calib_model_.

The distortion model of the image

double **coeffs**[5]

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

IntrinsicsEquidistant

struct **IntrinsicsEquidistant** : public mynteye::IntrinsicsBase

Stream intrinsics (Equidistant: KANNALA_BRANDT)

Public Members

double **coeffs**[8]

The distortion coefficients: k2,k3,k4,k5,mu,mv,u0,v0.

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 **drift**[3]

Zero-drift: X, Y, Z.

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.

6.4.5 Extrinsics

struct **Extrinsics**

Extrinsics, represent how the different datas are connected.

Public Functions

inline *Extrinsics* **Inverse**() const

Inverse this extrinsics.

Returns
the inversed extrinsics.

Public Members

double **rotation**[3][3]

Rotation matrix.

double **translation**[3]

Translation vector.

6.4.6 ImgData

struct **ImgData**

Image data.

Public Members

std::uint16_t **frame_id**

Image frame id.

std::uint64_t **timestamp**

Image timestamp in 1us.

std::uint16_t **exposure_time**

Image exposure time, virtual value in [1, 480].

bool **is_ets** = false

Is external time source.

6.4.7 ImuData

struct **ImuData**

IMU data.

Public Members

std::uint32_t **frame_id**

IMU frame id.

std::uint8_t **flag**

IMU accel or gyro flag.

0: accel and gyro are both valid

1: accel is valid

2: gyro is valid

bool **is_ets** = false

Is external time source.

std::uint64_t **timestamp**

IMU timestamp in 1us.

double **accel**[3]

IMU accelerometer data for 3-axis: X, Y, Z.

double **gyro**[3]

IMU gyroscope data for 3-axis: X, Y, Z.

double **temperature**

IMU temperature.

6.5 Utils

6.5.1 select

std::shared_ptr<*Device*> `mynteye::device::`**`select`**`()`

Detecting MYNT EYE devices and prompt user to select one.

Returns

the selected device, or `nullptr` if none.

6.5.2 select_request

MYNTEYE_NAMESPACE::StreamRequest `mynteye::device::`**`select_request`**`(`const std::shared_ptr<*Device*> &device, bool *ok`)`

List stream requests and prompt user to select one.

Returns

the selected request.

6.5.3 get_real_exposure_time

float `mynteye::utils::`**`get_real_exposure_time`**`(`std::int32_t frame_rate, std::uint16_t exposure_time`)`

Get real exposure time in ms from virtual value, according to its frame rate.

Parameters

- **frame_rate** – the frame rate of the device.
- **exposure_time** – the virtual exposure time.

Returns

the real exposure time in ms, or the virtual value if frame rate is invalid.

6.5.4 get_sdk_root_dir

std::string mynteye::utils::**get_sdk_root_dir**()

Get sdk root dir.

6.5.5 get_sdk_install_dir

std::string mynteye::utils::**get_sdk_install_dir**()

Get sdk install dir.

CHAPTER

SEVEN

TECHNICAL SUPPORT

7.1 FAQ

If you have problems using MYNT EYE, please first check the following docs

http://support.myntai.com/hc/

7.2 Contact Us

If you continue to encounter problems, please contact customer service.

http://support.myntai.com/hc/request/new/

INDEX

M