MYNT® EYE S SDK Guide

Release 2.2.2

JohnZhao

Dec 16, 2018

Contents

CHAPTER 1

MYNT® EYE

1.1 Product description

The MYNT® EYE Standard Edition utilizes the camera and the motion sensor to provide visually accurate SLAM results with higher precision, lower cost, simpler layout, along with the ability to achieve face and object recognition. The concept of combining binocular and IMU is the leading-edge technology in the current SLAM industry.

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

Using camera techniques such as frame synchronization, automatic exposure, and white balance control, the MYNT® EYE Standard Edition can produce synchronized image sources with high-precision, which decreases the difficulty of algorithms development, thus increasing efficiency. The standard version comes with six-axis sensor(IMU) and an infrared active light detector (IR). Among them, the six-axis sensor(IMU) can provide complementarity and correction of data from the visual positioning algorithms, and is suitable for visual inertial odometry(VIO) algorithms research to help improve the positioning accuracy. The infrared active light detector (IR) can help solve the problem of identification of objects such as indoor white walls and non-textured objects, as well as enhance the accuracy of image source recognition.

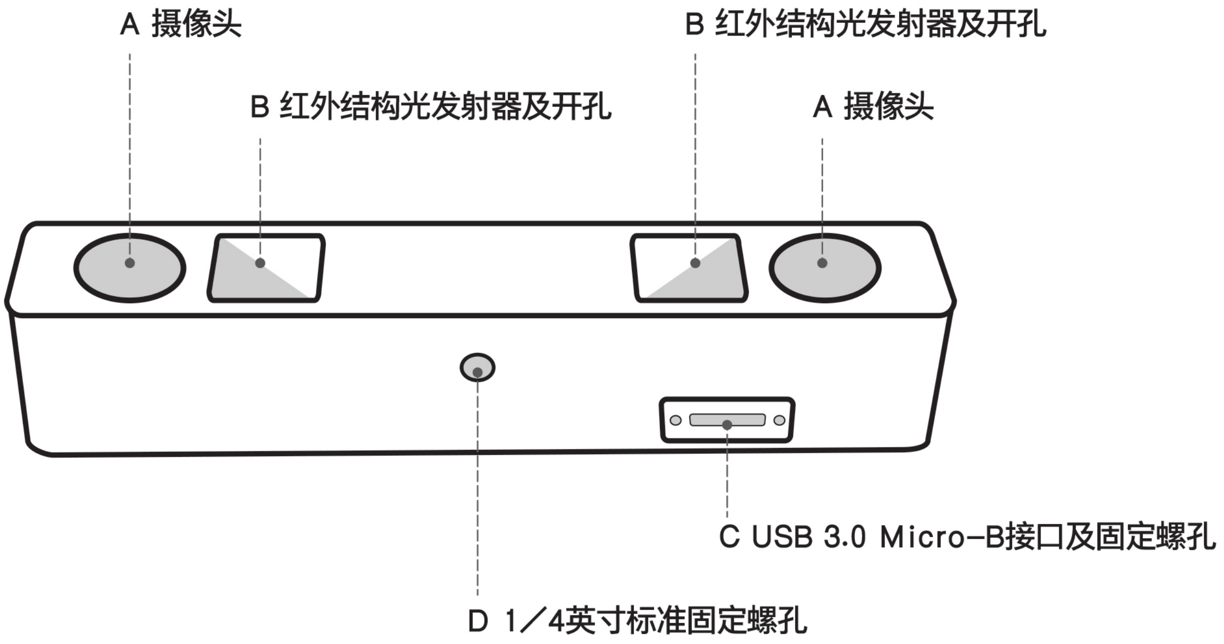
In order to ensure the quality of the output data of the camera products, we have calibrated the binocular and IMU. The product has passed various hardware stability tests, such as high temperature and humidity continuous work and operation, low-temperature dynamic aging, high-temperature operation,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

Since its first appearance, the product has been recognized by many well-known companies in and outside of China. In addition to the standard “S” series that has already been released, a new “D” series product with built-in deep computing power will also be launched in the future. Customers can select the appropriate product based on factors such as the computing power and operational architecture of the target platform.

With the goal of becoming the world's leading provider of stereo vision computing technology solutions, we have been continuously optimizing and perfecting hardware and software, making MYNT® EYE binocular camera the best stereo vision technology development and application platform.

1.2 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.

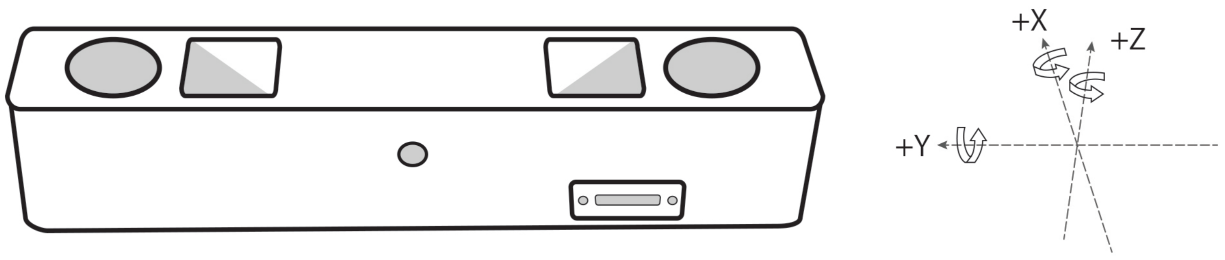
B. Infrared structured-light transmitter and outlet: the infrared structured-light can effectively solve the problem associated with the visual positioning calculations of white wall non-textured object(For non-IR version, the outlet is reserved but there is no internal structured-light emitter).

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. $\frac{1}{4}$ inch standardized set screw hole: for fixing the stereo camera to tripods or other devices.

1.3 IMU coordinata system

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


+X
+Z
+Y

CHAPTER 2

MYNT® EYE SDK

2.1 Supported platforms

SDK is built on CMake and can be used cross multiple platforms such as Linux, Windows, macOS, 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 16.04 / 14.04
- Jetson TX2

Tip: Do not support ppa installation on ubuntu 14.04. Support compile and install from source code only.

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.

2.2 Ubuntu SDK PPA Installation

Ubuntu 16.04
✓

2.2.1 PPA installation

```
$ sudo add-apt-repository ppa:slightech/mynteye2
$ sudo apt-get update
$ sudo apt-get install mynteye2
```

2.2.2 Run samples

Tip: samples path: /opt/mynteye/samples; tools path: /opt/mynteye/tools

```
$ cd /opt/mynteye/samples
$ ./api/camera_a
```

2.3 Windows SDK exe Installation

Windows 10
✓

2.3.1 Download and install SDK

Tip: Download here: mynteye-s-2.2.2-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.

2.3.2 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.

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

2.3.3 Start using SDK with Visual Studio 2017

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

2.4 Ubuntu SDK Source Installation

Ubuntu 16.04	Ubuntu 14.04
✓	✓

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

2.4.1 Getting Source Code

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

2.4.2 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
```

2.4.3 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 为 OpenCVConfig.cmake 所在目录
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 `<sdk>/_install` by default.

2.4.4 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* .

2.4.5 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.

2.4.6 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.

2.5 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.

2.5.1 Prerequisites

CMake (provide build)

- CMake， used 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
```

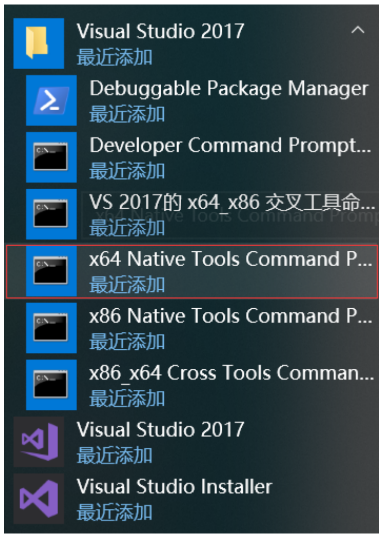
Visual Studio (provide compilation)

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

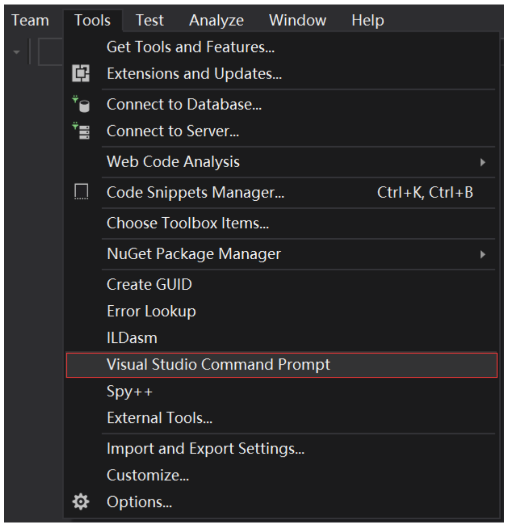
After installing Visual Studio, confirm that the following command can run in the Visual Studio Command Prompt:

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

>msbuild
Microsoft (R) 生成引擎版本 15.7.179.6572
```

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


Visual Studio 2017
最近添加
Debuggable Package Manager
最近添加
Developer Command Prompt...
最近添加
VS 2017的 x64_x86 交叉工具命...
最近添加
x64 Native Tools Command P...
最近添加
x86 Native Tools Command P...
最近添加
x86_x64 Cross Tools Comman...
最近添加
Visual Studio 2017
最近添加
Visual Studio Installer
最近添加

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

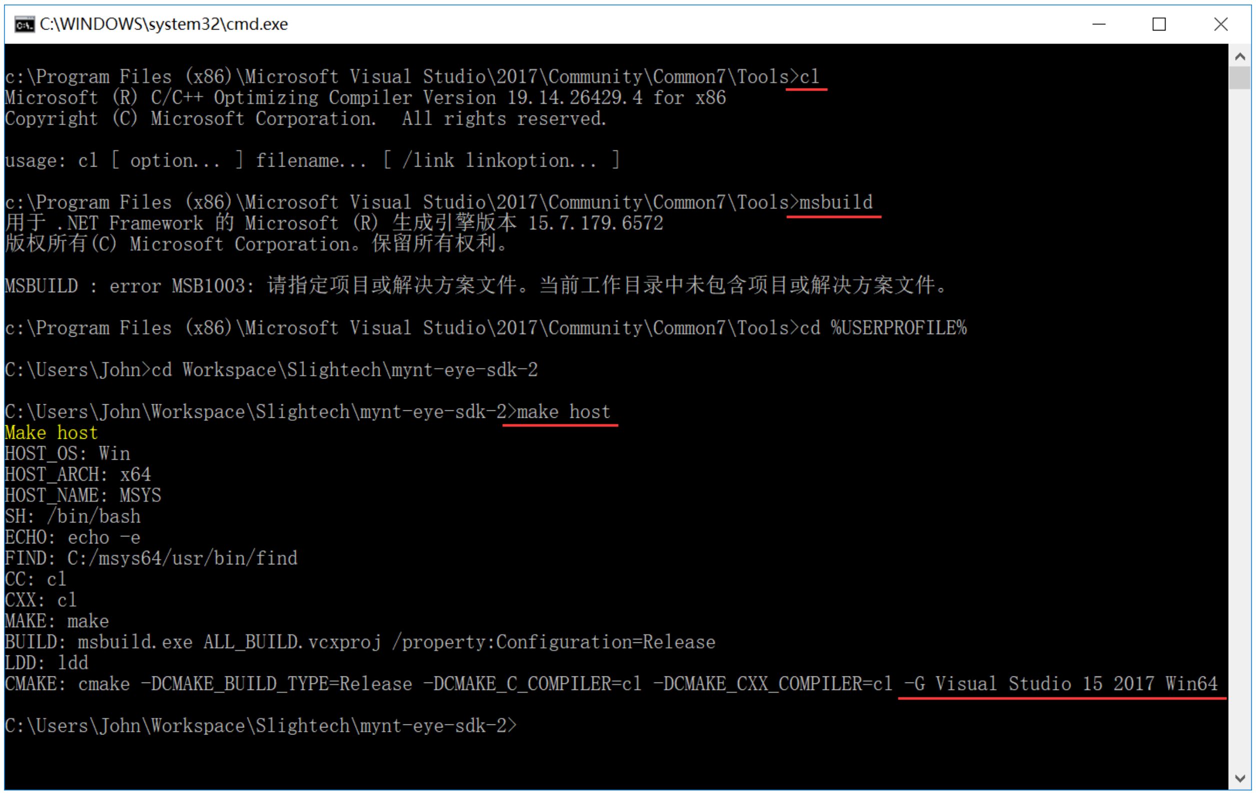


However, if you do not have the Visual Studio 2015 Tools menu, you can add one yourself.

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.



```
c:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\Tools>cl
Microsoft (R) C/C++ Optimizing Compiler Version 19.14.26429.4 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.

usage: cl [ option... ] filename... [ /link linkoption... ]

c:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\Tools>msbuild
用于 .NET Framework 的 Microsoft (R) 生成引擎版本 15.7.179.6572
版权所有(C) Microsoft Corporation。保留所有权利。

MSBUILD : error MSB1003: 请指定项目或解决方案文件。当前工作目录中未包含项目或解决方案文件。

c:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\Tools>cd %USERPROFILE%

C:\Users\John>cd Workspace\Slightech\mynt-eye-sdk-2

C:\Users\John\Workspace\Slightech\mynt-eye-sdk-2>make host
Make host
HOST_OS: Win
HOST_ARCH: x64
HOST_NAME: MSYS
SH: /bin/bash
ECHO: echo -e
FIND: C:/msys64/usr/bin/find
CC: cl
CXX: cl
MAKE: make
BUILD: msbuild.exe ALL_BUILD.vcxproj /property:Configuration=Release
LDD: ldd
CMAKE: cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_C_COMPILER=cl -DCMAKE_CXX_COMPILER=cl -G Visual Studio 15 2017 Win64

C:\Users\John\Workspace\Slightech\mynt-eye-sdk-2>
```

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
```

2.5.2 Getting Source Code

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

2.5.3 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
```

2.5.4 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 为 OpenCVConfig.cmake 所在目录
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.

2.5.5 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 .

2.5.6 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.

2.5.7 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.

2.6 MacOS Installation x

TODO

2.7 ROS Installation

ROS Kinetic	ROS Indigo
✓	✓

2.7.1 Prepare Environment

- ROS

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
```

2.7.2 Compiling Code

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

2.7.3 Running node

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

Run the node, and preview by RViz:

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

2.7.4 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
```

2.7.5 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

2.7.6 Conclusion

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

2.8 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

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

`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 MYNTEYE_BOX . The file structure is as follows:

```
Firmwares/
├─Checksum.txt                 # file checksum
├─MYNTEYE_S_2.0.0_rc0.img      # 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

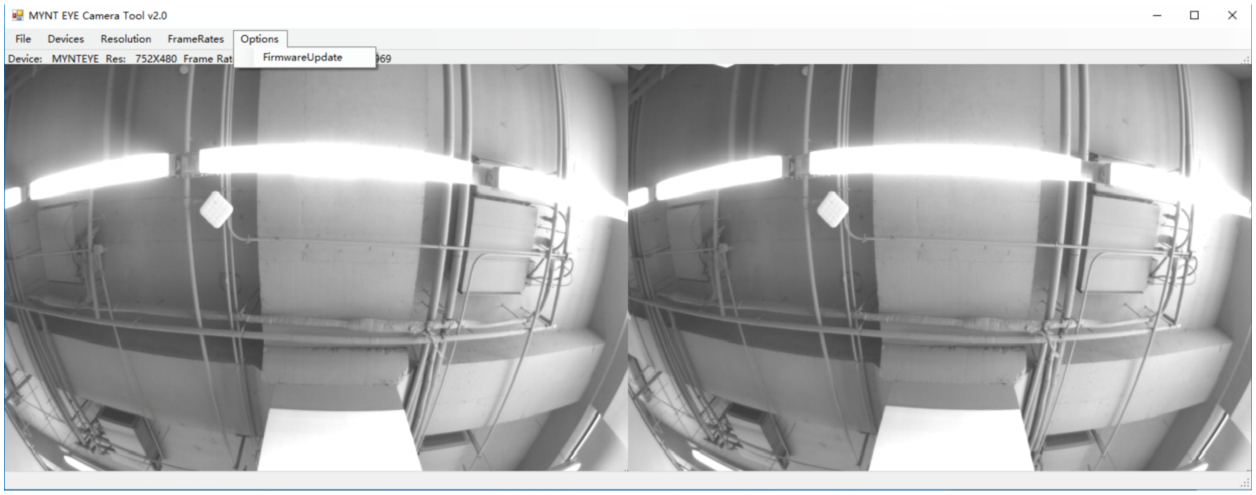
- Download and unzip `setup.zip`
- Download firmware, such as `MYNTEYE_S_2.0.0_rc0.img`
  - Please refer to *Firmware and SDK compatibility* to select the firmware suitable for the SDK version
  - Please refer to `Checksum.txt` to find the firmware check code as follows:
    * Run the command in CMD `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

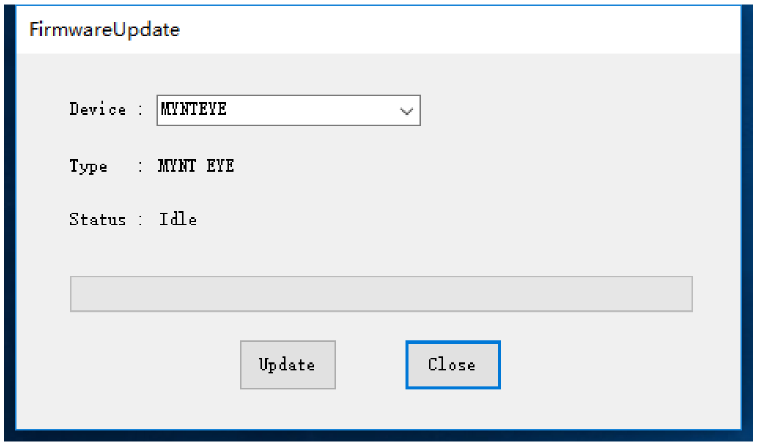
- Double click on `setup.msi` and install the application.

3.2.3 Update Firmware

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


MYNT EYE Camera Tool v2.0
File
Devices
Resolution
FrameRates
Options
FirmwareUpdate
Device: MYNTEYE Res: 752X480 Frame Rat
69

- Click `Update` .

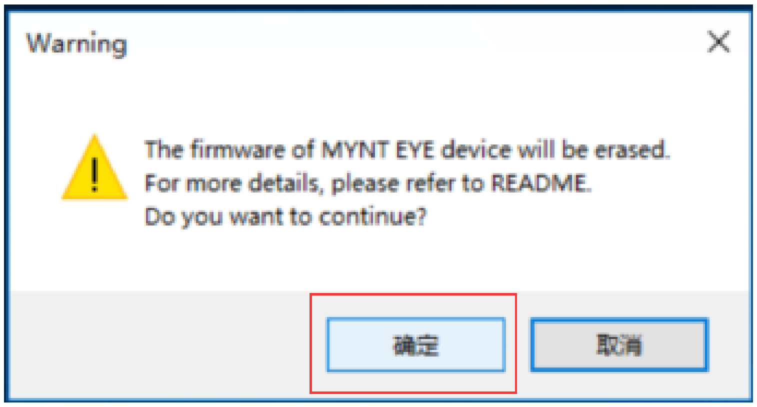


Device : MYNTEYE

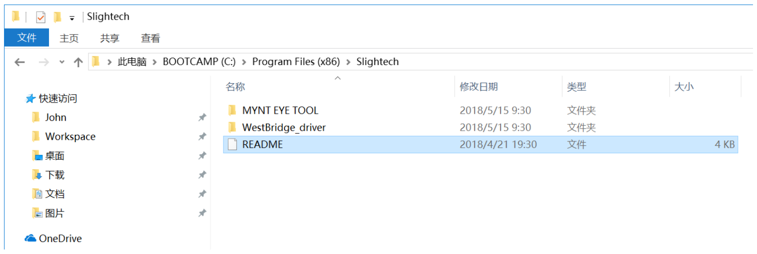
Type : MYNT EYE

Status : Idle

- 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.

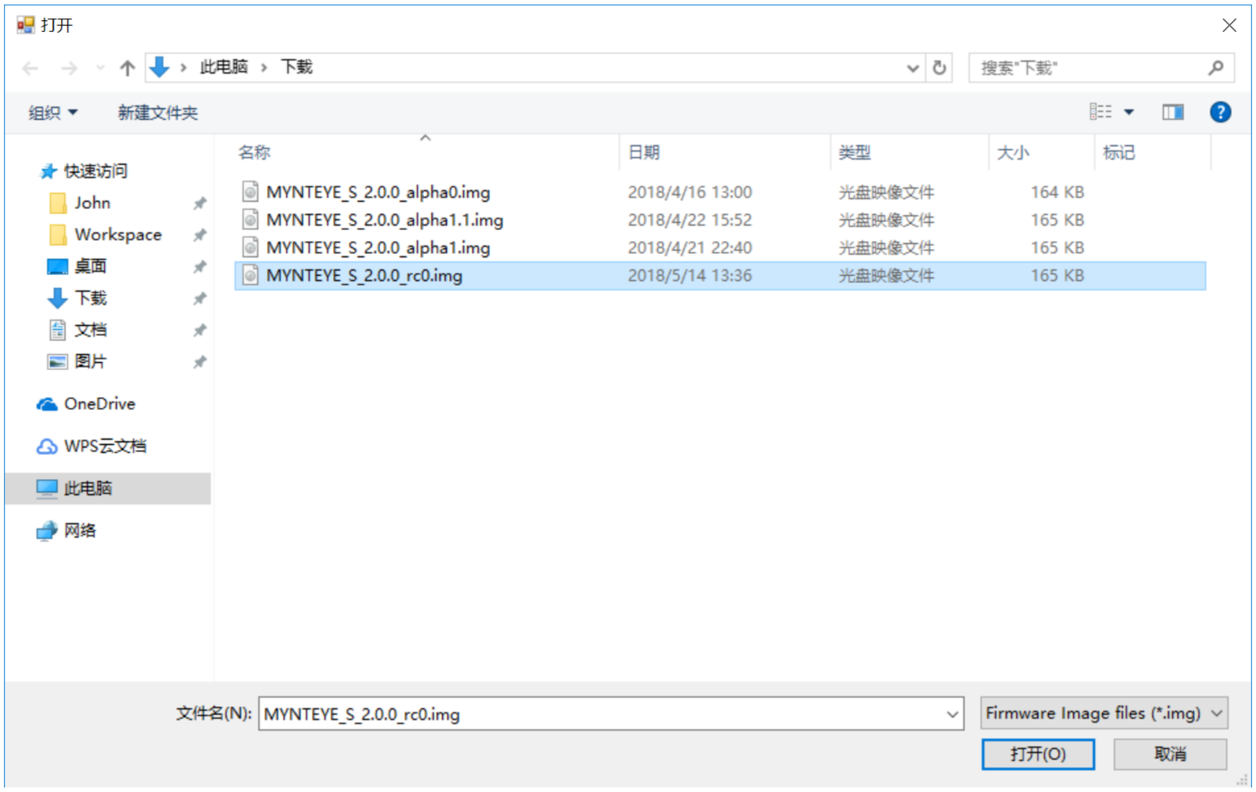


The firmware of MYNT EYE device will be erased.
For more details, please refer to README.
Do you want to continue?



名称	修改日期	类型	大小
MYNT EYE TOOL	2018/5/15 9:30	文件夹	
WestBridge_driver	2018/5/15 9:30	文件夹	
README	2018/4/21 19:30	文件	4 KB

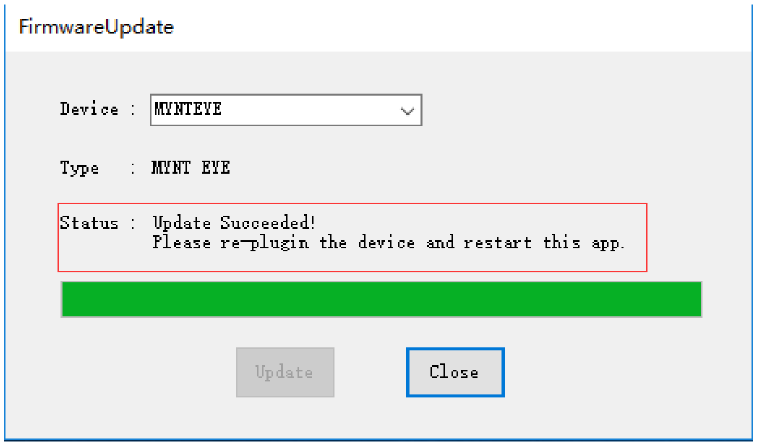
- In the open file selection box, select the firmware you want to upgrade and start upgrading.



名称	日期	类型	大小	标记
MYNTEYE_S_2.0.0_alpha0.img	2018/4/16 13:00	光盘映像文件	164 KB	
MYNTEYE_S_2.0.0_alpha1.1.img	2018/4/22 15:52	光盘映像文件	165 KB	
MYNTEYE_S_2.0.0_alpha1.img	2018/4/21 22:40	光盘映像文件	165 KB	
MYNTEYE_S_2.0.0_rc0.img	2018/5/14 13:36	光盘映像文件	165 KB	

文件名(N): MYNTEYE_S_2.0.0_rc0.img
Firmware Image files (*.img)

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



```
Device :  MYNTEYE
Type   :  MYNT EYE
Status :  Update Succeeded!
          Please re-plugin the device and restart this app.
```

- Close the MYNT EYE TOOL， finish.

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 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 version，Check 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.

Reference code snippet:

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

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

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() << "}";
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:

```cpp
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* .

Reference code snippet:

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

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

api->Start(Source::VIDEO_STREAMING);

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

(continues on next page)

(continued from previous page)

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

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

  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()`. 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();
```

(continues on next page)

(continued from previous page)

```
  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 .

Tip: 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);

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

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

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

(continues on next page)

(continued from previous page)

```
  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();

api->Start(Source::ALL);

CVPainter painter;

cv::namedWindow("frame");

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

(continues on next page)

(continued from previous page)

```
  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 data from callbacks

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

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

Reference code snippet:

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

// Attention: must not block the callbacks.
```

(continues on next page)

(continued from previous page)

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

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

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

api->Start(Source::ALL);

CVPainter painter;

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

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

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

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

(continues on next page)

(continued from previous page)

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

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

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

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

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

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

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

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

Complete code examples, see get_from_callbacks.cc .

4.11 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
```

(continues on next page)

(continued from previous page)

```
│   ├─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 指依赖库所在路径

# macOS
export DYLD_LIBRARY_PATH=/usr/local/lib:$DYLD_LIBRARY_PATH
# /usr/local/lib 指依赖库所在路径

# Windows
set PATH=C:\opencv\x64\vc14\bin;%PATH%
# 或者，添加进系统环境变量 Path 里。
```

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 指具体插件路径

# macOS
otool -L *.dylib
# *.dylib 指具体插件路径

# Windows
# 请下载如 Dependency Walker ，打开 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
```

4.12 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
```

4.13 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
```

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

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

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

4.14 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 values for the current device.

To set the image frame rate and IMU frequency, set `Option::FRAME_RATE` and `Option::IMU_FREQUENCY`.

Attention:

- The frame rate of image & IMU frequency must be set at the same time to take effect.
- The effective fps of the image: 10, 15, 20, 25, 30, 35, 40, 45, 50, 55.
- The effective frequency of IMU: 100, 200, 250, 333, 500

Reference Code:

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

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

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

Reference running results on Linux:

```
$ ./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
```

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 samples, please 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:

- The effective range of accelerometer(unit:g): 4, 8, 16, 32.
- The effective range of gyroscope(unit:deg/s): 500, 1000, 2000, 4000.

Reference Code:

```
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);
```

Reference running results on Linux:

```
$ ./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
```

(continues on next page)

(continued from previous page)

```
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
```

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 samples， please 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` . 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.

Reference Code:

```
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);
```

Reference running results on Linux:

```
$ ./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
```

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`. During manual exposure, the settings available for adjustment are:

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

Reference Code:

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

Reference running results on Linux:

```
$ ./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
```

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

Complete code samples， see 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.

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.

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 samples， see infrared.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* .

`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
  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.

```
<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/record

# Windows
.\tools\_output\bin\dataset\record.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  # Image, index 0
   │  └─...
   ├─right/
   │  ├─stream.txt  # Image information
   │  ├─000000.png  # Image, index 0
   │  └─...
   └─motion.txt  # IMU information
```

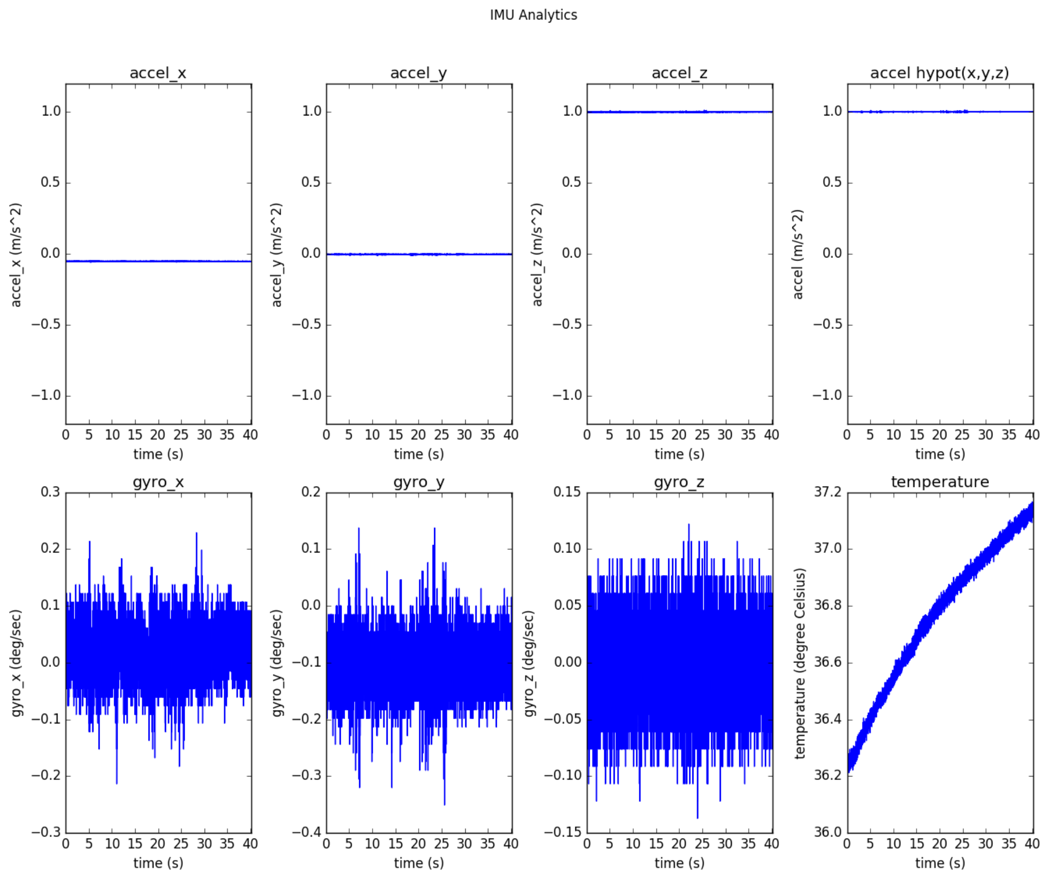
8.2 Analyzing IMU

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:

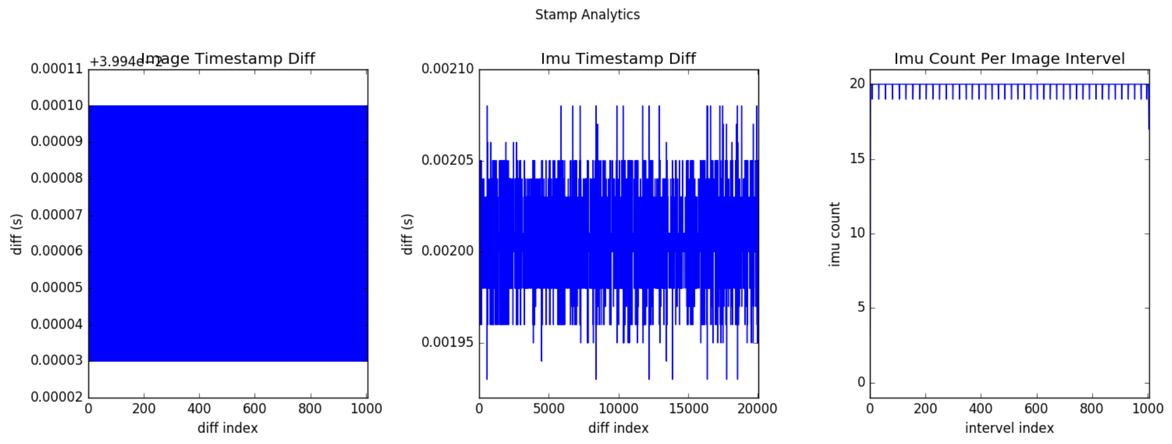
```
$ python tools/analytics/stamp_analytics.py -i dataset -c tools/config/mynteye/
↪mynteye_config.yaml
stamp analytics ...
  input: dataset
  outdir: dataset
open dataset ...
save to binary files ...
  binimg: dataset/stamp_analytics_img.bin
  binimu: dataset/stamp_analytics_imu.bin
  img: 1007, imu: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

(continues on next page)

(continued from previous page)

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

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

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

image timestamp duplicates: 0

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

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

SLAM

9.1 How to use in VINS-Mono

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

1. Download MYNT-EYE-S-SDK and install mynt_eye_ros_wrapper.
2. Follow the normal procedure to install VINS-Mono.
3. Update `distortion_parameters` and `projection_parameters` to here .
4. Run mynt_eye_ros_wrapper and VINS-Mono.

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

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

9.1.3 Install MYNT-EYE-VINS-Sample

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
git clone -b mynteye-s 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
```

9.1.4 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 .

9.1.5 Run VINS-Mono with MYNT® EYE

```
cd ~/catkin_ws
roslaunch mynt_eye_ros_wrapper mynteye.launch
roslaunch vins_estimator mynteye.launch
```

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

9.2 How to use in ORB_SLAM2

9.2.1 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.

9.2.2 Binocular camera sample

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

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

- Run stereo sample using the follow type:

```
./Examples/Stereo/stereo_mynt ./Vocabulary/ORBvoc.txt ./config/mynteye_stereo.yaml␣
↪true /mynteye/left/image_raw /mynteye/right/image_raw
```

9.2.3 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
```

Mono_ROS Example

- Update `distortion_parameters` and `projection_parameters` in `<ORBSLAM2>/config/mynteye_mono.yaml`

```
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 `<ORB_SLAM2>/config/mynteye_mono.yaml`.

- Launch ORB_SLAM2 `Mono_ROS`

```
rosrun ORB_SLAM2 mynteye_mono ./Vocabulary/ORBvoc.txt ./config/mynteye_mono.yaml /
↪mynteye/left/image_raw
```

Stereo_ROS Example

- Calibrate a stereo camera with ROS-StereoCalibration or OpenCV, and then update parameters to `<ORB_SLAM2>/config/mynteye_stereo.yaml`.
- Launch ORB_SLAM2 `Stereo_ROS`

```
rosrun ORB_SLAM2 ros_mynteye_stereo ./Vocabulary/ORBvoc.txt ./config/mynteye_stereo.
↪yaml true /mynteye/left/image_raw /mynteye/right/image_raw
```

9.3 How to use in OKVIS

9.3.1 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.

9.3.2 Install MYNTEYE OKVIS

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

```
git clone -b mynteye-s https://github.com/slightech/MYNT-EYE-OKVIS-Sample.git
mkdir build && cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make -j4
```

9.3.3 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, then update to here .

9.3.4 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.yaml
```

9.4 How to use in VIORB

9.4.1 If you wanna run VIORB with MYNT® EYE，please 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/config_mynteye.yaml`.
4. Run mynt_eye_ros_wrapper and VIORB.

9.4.2 安装 MYNT-EYE-VIORB-Sample.

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

Add the path including `Examples/ROS/ORB_VIO` 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 `MYNT-EYE-VIORB-Sample`:

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

Execute:

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

9.4.3 Get image calibration parameters

Assume that the left eye of the mynteye camera is used with IMU. Through the `GetIntrinsics()` and `GetExtrinsics()` function of the MYNT-EYE-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`.

9.4.4 Run VIORB and mynt_eye_ros_wrapper

```
roslaunch mynt_eye_ros_wrapper mynteye.launch
roslaunch ORB_VIO testmynteye.launch
```

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

9.5 How to use in Maplab x