MYNT® EYE S SDK Documentation

Release 2.3.4

MYNTAI

Mar 22, 2019

Contents

CHAPTER 1

MYNT® EYE product introduction

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 1280x800/60fps 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.

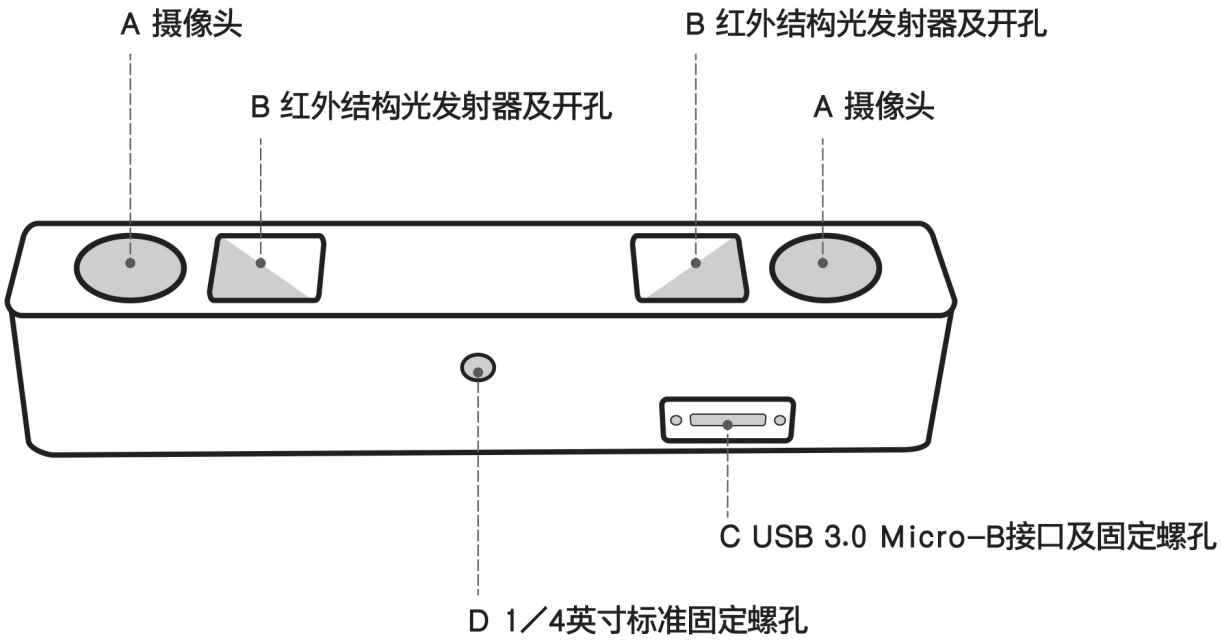
As a hardware product for research and development of stereo vision computing applications, MYNT® EYE S Series can be widely used in a field of visual positioning navigation (vSLAM), including visual real-time positioning navigation system of driverless vehicle and robots, visual positioning system of 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 MYNTEYE-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.

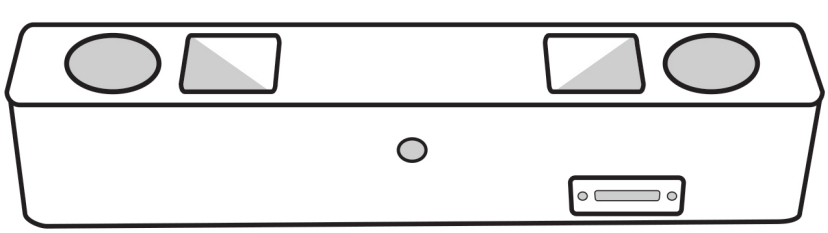
B. Infrared structured-light transmitter and outlet: the infrared structured-light can effectively solve the problem associated with the visual positioning calculations of white wall non-textured object(For non-IR version, the outlet is reserved but there is no internal structured-light emitter).

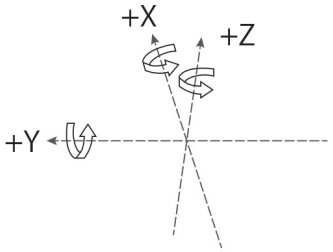
C. USB Micro-B interface and set screw holes: during usage, plug in the USB Micro-B cable and secure it by fastening the set screws to avoid damage to the interface and to ensure stability in connection.

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

1.3 MYNTEYE-S1030 IMU coordinata system

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





1.4 S1030-IR-120/Mono Product Specification

1.4.1 Product Specification:

Model	S1030-IR-120/Mono
Size	PCB dimension:149x24mm, Total dimension:165x31.5x29.6mm
Frame Rate	10/15/20/25/30/35/40/45/50/55/60FPS
Resolution	752*480,376*240
Depth Resolution	Base on CPU/GPU Up to 752*480@60FPS
Pixel Size	6.0*6.0μm
Baseline	120.0mm
Camera Lens	Replacable Standard M12
Visual Angle	D:146° H:122° V:76°
Focal Length	2.1mm
IR Support	Yes
IR detectable range	3m
Color Mode	Monochrome
Working Distance	0.7-2m
Scanning Mode	Global Shutter
Power	1~2.7W@5V DC from USB
Output data format	Raw data
Data transfer Interface	USB3.0
Weight	184g
UVC MODE	Yes

1.4.2 Work Environment

Operating Temperature	50°C
Storage Temperature	-20°C

1.4.3 Package:

Package Contents	MYNT EYE x1 USB Micro-B Cable x1

1.4.4 Warranty

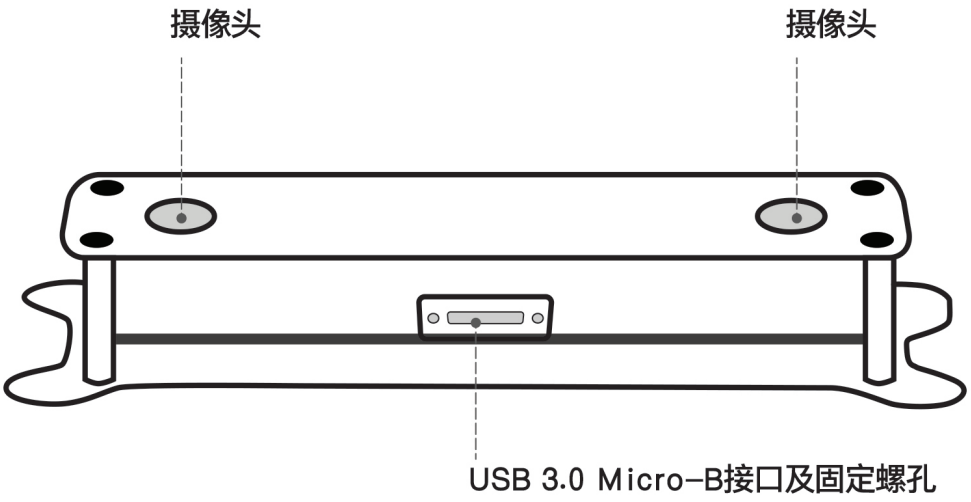
Product Warranty	12 Months Limited Manufacturer's Warranty

1.4.5 Accuracy

Depth Distance Deviation	Less than 4%

1.5 MYNTEYE-S2100 Size and structure

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

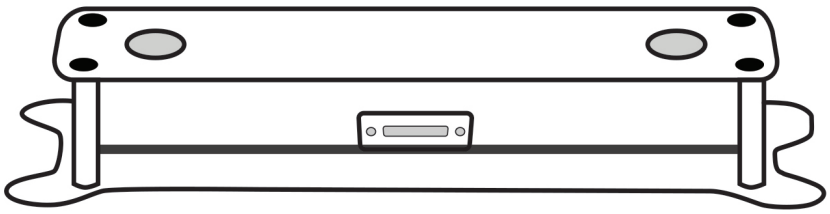


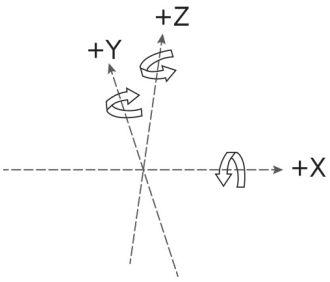
A. Camera:please pay attention to protect the camera sensor lenses, to avoid imaging quality degradation.

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.

1.6 MYNTEYE-S2100 IMU coordinata system

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





1.7 S2100-146/Color Product Specification

1.7.1 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
Pixel Size	3.0*3.0μm
Baseline	80.0mm
Camera Lens	Replacable Standard M7
Visual Angle	D:141° H:124° V:87°
Focal Length	0.95mm
Color Mode	Color
Working Distance	0.26-1m
Scanning Mode	Global Shutter
Power	1.1W@5V DC from USB
Output data format	YUYV
Data transfer Interface	USB3.0
Weight	62g
UVC MODE	Yes

1.7.2 Work Environment

Operating Temperature	-15°C~60°C
Storage Temperature	-20°C~80°C

1.7.3 Package:

Package Contents	MYNT EYE x1 USB Micro-B Cable x1

1.7.4 Warranty

Product Warranty	12 Months Limited Manufacturer's Warranty

1.7.5 Accuracy

Depth Distance Deviation	Less than 4%

CHAPTER 2

MYNT® EYE SDK

2.1 Changelog

2.1.1 2019-03-18

1. Add API to get subsidiary chip&ISP's version(Depend on S2100/S210A 1.1 firmware & 1.0 subsidiary chip firmware).
2. Fix point fragment issue in BM 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.

2.2 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 / 16.04 / 14.04
- 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.

2.3 Ubuntu SDK PPA Installation

Ubuntu 14.04	Ubuntu 16.04	Ubuntu 18.04
✓	✓	✓

2.3.1 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
```

2.3.2 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
```

2.3.3 Run samples

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

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

2.4 Windows SDK exe Installation

Windows 10
✓

2.4.1 Download and install SDK

Tip: Download here: mynteye-s-2.3.4-win-x64-opencv-3.4.3.exe Google Drive Baidu Pan .

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

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

2.4.2 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.

2.4.3 Start using SDK with Visual Studio 2017

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

2.5 Ubuntu SDK 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

2.5.1 Getting Source Code

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

2.5.2 Required Packages

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

- OpenCV

Tip: To build and install Opencv, Please refer to Installation in Linux . Alternatively, refer to the command below:

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

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

$ 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
```

2.5.3 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 independency* .

Build and install:

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

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

2.5.4 Building samples

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

Run samples:

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

Tutorial samples, please read *MYNT® EYE Data* and *MYNT® EYE Control* .

2.5.5 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.

2.5.6 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.

2.6 Windows SDK 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.

2.6.1 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
```

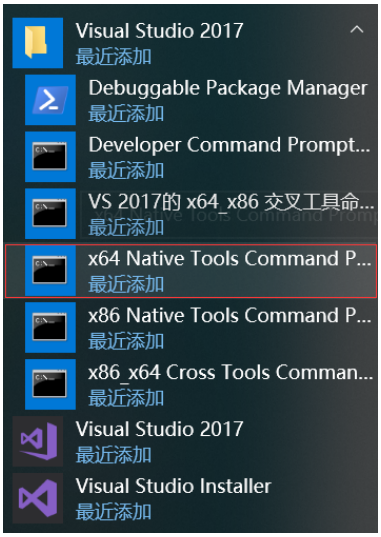
Visual Studio (provide compilation)

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

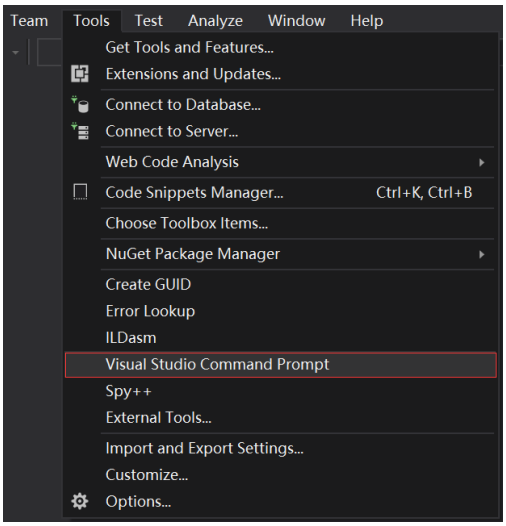
After installing Visual Studio, confirm that the following command can run in the Visual Studio Command Prompt:

```
>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.

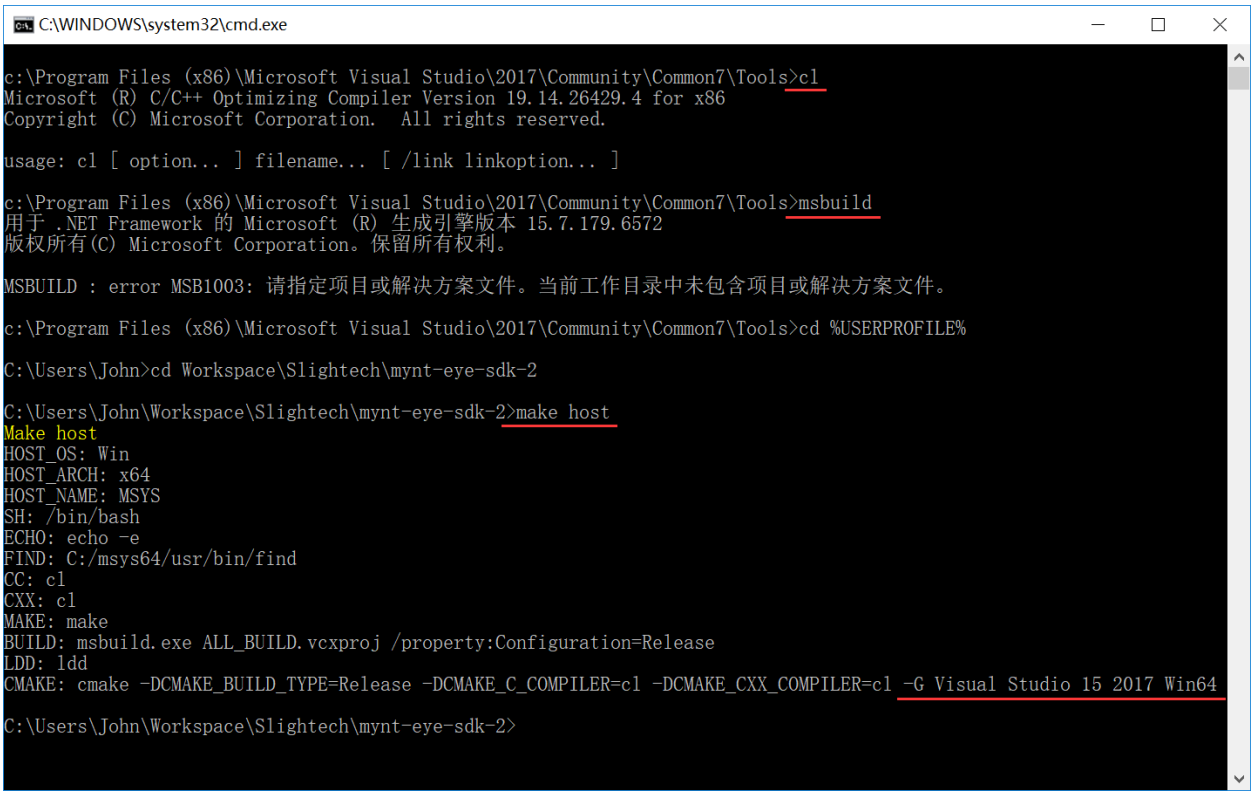


However, if you do not have the Visual Studio 2015 Tools menu, you can add one yourself.

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
```

2.6.2 Getting Source Code

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

2.6.3 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
```

2.6.4 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 independency* .

Build and install:

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

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

2.6.5 Building samples

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

Run samples:

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

For tutorial samples, please read *MYNT® EYE Data* and *MYNT® EYE Control* .

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 .

2.6.6 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.

2.6.7 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.

2.7 MacOS Installation x

TODO

2.8 ROS Installation

ROS Kinetic	ROS Indigo
✓	✓

2.8.1 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
```

2.8.2 Compiling Code

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

2.8.3 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
```

2.8.4 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
```

2.8.5 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

2.8.6 Conclusion

About more details, check the *How to use ROS* .

2.9 OpenCV independency

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 .

CHAPTER 3

MYNT® EYE Firmware

3.1 Firmware and SDK compatibility

S1030 Firmwares	SDK Version
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
MYNTEYE_S_2.2.2.img	2.3.0 (2.2.2-rc1 ~ 2.3.0)
MYNTEYE_S_2.3.0.img	2.3.0 (2.2.2-rc1 ~ 2.3.3)
MYNTEYE_S_2.4.0.img	2.3.4

S2100 Firmwares	SDK Version
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 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 How to upgrade the firmware

Please use the MYNT EYE TOOL to upgrade the firmware.

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

```
Firmwares/
├─Checksum.txt                  # file checksum
├─MYNTEYE_S_2.4.0.img           # S1030 firmware
├─MYNTEYE_S2100_1.1.img         # S2100 firmware
├─...
└─setup.zip                    # MYNTEYE TOOL zip
```

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

3.2.1 Download preparation

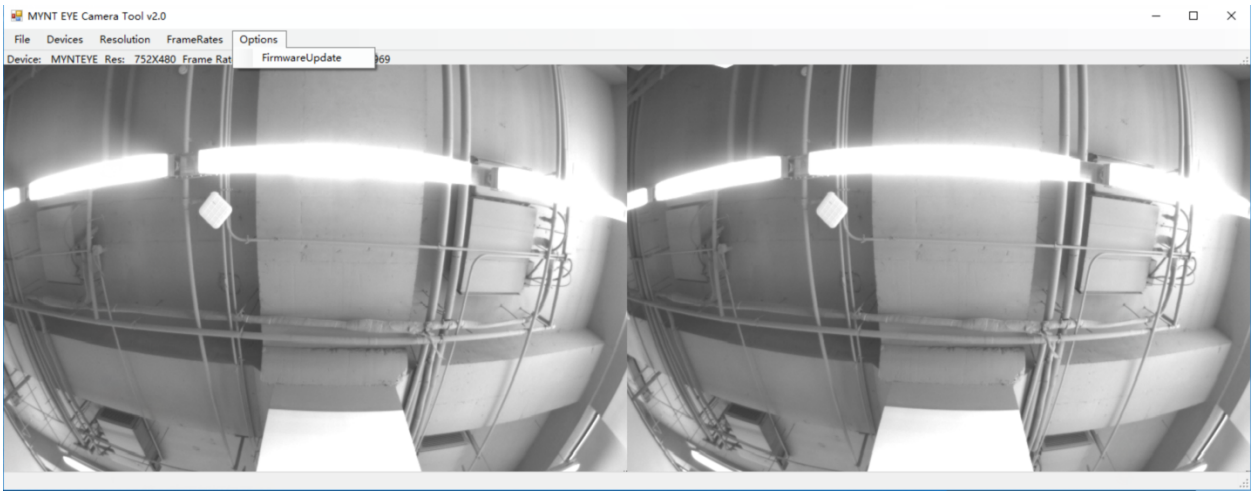
- Download and unzip `setup.zip`
- Find firmware, such as `MYNTEYE_S_2.4.0.img`
  - Please refer to *Firmware and SDK compatibility* to select the firmware suitable for the SDK version
  - Please refer to `Checksum.txt` to find the firmware check code as follows:
    - Run the command in CMD `certutil -hashfile <*.img> MD5` .
    - If the check code is incorrect, it means that the download went wrong. Please download it again!

3.2.2 Install MYNT EYE TOOL

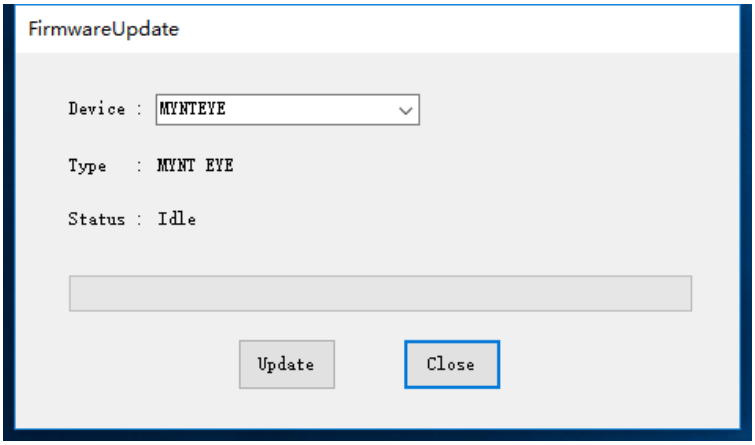
- Double click on `setup.msi` and install the application.

3.2.3 Update Firmware

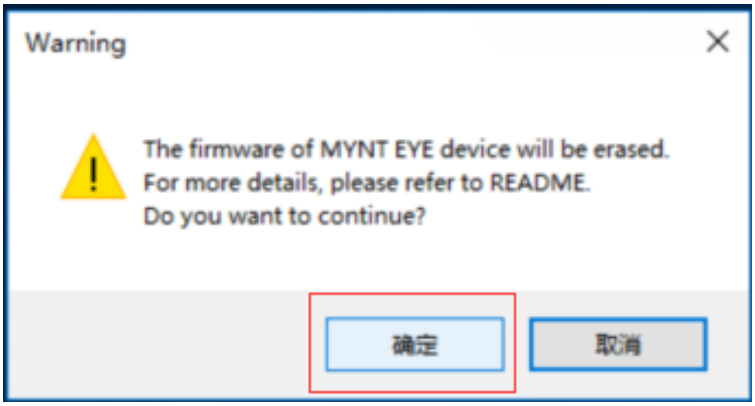
- Plug in the MYNT® EYE camera into a USB3.0 port
- Open MYNT EYE TOOL and select `Options/FirmwareUpdate` .

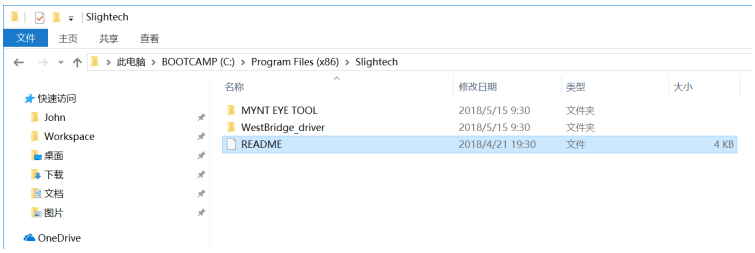


- Click `Update` .

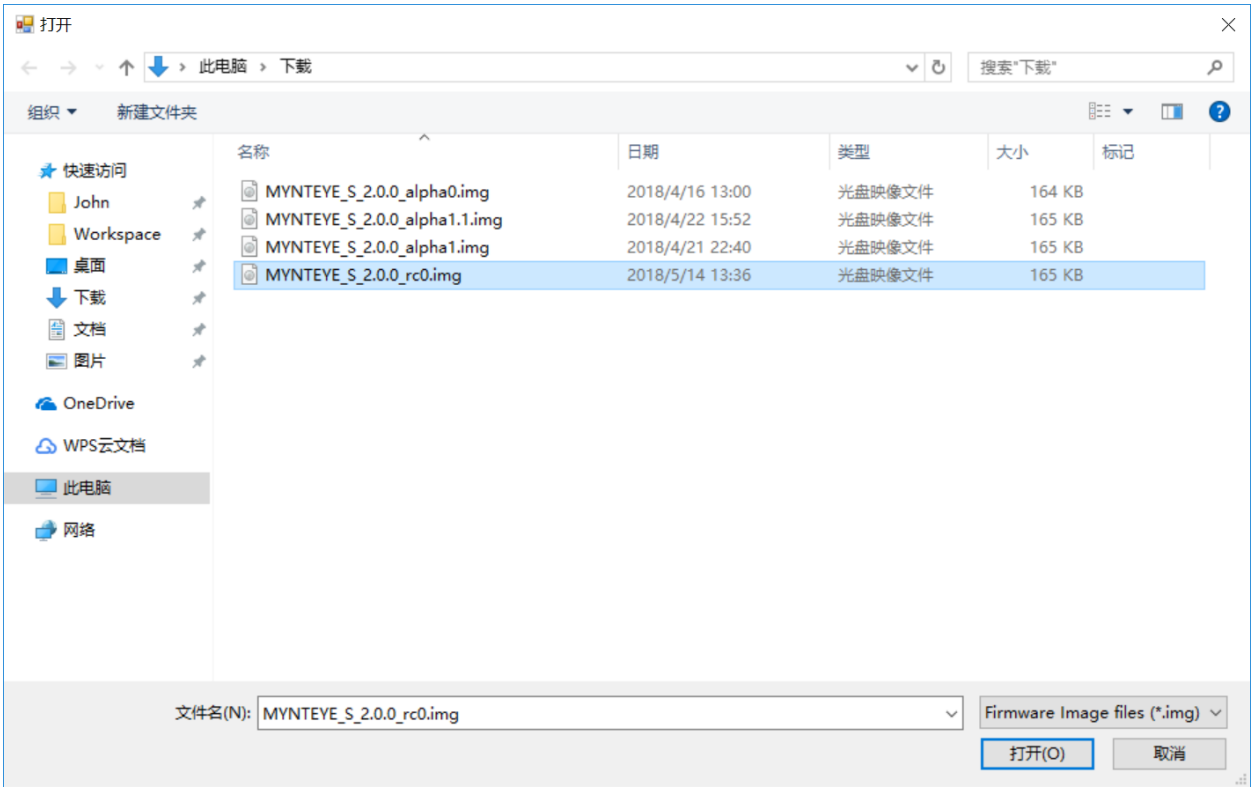


- A warning dialog box will pop up, click `yes` .
  - This operation will erase the firmware, for details see README.
    - Usually, the MYNT EYE TOOL automatically installs the driver during the upgrade process.
    - If the upgrade fails, refer to README.

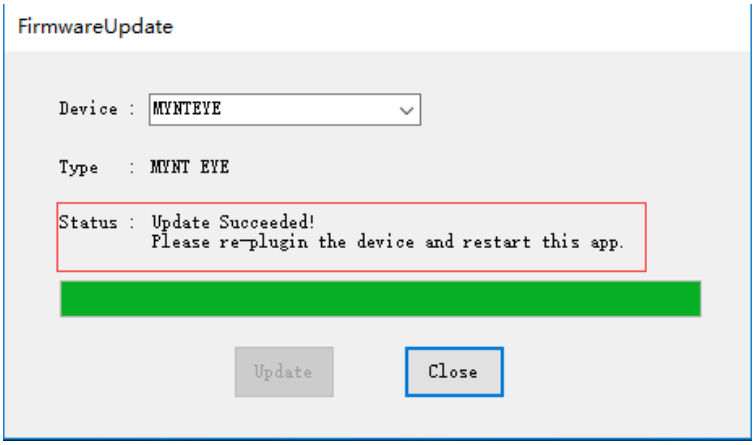




- In the open file selection box, select the firmware you want to upgrade and start upgrading.



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

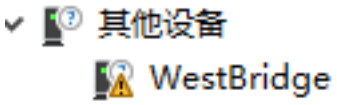


- Close the MYNT EYE TOOLfinish.

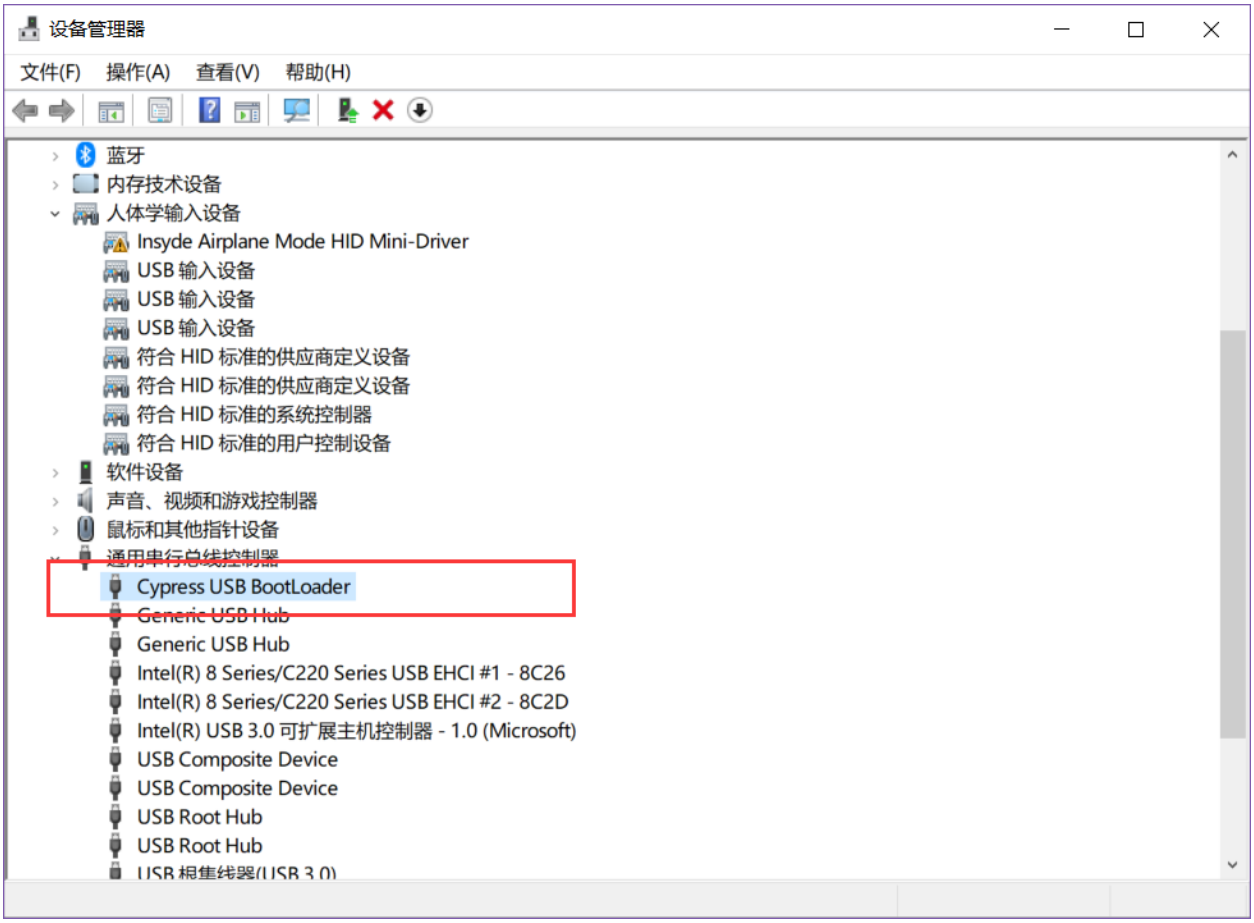
Attention: If you can't find MYNT image device, `WestBridge_driver`, and `Cypress USB BootLoader` at the same time in the device manager, try another computer to perform the above operation. If you can not upgrade successfully, please contact us in time.

3.2.4 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.



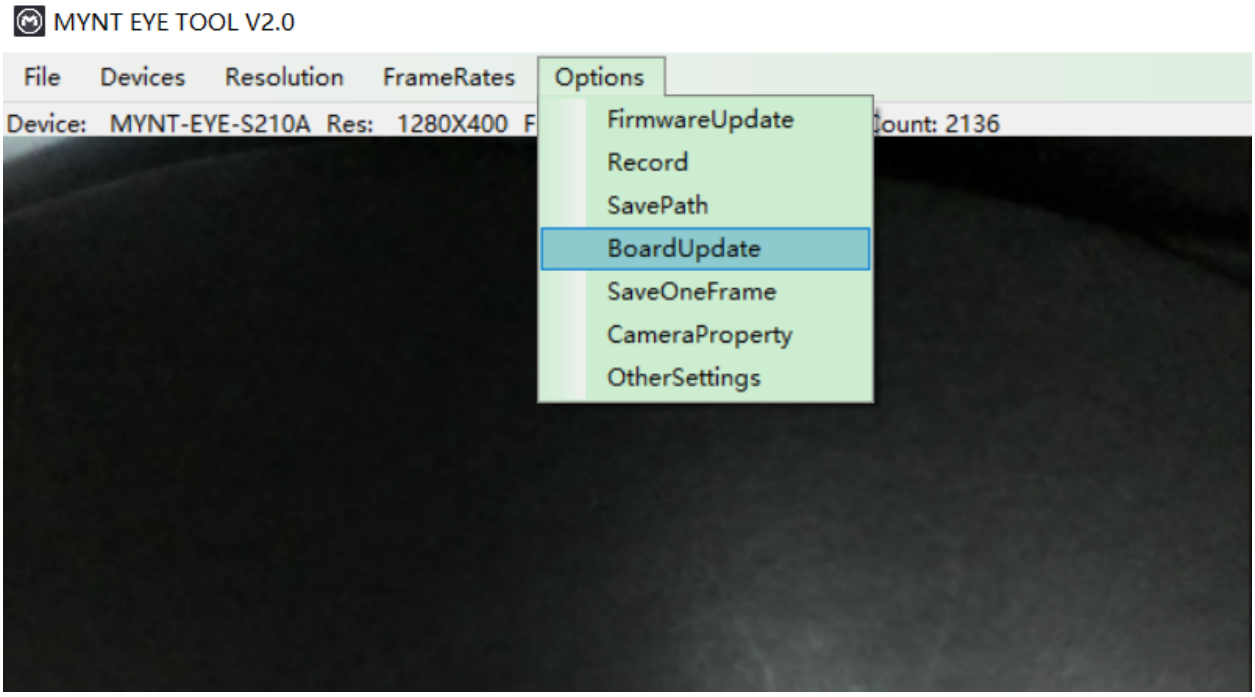
- Then plug in the camera and open the application again to update.

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.

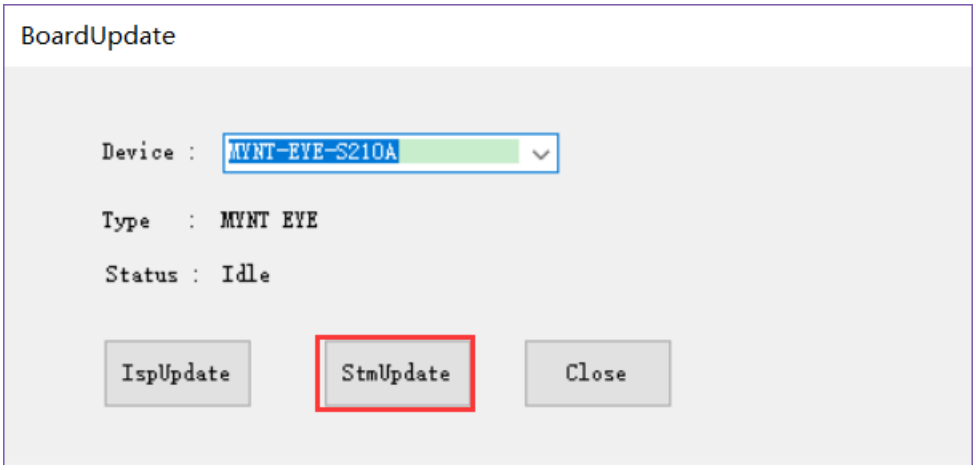
3.3 How to upgrade the auxiliary chip

3.3.1 Update auxiliary chip

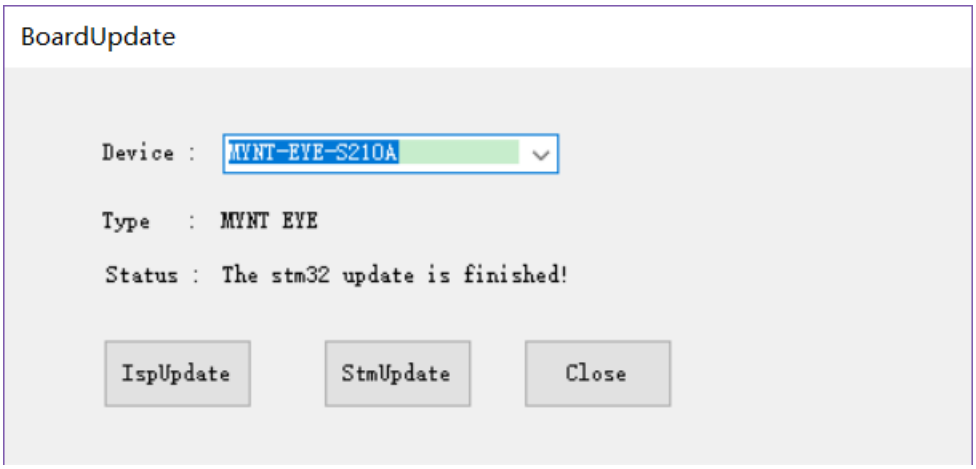
- Plug in the MYNT® EYE camera into a USB3.0 port
- Open MYNT EYE TOOL and select `Options/BoardUpdate` .



- Click `StmUpdate` .



- In the open file selection box, select the firmware `MYNTEYE-S210x-auxiliary-chip-v1.0.bin` and start upgrading.
- Once the upgrade is complete, it will display update finished.



3.4 Change from SDK 1.x to 2.x

To replace the SDK version 1.x to 2.x, need to:

1 Install SDK 2, Check the *MYNT® EYE SDK* .

2 Upgrade firmware to 2.x versionCheck the *MYNT® EYE Firmware* .

3 After launch the SDK 1.x , the image calibration parameters will be saved in `<MYNTEYE_SDK_ROOT>/settings/SN*.conf` .Please check the *Write image parameters* and write `SN*.conf` into the devices.

CHAPTER 4

MYNT® EYE Data

4.1 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/tutorials/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 .

4.2 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 S2100/S210A are in `tools/writer/config/S210A` the image calibration parameters of S1030 are in `tools/writer/config/S1030`

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/tutorials/get_img_params
I0510 15:00:22.643263  6980 utils.cc:26] Detecting MYNT EYE devices
I0510 15:00:23.138811  6980 utils.cc:33] MYNT EYE devices:
I0510 15:00:23.138849  6980 utils.cc:37]   index: 0, name: MYNT-EYE-S1000
I0510 15:00:23.138855  6980 utils.cc:43] Only one MYNT EYE device, select index: 0
I0510 15:00:23.210491  6980 get_img_params.cc:23] Intrinsics left: {width: 752,␣
↪height: 480, fx: 736.38305001095545776, fy: 723.50066150722432212, cx: 356.
↪91961817119693023, cy: 217.27271340923883258, model: 0, coeffs: [-0.
↪54898645145016478, 0.52837141203888638, 0.00000000000000000, 0.00000000000000000, 0.
↪00000000000000000]}
I0510 15:00:23.210551  6980 get_img_params.cc:24] Intrinsics right: {width: 752,␣
↪height: 480, fx: 736.38305001095545776, fy: 723.50066150722432212, cx: 456.
↪68367112303980093, cy: 250.70083335536796199, model: 0, coeffs: [-0.
↪51012886039889305, 0.38764476500996770, 0.00000000000000000, 0.00000000000000000, 0.
↪00000000000000000]}
I0510 15:00:23.210577  6980 get_img_params.cc:26] Extrinsics left to right:
↪{rotation: [0.99701893306553813, -0.00095378124886237, -0.07715139279485062, 0.
↪00144939967628305, 0.99997867219985104, 0.00636823256494144, 0.07714367342455503, -
↪0.00646107164115277, 0.99699905125522237], translation: [-118.88991734400046596, -0.
↪04560580387053091, -3.95313736911933855]}
```

Complete code examples, see get_img_params.cc .

4.3 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() << "}";
```

(continues on next page)

(continued from previous page)

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

Complete code examples, see get_imu_params.cc .

4.4 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;
  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 .

4.5 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 data 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);
  }

  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 .

4.6 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 original binocular image Get stereo camera correction image* .

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

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()) {
  //   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 .

4.7 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 original binocular image Get stereo camera correction image*.

In addition, it is recommended to use plugins to calculate depth: Depth images work better and operate faster. Please refer to *Using the plugin to get data*.

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);
  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 .

4.8 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 original binocular image Get stereo camera correction image* .

It is recommended to use plugin to calculate depth: the depth map will be better with a higher frame rate. Please see *Using the plugin to get data* for detail.

Sample code snippet:

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

api->EnableStreamData(Stream::POINTS);

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

(continues on next page)

(continued from previous page)

```
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));
  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_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 tutorials/CMakeLists.txt to the path of `PCLConfig.cmake` . You can find `CMAKE_PREFIX_PATH` near `find_package(PCL)` .

4.9 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 data from callbacks* .

Sample code snippet:

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

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

(continues on next page)

(continued from previous page)

```
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();
  /*
  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);
  if (!motion_datas.empty()) {
    painter.DrawImuData(img, *motion_datas[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 .

4.10 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");

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/tutorials/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
```

(continues on next page)

(continued from previous page)

```
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
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
```

(continues on next page)

(continued from previous page)

```
I/get_imu_correspondence.cc:61 Imu timestamp: 171421660, diff_prev=2010, diff_
↪img=18624
```

Complete code examples, see get_imu_correspondence.cc .

4.11 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.

// 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);
```

(continues on next page)

(continued from previous page)

```
        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);

  // 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
```

(continues on next page)

(continued from previous page)

```
    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 .

4.12 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_cuda9.1_opencv3.4.0.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
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.

Listing 1: e.g. libmynteye.conf

```
# libmynteye configuration
#
# 1) Copy this file to: /etc/ld.so.conf.d/libmynteye.conf
# 2) Run this cmd in Terminal: sudo /sbin/ldconfig

/usr/local/cuda/lib64
$HOME/opencv-3.4.1/lib
```

4.13 Save device infomation and parameters

The SDK provides a tool `save_all_infos` for save information and parameters. For more information, please read tools/README.md .

Reference commands:

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

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

Reference result on Linux:

```
$ ./tools/_output/bin/writer/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
```

4.14 Write image parameters

The SDK provides a tool `img_params_writer` for writing image parameters. For details, read tools/README.md .

For getting image parameters, please read *Get image calibration parameters*. This is used to calculate the deviation.

Reference commands:

```
./tools/_output/bin/writer/img_params_writer tools/writer/config/img.params

# Windows
.\tools\_output\bin\writer\img_params_writer.bat tools\writer\config\img.params
```

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

And, tools/writer/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 S2100/S210A are in `tools/writer/config/S210A` The image calibration parameters of S1030 are in `tools/writer/config/S1030`

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

4.15 Write IMU parameters

SDK provides the tool `imu_params_writer` to write IMU parameters. For deltail, please read tools/README.md .

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

Reference commands:

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

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

The path of parameters file can be found in tools/writer/config/img.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.

CHAPTER 5

MYNT® EYE Control

5.1 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 s2100/s210a, 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.
```

(continues on next page)

(continued from previous page)

```
// FRAME_RATE values: 10, 15, 20, 25, 30, 35, 40, 45, 50, 55
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);
```

s2100/s210a

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

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/tutorials/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
```

s2100/s210a

```
$ ./samples/_output/bin/tutorials/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
```

(continues on next page)

(continued from previous page)

```
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 framerate.cc .

5.2 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 s2100/s210a, the available settings are:

- The effective range of accelerometer(unit:g): 6, 12, 24, 32.
- 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);
```

s2100/s210a

```
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, 32
api->SetOptionValue(Option::ACCELEROMETER_RANGE, 6);
// GYROSCOPE_RANGE values: 250, 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);
```

Reference running results on Linux:

s1030

```
$ ./samples/_output/bin/tutorials/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
```

s2100/s210a

```
$ ./samples/_output/bin/tutorials/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
```

(continues on next page)

(continued from previous page)

```
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 imu_range.cc.

5.3 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 s2100/s210a, 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);
```

s2100/s210a

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

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

// 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/tutorials/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
```

s2100/s210a

```
$ ./samples/_output/bin/tutorials/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
```

(continues on next page)

(continued from previous page)

```
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/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 auto_exposure.cc .

5.4 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 s2100/s210a, 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 [0,255], default 127
api->SetOptionValue(Option::CONTRAST, 127);

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);
```

s2100/s210a

```
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 [0,240], default 120
api->SetOptionValue(Option::BRIGHTNESS, 120);

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/tutorials/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 127
```

s2100/s210a

```
$ ./samples/_output/bin/tutorials/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,␣
↪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
```

(continues on next page)

(continued from previous page)

```
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 120
```

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

Complete code samplessee manual_exposure.cc .

5.5 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 s2100/s210a doesn't support this feature.

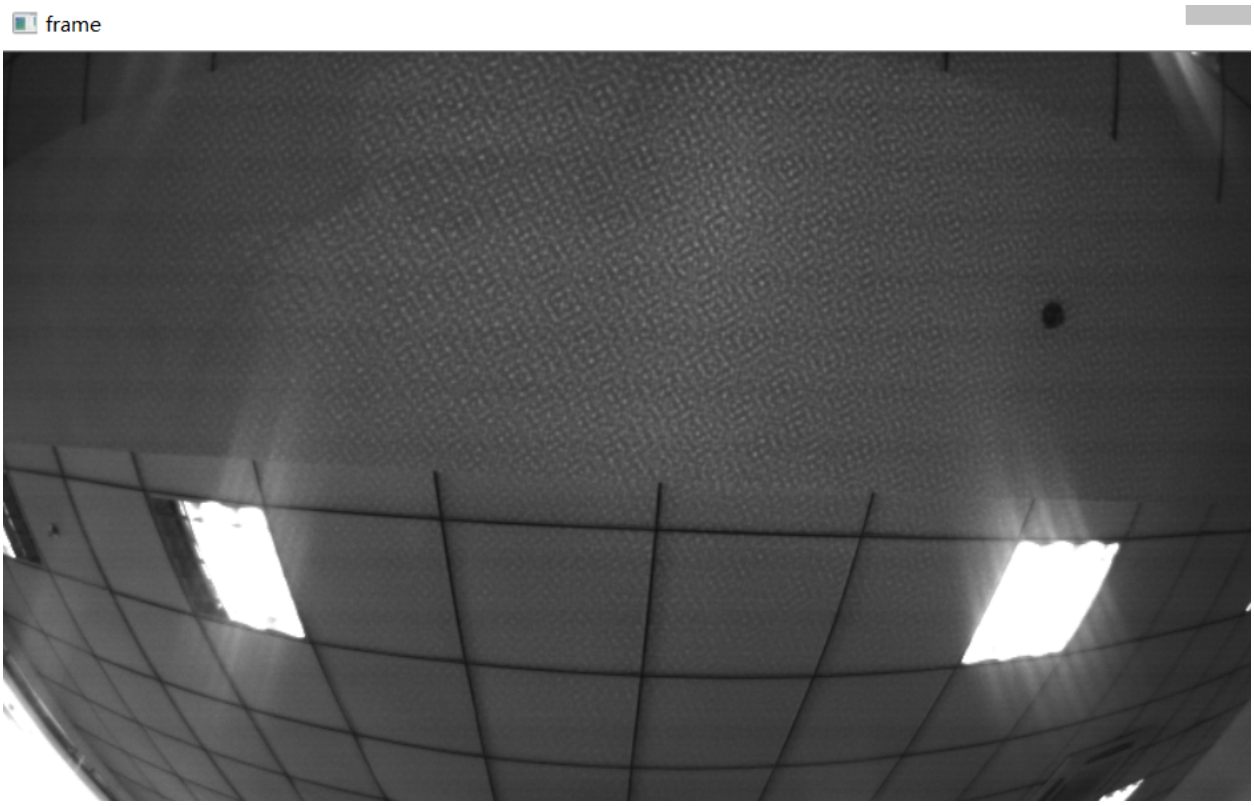
Reference Code:

```
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/tutorials/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
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. In order to keep IR enabled, you must set the IR value after turning on the device.

Complete code samplessee infrared.cc .

5.6 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;
```

(continues on next page)

(continued from previous page)

```
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/tutorials/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 imu_low_pass_filter.cc

CHAPTER 6

Running log

6.1 Enable log file

Tip: If import glog to build.

The general configuration of the log in the head file logger.h .

Uncomment `FLAGS_log_dir = \".\";` recompile and save to current work directory. Run `camera_a` log file as follows:

```
<workdir>/
├─camera_a.ERROR
├─camera_a.FATAL
├─camera_a.INFO
├─camera_a.WARNING
├─camera_a.john-ubuntu.john.log.ERROR.20180513-141833.519
├─camera_a.john-ubuntu.john.log.FATAL.20180513-141833.519
├─camera_a.john-ubuntu.john.log.INFO.20180513-141832.519
└─camera_a.john-ubuntu.john.log.WARNING.20180513-141833.519
```

`camera_a.INFO` shows the program and levers of log it is running. The link to the real log file is `camera_a.john-ubuntu.john.log.INFO.20180513-141832.519`. Even if it ran several times, `camera_a.INFO` still leaves the link to last log file.

Excute *make cleanlog* to clean all log files.

6.2 Enabled detailed level

Tip: If import glog to build.

The general configuration of the log is in the head file logger.h .

Uncomment `FLAGS_v = 2` ; and recompile to enable the detail levels, the log is printed by `VLOG(n)`

For information on how to use the log library, such as how to configure, print, etc., please open its document and learn more:

```
$ ./scripts/open.sh third_party/glog/doc/glog.html
```

CHAPTER 7

Wrapper interface

7.1 How to use ROS

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

Tip: Before doing below you need open a terminal to launch ros node first

`rostopic list` lists all released nodes:

```
$ rostopic list
/mynteye/depth/image_raw
/mynteye/disparity/image_norm
/mynteye/disparity/image_raw
/mynteye/imu/data_raw
/mynteye/left/camera_info
/mynteye/left/image_raw
/mynteye/left/image_rect
/mynteye/points/data_raw
/mynteye/right/camera_info
/mynteye/right/image_raw
/mynteye/right/image_rect
/mynteye/temp/data_raw
...
```

`rostopic hz <topic>` checks the data:

```
$ rostopic hz /mynteye/imu/data_raw
subscribed to [/mynteye/imu/data_raw]
average rate: 505.953
  min: 0.000s max: 0.018s std dev: 0.00324s window: 478
average rate: 500.901
  min: 0.000s max: 0.018s std dev: 0.00327s window: 975
average rate: 500.375
```

(continues on next page)

(continued from previous page)

```
  min: 0.000s max: 0.019s std dev: 0.00329s window: 1468
...
```

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

The ROS file is structured like follows:

```
<sdk>/wrappers/ros/
├─src/
│  └─mynt_eye_ros_wrapper/
│     ├─launch/
│     │  ├─display.launch
│     │  └─mynteye.launch
│     ├─msg/
│     ├─rviz/
│     ├─src/
│     │  ├─wrapper_node.cc
│     │  └─wrapper_nodelet.cc
│     ├─CMakeLists.txt
│     ├─nodelet_plugins.xml
│     └─package.xml
└─README.md
```

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

```
# s2100/s210a modify frame/resolution
<arg name="request_index" default="$(arg index_s2_2)" />

# s1030 modify frame/imu hz
<!-- standard/frame_rate range: {10,15,20,25,30,35,40,45,50,55,60} -->
<arg name="standard/frame_rate" default="-1" />
<!-- <arg name="standard/frame_rate" default="25" /> -->

<!-- standard/imu_frequency range: {100,200,250,333,500} -->
<arg name="standard/imu_frequency" default="-1" />
<!-- <arg name="standard/imu_frequency" default="200" /> -->
...

# s2100 modify brightness
<!-- standard2/brightness range: [0,240] -->
<arg name="standard2/brightness" default="-1" />
<!-- <arg name="standard2/brightness" default="120" /> -->
...

# s210a modify brightness
<!-- standard210a/brightness range: [0,240] -->
<arg name="standard210a/brightness" default="-1" />
<!-- <arg name="standard210a/brightness" default="120" /> -->
...
```

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

For printing debug info, replace `Info` in `wrapper_node.cc` to `Debug` :

```
ros::console::set_logger_level(
    ROSCONSOLE_DEFAULT_NAME, ros::console::levels::Info);
```

CHAPTER 8

Data Analytics

8.1 Recording data sets

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

Reference run command:

```
./tools/_output/bin/dataset/record2

# Windows
.\tools\_output\bin\dataset\record2.bat
```

Reference run results on Linux:

```
$ ./tools/_output/bin/dataset/record
I0513 21:28:57.128947 11487 utils.cc:26] Detecting MYNT EYE devices
I0513 21:28:57.807116 11487 utils.cc:33] MYNT EYE devices:
I0513 21:28:57.807155 11487 utils.cc:37]   index: 0, name: MYNT-EYE-S1000
I0513 21:28:57.807163 11487 utils.cc:43] Only one MYNT EYE device, select index: 0
I0513 21:28:57.808437 11487 channels.cc:114] Option::GAIN: min=0, max=48, def=24,␣
↪cur=24
I0513 21:28:57.809999 11487 channels.cc:114] Option::BRIGHTNESS: min=0, max=240,␣
↪def=120, cur=120
I0513 21:28:57.818678 11487 channels.cc:114] Option::CONTRAST: min=0, max=255,␣
↪def=127, cur=127
I0513 21:28:57.831529 11487 channels.cc:114] Option::FRAME_RATE: min=10, max=60,␣
↪def=25, cur=25
I0513 21:28:57.848914 11487 channels.cc:114] Option::IMU_FREQUENCY: min=100, max=500,␣
↪def=200, cur=500
I0513 21:28:57.865185 11487 channels.cc:114] Option::EXPOSURE_MODE: min=0, max=1,␣
↪def=0, cur=0
I0513 21:28:57.881434 11487 channels.cc:114] Option::MAX_GAIN: min=0, max=48, def=48,␣
↪cur=48
I0513 21:28:57.897598 11487 channels.cc:114] Option::MAX_EXPOSURE_TIME: min=0,␣
↪max=240, def=240, cur=240
```

(continues on next page)

(continued from previous page)

```
I0513 21:28:57.913918 11487 channels.cc:114] Option::DESIRED_BRIGHTNESS: min=0,␣
↪max=255, def=192, cur=192
I0513 21:28:57.930177 11487 channels.cc:114] Option::IR_CONTROL: min=0, max=160,␣
↪def=0, cur=0
I0513 21:28:57.946341 11487 channels.cc:114] Option::HDR_MODE: min=0, max=1, def=0,␣
↪cur=0
Saved 1007 imgs, 20040 imus to ./dataset
I0513 21:29:38.608772 11487 record.cc:118] Time beg: 2018-05-13 21:28:58.255395, end:␣
↪2018-05-13 21:29:38.578696, cost: 40323.3ms
I0513 21:29:38.608853 11487 record.cc:121] Img count: 1007, fps: 24.9732
I0513 21:29:38.608873 11487 record.cc:123] Imu count: 20040, hz: 496.983
```

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

Record contents:

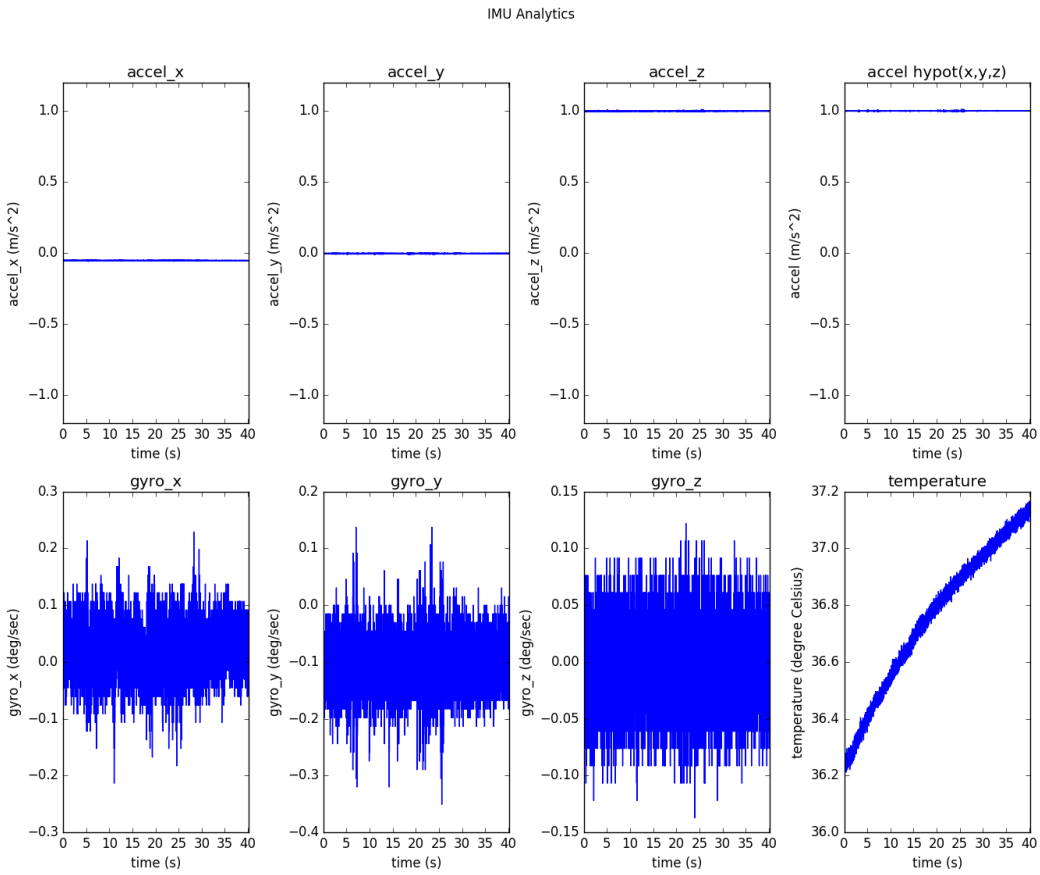
```
<workdir>/
└─dataset/
    ├─left/
    │   ├─stream.txt  # Image infomation
    │   ├─000000.png  # Imageindex 0
    │   └─...
    ├─right/
    │   ├─stream.txt  # Image information
    │   ├─000000.png  # Imageindex 0
    │   └─...
    └─motion.txt  # IMU information
```

8.2 Analyzing IMU

The SDK provides the script `imu_analytics.py` for IMU analysis. The tool details can be seen in tools/README.md . Refer to run commands and results on Linux:

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

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



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

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

8.3 Analyze time stamps

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

Reference run commands and results on Linux:

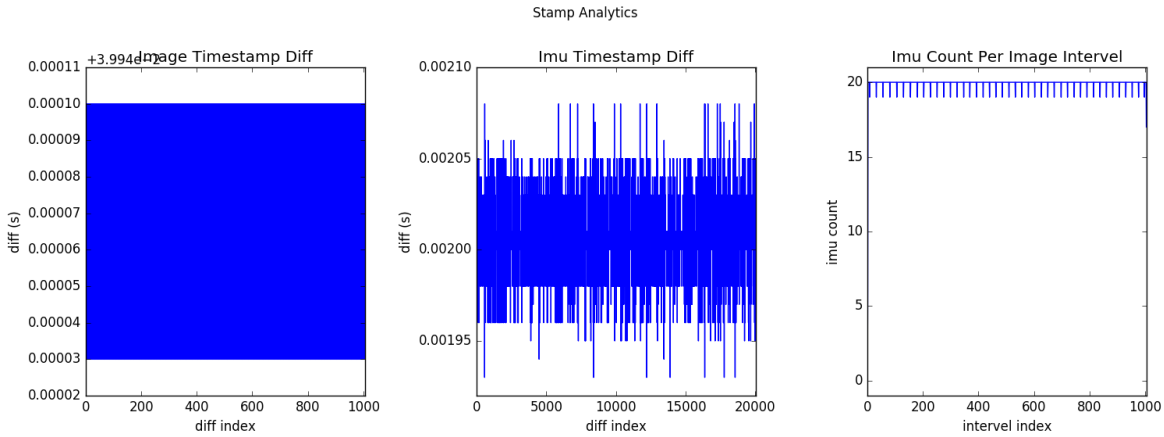
```
$ python tools/analytics/stamp_analytics.py -i dataset -c tools/config/mynteye/
↪mynteye_config.yaml
stamp analytics ...
  input: dataset
  outdir: dataset
open dataset ...
save to binary files ...
  binimg: dataset/stamp_analytics_img.bin
  binimu: dataset/stamp_analytics_imu.bin
  img: 1007, imu: 20040

rate (Hz)
```

(continues on next page)

(continued from previous page)

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

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

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

image timestamp duplicates: 0

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

The analysis result graph will be saved in the dataset directory, as follows:



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

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

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

CHAPTER 9

Open Source project Support

9.1 How to calibrate MYNTEYE by kalibr

9.1.1 Target

- Calibrate the pose relationship between left and right camera
- Calibrate the pose relationship left camera between and IMU

9.1.2 Preparation

- **Install kalibr**:Refer to kalibr wiki and follow the steps to install
- **Calibration board** kalibr supports `chessbord` , `circlegrid` , `aprilgrid` ,choose `aprilgrid` here Calibration board file can be directly download , Or you can also generate calibration board by Kalibr tool.

```
$ kalibr_create_target_pdf --type 'apriltag' --nx 6 --ny 6 --tsize 0.08 --tspace 0.3
```

View parameters' meanings by kalibr_create)target_pdf command:

```
$ kalibr_create_target_pdf --h
usage:
    Example Aprilgrid:
        kalibr_create_target_pdf --type apriltag --nx 6 --ny 6 --tsize 0.08 --tspace␣
↪0.3
    Example Checkerboard:
        kalibr_create_target_pdf --type checkerboard --nx 6 --ny 6 -csx 0.05 --csy 0.1


Generate PDFs of calibration patterns.

optional arguments:
  -h, --help            show this help message and exit
```

(continues on next page)

(continued from previous page)

```
Output options:
  output                Output filename
  --eps                 Also output an EPS file

Generic grid options:
  --type GRIDTYPE       The grid pattern type. ('apriltag' or 'checkerboard')
  --nx N_COLS           The number of tags in x direction (default: 6)
  --ny N_ROWS           The number of tags in y direction (default: 7)

Apriltag arguments:
  --tsize TSIZE         The size of one tag [m] (default: 0.08)
  --tspace TAGSPACING   The space between the tags in fraction of the edge size
                       [0..1] (default: 0.3)
  --tfam TAGFAMILIY     Familiy of April tags ['t16h5', 't25h9', 't25h7',
                       't36h11'] (default: t36h11)

Checkerboard arguments:
  --csx CHESSSZX        The size of one chessboard square in x direction [m]
                       (default: 0.05)
  --csy CHESSSZY        The size of one chessboard square in y direction [m]
                       (default: 0.05)
```

- **Calibrate the intrinsic IMU parameters** kalibr requires imu data to be calibrated by intrinsic parameters by default.The intrinsic parameters calibration tool uses imu-tk .
- **Count imu data parameter**
  - noise density
  - bias random walk

Using Allan analyzing tool imu_utils, We can get the characteristics of above imu data,and to format the output as `imu.yaml`

```
#Accelerometers
accelerometer_noise_density: 0.02680146180736048   #Noise density (continuous-time)
accelerometer_random_walk:   0.0026296086159332804   #Bias random walk

#Gyroscopes
gyroscope_noise_density:     0.008882328296710996   #Noise density (continuous-time)
gyroscope_random_walk:       0.00037956578292701033   #Bias random walk

rostopic:                    /mynteye/imu/data_raw      #the IMU ROS topic
update_rate:                 200.0      #Hz (for discretization of the values above)
```

9.1.3 Calibrate the pose relationship between left and right camera

- Collect calibration images: kalibr supports the collection of the required calibration images through two ways:by `rosbag` or collect offline images . Using `rosbag` here for convenience,Reference link for collecting images.
- Method of collecting images by `rosbag` :fix mynteye camera,move `aprilgrid` calibration board in the camera field of view.
- To increase the calibration time,try to use image acquisition data with lower frame rate,kalibr recommends using `4Hz` frame rate,here uses `10hz` .

- MYNTEYE S series camera offers images at least 10Hz,You can use topic_tools to modify frequency,because using 10Hz requires more calibration time.
- Record `static.bag` : After fix the mynteye camera,start wrapper, record the topic of the left and right images to `static.bag` .

```
$ source wrappers/ros/devel/setup.bash
$ roslaunch mynt_eye_ros_wrapper display.launch
$ cd ~
$ mkdir -p bag
$ cd bag
$ rosbag record -O static_10hz /mynteye/left/image_raw /mynteye/right/image_raw
↪#recommand use 10hz,you can also use topic_tools to publish 4hz.
```

- kalibr calibration:

```
$ kalibr_calibrate_cameras --target aprilgrid.yaml --bag ~/bag/static_10hz.bag --
↪models pinhole-radtan pinhole-radtan --topics /mynteye/left/image_raw /mynteye/
↪right/image_raw
```

View parameters' meanings by kalibr_calibrate_cameras command:

```
$ kalibr_calibrate_cameras --h

Calibrate the intrinsics and extrinsics of a camera system with non-shared
overlapping field of view.

usage:
  Example usage to calibrate a camera system with two cameras using an aprilgrid.

  cam0: omnidirection model with radial-tangential distortion
  cam1: pinhole model with equidistant distortion

  kalibr_calibrate_cameras --models omni-radtan pinhole-equi --target aprilgrid.yaml \
            --bag MYROSBAG.bag --topics /cam0/image_raw /cam1/image_raw

  example aprilgrid.yaml:
      target_type: 'aprilgrid'
      tagCols: 6
      tagRows: 6
      tagSize: 0.088  #m
      tagSpacing: 0.3 #percent of tagSize

optional arguments:
-h, --help            show this help message and exit
--models MODELS [MODELS ...]
                      The camera model ['pinhole-radtan', 'pinhole-equi',
                      'omni-radtan', 'pinhole-fov'] to estimate

Data source:
--bag BAGFILE         The bag file with the data
--topics TOPICS [TOPICS ...]
                      The list of image topics
--bag-from-to bag_from_to bag_from_to
                      Use the bag data starting from up to this time [s]

Calibration target configuration:
--target TARGETYAML   Calibration target configuration as yaml file
```

(continues on next page)

(continued from previous page)

```
Image synchronization:
--approx-sync MAX_DELTA_APPROXSYNC
                      Time tolerance for approximate image synchronization
                      [s] (default: 0.02)

Calibrator settings:
--qr-tol QRTOL        The tolerance on the factors of the QR decomposition
                      (default: 0.02)
--mi-tol MITOL        The tolerance on the mutual information for adding an
                      image. Higher means fewer images will be added. Use -1
                      to force all images. (default: 0.2)
--no-shuffle          Do not shuffle the dataset processing order

Outlier filtering options:
--no-outliers-removal
                      Disable corner outlier filtering
--no-final-filtering  Disable filtering after all views have been processed.
--min-views-outlier MINVIEWOUTLIER
                      Number of raw views to initialize statistics (default:
                      20)
--use-blakezisserman  Enable the Blake-Zisserman m-estimator
--plot-outliers       Plot the detect outliers during extraction (this could
                      be slow)

Output options:
--verbose             Enable (really) verbose output (disables plots)
--show-extraction     Show the calibration target extraction. (disables
                      plots)
--plot                Plot during calibration (this could be slow).
--dont-show-report    Do not show the report on screen after calibration.
```

Output the following three files after finish calibration:

- `camchain-homezhangsbagstatic_10hz.yaml`
- `report-cam-homezhangsbagstatic_10hz.pdf`
- `results-cam-homezhangsbagstatic_10hz.txt`

Tip: If you use camera parameters in Vins,it would be better to choose the pinhole-equi model or the omni-radtan model.If you use camera parameters in Maplab,please choose pinhole-equi model

9.1.4 Calibrate the pose relationship between camera and IMU coordinate system

- **Collect calibration dataas calibrate the pose relationship of camera,Kalibr supports two ways to collect data,we still use `r`**
  - Method of collecting image: fix `apilgrid` calibration board, move camera
  - Make sure that the data collected is good:the brightness of the calibration board should be appropriate,too bright or too dark can't guarantee the quality of data,meanwhile do not shake too fast to avoid blurring of the image.
  - Set the imu publishing frequency to 200Hz,image to 20Hz(recommended by kalibr)

  - Fully motivate each axis of the imu,for example ,3 actions on each axis,then in the "8-shaped" motion

- Record camera and imu as `dynamic.bag`.

```
$ roslaunch mynt_eye_ros_wrapper display.launch
$ cd bag
$ rosbag record -O dynamic /mynteye/left/image_raw /mynteye/right/image_raw /mynteye/
↪imu/data_raw #remember set image hz to 20hz, imu hz to 200hz
```

- kalibr calibration:

```
$ kalibr_calibrate_imu_camera --cam camchain-homezhangsbagstatic_10hz.yaml --target␣
↪aprilgrid.yaml --imu imu.yaml --time-calibration--bag ~/bag/dynamic.bag
```

View the parameters' meanings by kalibr_calibrate_imu_camera command

```
$ kalibr_calibrate_imu_camera --h

Calibrate the spatial and temporal parameters of an IMU to a camera chain.

usage:
    Example usage to calibrate a camera system against an IMU using an aprilgrid
    with temporal calibration enabled.

    kalibr_calibrate_imu_camera --bag MYROSBAG.bag --cam camchain.yaml --imu imu.yaml␣
↪\
            --target aprilgrid.yaml --time-calibration

    camchain.yaml: is the camera-system calibration output of the multiple-camera
                   calibratin tool (kalibr_calibrate_cameras)

    example aprilgrid.yaml:       |  example imu.yaml: (ADIS16448)
        target_type: 'aprilgrid'  |      accelerometer_noise_density: 0.006
        tagCols: 6                |      accelerometer_random_walk: 0.0002
        tagRows: 6                |      gyroscope_noise_density: 0.0004
        tagSize: 0.088            |      gyroscope_random_walk: 4.0e-06
        tagSpacing: 0.3           |      update_rate: 200.0

optional arguments:
  -h, --help            show this help message and exit

Dataset source:
  --bag BAGFILE         Ros bag file containing image and imu data (rostopics
                        specified in the yamls)
  --bag-from-to bag_from_to bag_from_to
                        Use the bag data starting from up to this time [s]
  --perform-synchronization
                        Perform a clock synchronization according to 'Clock
                        synchronization algorithms for network measurements'
                        by Zhang et al. (2002).

Camera system configuration:
  --cams CHAIN_YAML     Camera system configuration as yaml file
  --recompute-camera-chain-extrinsics
                        Recompute the camera chain extrinsics. This option is
                        exclusively recommended for debugging purposes in
                        order to identify problems with the camera chain
                        extrinsics.
```

(continues on next page)

(continued from previous page)

```
  --reprojection-sigma REPROJECTION_SIGMA
                        Standard deviation of the distribution of reprojected
                        corner points [px]. (default: 1.0)

IMU configuration:
  --imu IMU_YAMLS [IMU_YAMLS ...]
                        Yaml files holding the IMU noise parameters. The first
                        IMU will be the reference IMU.
  --imu-delay-by-correlation
                        Estimate the delay between multiple IMUs by
                        correlation. By default, no temporal calibration
                        between IMUs will be performed.
  --imu-models IMU_MODELS [IMU_MODELS ...]
                        The IMU models to estimate. Currently supported are
                        'calibrated', 'scale-misalignment' and 'scale-
                        misalignment-size-effect'.

Calibration target:
  --target TARGET_YAML  Calibration target configuration as yaml file

Optimization options:
  --time-calibration    Enable the temporal calibration
  --max-iter MAX_ITER   Max. iterations (default: 30)
  --recover-covariance  Recover the covariance of the design variables.
  --timeoffset-padding TIMEOFFSET_PADDING
                        Maximum range in which the timeoffset may change
                        during estimation [s] (default: 0.01)

Output options:
  --show-extraction     Show the calibration target extraction. (disables
                        plots)
  --extraction-stepping
                        Show each image during calibration target extraction
                        (disables plots)
  --verbose             Verbose output (disables plots)
  --dont-show-report    Do not show the report on screen after calibration.
```

Output the follwing 4 files after finish calibration:

- `camchain-imucam-homezhangsbagdynamic.yaml`
- `imu-homezhangsbagdynamatic.yaml`
- `report-imucam-homezhangsbagdynamic.pdf`
- `results-imucam-homezhangsbagdynamic.yaml`

9.2 How to use in VINS-Mono

9.2.1 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. Update `distortion_parameters` and `projection_parameters` to here .

4. Run mynt_eye_ros_wrapper and VINS-Mono.

9.2.2 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
```

9.2.3 Install MYNT-EYE-VINS-Sample

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

9.2.4 Get image calibration parameters

Use MYNT® EYE's left eye camera and IMU. By MYNT-EYE-S-SDK API `GetIntrinsics()` function and `GetExtrinsics()` function, you can "get the image calibration parameters of the current working 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 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 image parameters* to write a default pinhole config file to your device.

9.2.5 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 mynteye.launch
```

2. Open another terminal and run vins

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

Note: If you want to use a fish-eye camera model, please click here .

9.3 How to use in VINS-Fusion

9.3.1 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.

9.3.2 Prerequisites

1. Install Ubuntu 64-bit 16.04 or 18.04. ROS Kinetic or Melodic.(if already installed, please ignore). ROS Installation
2. Install Ceres

9.3.3 Install MYNT-EYE-VINS-FUSION-Samples

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

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

9.3.4 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 mynteye.launch
```

2. Open another terminal and run vins

```
cd ~/catkin_ws
roslaunch vins mynteye-s-mono-imu.launch  # mono+imu fusion
# roslaunch vins mynteye-s-stereo.launch  # Stereo fusion / Stereo+imu fusion
# roslaunch vins mynteye-avarta-mono-imu.launch  # mono+imu fusion with mynteye-avarta
# roslaunch vins mynteye-avarta-stereo.launch  # Stereo fusion / Stereo+imu fusion␣
↪with mynteye-avarta
```

9.4 How to use in ORB_SLAM2

9.4.1 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. Update `distortion_parameters` and `projection_parameters` to `<ORB_SLAM2>/config/mynteye_*.yaml`.
4. Run examples by MYNT® EYE.

9.4.2 Binocular camera sample

- Calibrate a stereo camera with ROS-StereoCalibration or OpenCV, and then update parameters to `<ORB_SLAM2>/config/mynteye_s_stereo.yaml`.
- Execute `build.sh`:

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

- Run stereo sample using the follow type:

```
./Examples/Stereo/stereo_mynt_s ./Vocabulary/ORBvoc.txt ./config/mynteye_s_stereo.
↪yaml true /mynteye/left/image_raw /mynteye/right/image_raw
```

9.4.3 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. Replace `PATH` by the folder where you cloned ORB_SLAM2:

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

- Execute *build_ros.sh*:

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

Stereo_ROS Example

- Reference `Get camera calibration parameters` in *How to use in OKVIS* to get `distortion_parameters` and `projection_parameters` , and update `<ORB_SLAM2>/config/mynteye_s_stereo.yaml` .
- 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 true /mynteye/left/image_raw /mynteye/right/image_raw
```

9.5 How to use in OKVIS

9.5.1 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.

9.5.2 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
```

9.5.3 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 image parameters* 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]
principal_point: [cx, cy]
distortion_type: radialtangential
```

9.5.4 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
```

9.6 How to use in VIORB

9.6.1 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.

9.6.2 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
```

9.6.3 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 image parameters* to write a default pinhole config file to your device.

9.6.4 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.

9.7 How to use in Maplab x

CHAPTER 10

API DOC

10.1 API

10.1.1 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**

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 **setDuplicate**(bool *isEnable*)
Set if the duplicate frames is enable.

void **SetOptionValue**(**const** *Option* &*option*, std::int32_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 **EnableTimestampCorrespondence**(**const** *Stream* &*stream*, bool *keep_accel_then_gyro* = true)
Enable motion datas with timestamp correspondence of some stream.

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

Public Static Functions

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

Return 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.

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

Return the *API* instance.

Note This will init glog with args.

Parameters

- `argc`: the arg count.
- `argv`: the arg values.
- `device`: the selected device.

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

Return the *API* instance.

Parameters

- `device`: the selected device.

10.1.2 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.

10.1.3 api::MotionData

struct MotionData
API motion data.

Public Members

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

10.2 Device

10.2.1 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

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 **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*)

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 **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.

Return the *Device* instance.

Parameters

- `name`: the device name.
- `device`: the device from uvc.

10.2.2 device::Frame

class Frame
Frame with raw data.

Public Functions

Frame(**const** *StreamRequest* &*request*, **const** void **data*)
Construct the frame with *StreamRequest* and raw data.

Frame(std::uint16_t *width*, std::uint16_t *height*, *Format* *format*, **const** void **data*)
Construct the frame with stream info and raw data.

std::uint16_t **width**() **const**
Get the width.

std::uint16_t **height**() **const**
Get the height.

Format **format**() **const**
Get the format.

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

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

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

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

10.2.3 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.

10.2.4 device::MotionData

struct MotionData
Device motion data.

Public Members

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

10.3 Enums

10.3.1 Model

enum mynteye::**Model**

Device model.

Values:

STANDARD
Standard.

STANDARD2
Standard 2.

STANDARD210A
Standard 210a.

10.3.2 Stream

enum mynteye::**Stream**

Streams define different type of data.

Values:

LEFT
Left stream.

RIGHT
Right stream.

LEFT_RECTIFIED
Left stream, rectified.

RIGHT_RECTIFIED
Right stream, rectified.

DISPARITY
Disparity stream.

DISPARITY_NORMALIZED
Disparity stream, normalized.

DEPTH
Depth stream.

POINTS
Point cloud stream.

10.3.3 Capabilities

enum mynteye::**Capabilities**

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

Values:

STEREO
Provides stereo stream.

STEREO_COLOR
Provide stereo color stream.

COLOR
Provides color stream.

DEPTH
Provides depth stream.

POINTS
Provides point cloud stream.

FISHEYE
Provides fisheye stream.

INFRARED
Provides infrared stream.

INFRARED2
Provides second infrared stream.

IMU
Provides IMU (accelerometer, gyroscope) data.

10.3.4 Info

enum mynteye::**Info**
Camera info fields are read-only strings that can be queried from the device.

Values:

DEVICE_NAME
Device name.

SERIAL_NUMBER
Serial number.

FIRMWARE_VERSION
Firmware version.

HARDWARE_VERSION
Hardware version.

SPEC_VERSION
Spec version.

LENS_TYPE
Lens type.

IMU_TYPE
IMU type.

NOMINAL_BASELINE
Nominal baseline.

AUXILIARY_CHIP_VERSION
Auxiliary chip version.

ISP_VERSION
Isp version.

10.3.5 Option

enum mynteye::**Option**

Camera control options define general configuration controls.

Values:

GAIN

Image gain, valid if manual-exposure.

range: [0,48], default: 24

BRIGHTNESS

Image brightness, valid if manual-exposure.

range: [0,240], default: 120

CONTRAST

Image contrast, valid if manual-exposure.

range: [0,255], default: 127

FRAME_RATE

Image frame rate, must set IMU_FREQUENCY together.

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

IMU_FREQUENCY

IMU frequency, must set FRAME_RATE together.

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

EXPOSURE_MODE

Exposure mode.

0: enable auto-exposure 1: disable auto-exposure (manual-exposure)

MAX_GAIN

Max gain, valid if auto-exposure.

range of standard 1: [0,48], default: 48 range of standard 2: [0,255], default: 8

MAX_EXPOSURE_TIME

Max exposure time, valid if auto-exposure.

range of standard 1: [0,240], default: 240 range of standard 2: [0,1000], default: 333

MIN_EXPOSURE_TIME

min exposure time, valid if auto-exposure

range: [0,1000], default: 0

DESIRED_BRIGHTNESS

Desired brightness, valid if auto-exposure.

range of standard 1: [0,255], default: 192 range of standard 2: [1,255], default: 122

IR_CONTROL

IR control.

range: [0,160], default: 0

HDR_MODE

HDR mode.

0: 10-bit 1: 12-bit

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

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

ACCELEROMETER_LOW_PASS_FILTER
The parameter of accelerometer low pass filter.

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

GYROSCOPE_LOW_PASS_FILTER
The parameter of gyroscope low pass filter.

values: {23,64}, default: 64

ZERO_DRIFT_CALIBRATION
Zero drift calibration.

ERASE_CHIP
Erase chip.

10.3.6 Source

enum mynteye::**Source**
Source allows the user to choose which data to be captured.

Values:

VIDEO_STREAMING
Video streaming of stereo, color, depth, etc.

MOTION_TRACKING
Motion tracking of IMU (accelerometer, gyroscope)

ALL
Enable everything together.

10.3.7 AddOns

enum mynteye::**AddOns**
Add-Ons are peripheral modules of our hardware.

Values:

INFRARED
Infrared.

INFRARED2
Second infrared.

10.3.8 Format

enum mynteye::**Format**

Formats define how each stream can be encoded.

Values:

GREY = ((std::uint32_t)('G') | ((std::uint32_t)('R') << 8) | ((std::uint32_t)('E') << 16) | ((std::uint32_t)('Y') << 24))
Greyscale, 8 bits per pixel.

YUYV = ((std::uint32_t)('Y') | ((std::uint32_t)('U') << 8) | ((std::uint32_t)('Y') << 16) | ((std::uint32_t)('V') << 24))
YUV 4:2:2, 16 bits per pixel.

BGR888 = ((std::uint32_t)('B') | ((std::uint32_t)('G') << 8) | ((std::uint32_t)('R') << 16) | ((std::uint32_t)('3') << 24))
BGR 8:8:8, 24 bits per pixel.

RGB888 = ((std::uint32_t)('R') | ((std::uint32_t)('G') << 8) | ((std::uint32_t)('B') << 16) | ((std::uint32_t)('3') << 24))
RGB 8:8:8, 24 bits per pixel.

10.3.9 CalibrationModel

enum mynteye::**CalibrationModel**

Camera calibration model.

Values:

PINHOLE = 0
Pinhole.

KANNALA_BRANDT = 1
Equidistant: KANNALA_BRANDT.

UNKNOW
Unknow.

10.3.10 DisparityComputingMethod

enum mynteye::**DisparityComputingMethod**

Camera disparity computing method type.

Values:

SGBM = 0
bm

BM = 1
sgbm

UNKNOW
unknow

10.4 Types

10.4.1 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.

10.4.2 Resolution

struct Resolution
Resolution.

Public Members

std::uint16_t **width**
Width.

std::uint16_t **height**
Height.

10.4.3 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.

10.4.4 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**
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 **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.

10.4.5 Extrinsics

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

Public Functions

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

Return the inversed extrinsics.

Public Members

double **rotation**[3][3]
Rotation matrix.

double **translation**[3]
Translation vector.

10.4.6 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].

10.4.7 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

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.

10.5 Utils

10.5.1 select

std::shared_ptr<*Device*> mynteye::device::**select**()
Detecting MYNT EYE devices and prompt user to select one.

Return the selected device, or nullptr if none.

10.5.2 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.

Return the selected request.

10.5.3 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.

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

Parameters

- `frame_rate`: the frame rate of the device.
- `exposure_time`: the virtual exposure time.

10.5.4 get_sdk_root_dir

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

Get sdk root dir.

10.5.5 get_sdk_install_dir

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

Get sdk install dir.

Index

M