MYNT EYE SDK

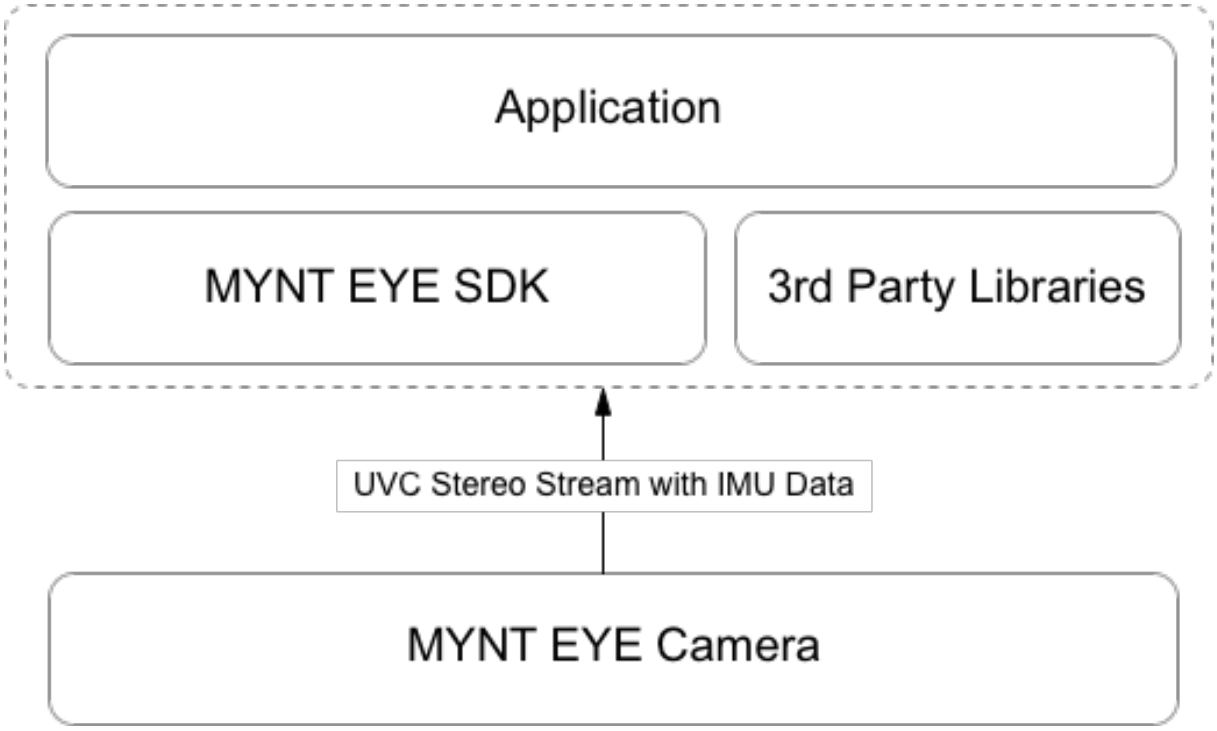
v1.5

Generated by Slightech Inc

Contents

Chapter 1

SDK Introduction



SDK Files

The SDK could be installed to a arbitrary location. Just run the install script under the SDK directory.

The SDK files falls into the following directories:

- **apps**/ contains binary executables of some MYNT EYE camera apps
  - **3rdparty**/ contains the apps' dependencies
- **bin**/ contains the windows DLLs required to use the SDK (Windows only)
- **doc**/ contains documentations and licence
  - **api**/ contains the API documentation on Web format

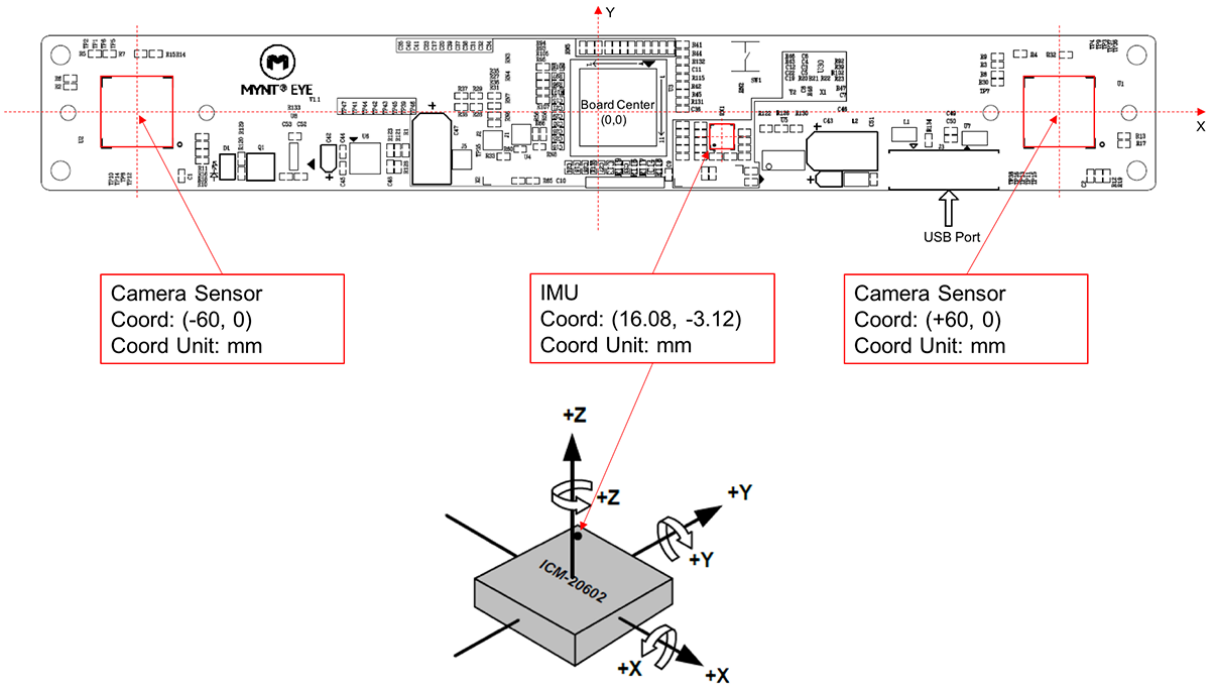
  - **license**/ contains the licenses of the SDK and its third-parties'
- **deps**/ contains samples and SDK (OpenCV) dependencies (Windows only)
- **include**/ contains C/C++ header files
- **lib**/ contains the libraries files required to link with the SDK (".so" on Linux, ".dylib" on macOS and ".lib" on Windows)
  - **3rdparty**/ contains the libraries' dependencies
- **samples**/ contains the samples' source codes about how to use SDK
- **tools**/ contains binary executables included in the SDK that help to use the MYNT EYE camera
- **install.sh** the install script to configure the SDK ("install.bat" on Windows)
- **README** the readme about how to get started with your MYNT EYE

Please read the README file or the Getting Started section to get started with application development.

IMU Introduction

IMU has 3-axis accelerometer and 3-axis gyroscope to detect linear acceleration and rotational rate.

IMU sampling rate is 250 Hz.



IMU Rotation and Translation

Right-handed coordinate system, RUB (right-up-back).

IMU Rotation matrix (rotate 90° around the z-axis):

```
[0, -1, 0,
 1,  0, 0,
 0,  0, 1]
```

Translation vector from camera to IMU (in meter):

	Translation
Camera Left	[0.07608, -0.00312, -0.01464]
Camera Right	[-0.04392, -0.00312, -0.01464]

The value of z-axis is from the center of the lens to the board. The deviation maybe $\pm$0.25mm because of the difference in focal length.

IMU Noise Density and Random Walk

	Noise Density	Random Walk
Accelerometer	7.6509e-02	5.3271e-02
Gyroscope	9.0086e-03	5.5379e-05

Chapter 2

Getting Started

1. Connect your MYNT EYE camera

Thank you for purchasing the MYNT EYE camera!

Unpack your MYNT EYE and plug the camera in a USB 3.0 port. The MYNT EYE is a UVC compliant camera so it should be automatically recognized by your computer.

2. Download and install the MYNT EYE SDK

The MYNT EYE SDK is avalible for Linux, macOS, Windows and Nvidia Jetson boards. Get the latest MYNT EYE SDK from our `downloads` page.

The following guides show you how to get started with your MYNT EYE on different platforms:

- Getting Started on Linux
- Getting Started on macOS
- Getting Started on Windows (MSVC)
- Getting Started on Tegra (TX1, TX2)

3. Run the MYNT EYE Previewer

The MYNT EYE Previewer uses the SDK to capture and display the images and depth map.

4. Play with the samples

The SDK contains some samples to show you how to use the SDK, please see "samples/README.md" to learn more.

2.1 Getting Started on Linux

The MYNT EYE SDK is compatible with Ubuntu 14.04/16.04.

We provide different versions for GCC 4 and 5 because GCC 5 uses a new ABI by default (see here). It is recommended that you use the consistent version.

Prerequisites

```
# Install build tools
$ sudo apt-get install build-essential cmake git

# Install OpenCV dependencies
$ sudo apt-get install pkg-config libgtk2.0-dev

# Install ssl for https, v4l for video
$ sudo apt-get install libssl-dev libv4l-dev v4l-utils
```

Install OpenCV 3.2

```
$ git clone https://github.com/opencv/opencv.git
$ cd opencv/
$ git checkout tags/3.2.0

$ cd opencv/
$ 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 \
..

# If CMake hangs during "ICV: Downloading ippicv_linux_20151201.tgz..."
$ ICV_PATH=../3rdparty/ippicv/downloads/linux-808b791a6eac9ed78d32a7666804320e/ippicv_linux_20151201.tgz &&
       \

      ICV_URL=https://github.com/opencv/opencv_3rdparty/raw/ippicv/master_20151201/ippicv/ippicv_linux_20151201.tgz && \
rm -f $ICV_PATH && wget $ICV_URL && mv ippicv_linux_20151201.tgz $ICV_PATH

$ make -j$(nproc --all)
$ sudo make install
```

Install CUDA (Optional)

If you have Nvidia's graphic card, you could enable GPU compute capability as follows:

- Ensure your graphic card drivers are up to date.
- Download and install latest version of CUDA Toolkit.

Install SDK

Go to the SDK directory and edit `sdk.cfg` to set your OpenCV install directory etc., then install:

```
$ cd <sdk directory>
$ ./install.sh
$ source ~/.bashrc
```

Check Installation

After `./install.sh`, you could check the environment variables:

```
$ echo $MYNTEYE_SDK_ROOT
<Should be the sdk path>
```

Build SDK Samples

```
$ cd samples/
$ mkdir build
$ cd build/
$ cmake -DCMAKE_BUILD_TYPE=Release ..
$ make
```

Then you could run the sample like this:

```
$ ./samples/bin/camera [name]
```

Please see "samples/README.md" to learn more about the samples.

About SDK Tools

Please see "tools/README.md".

Open SDK Documentation

```
$ xdg-open doc/api/html/index.html
```

FAQ

If have issues when installing, you could make an attempt to find it in FAQ.

2.2 Getting Started on macOS

Prerequisites

Install Homebrew to manage packages,

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Then,

```
# Install build tools
$ brew install cmake git

# Install ssl for https
$ brew install openssl
```

Install OpenCV 3.2

```
$ git clone https://github.com/opencv/opencv.git
$ cd opencv/
$ git checkout tags/3.2.0

$ cd opencv/
$ 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 -j8
$ sudo make install
```

Install CUDA (Optional)

If you have Nvidia's graphic card, you could enable GPU compute capability as follows:

- Ensure your graphic card drivers are up to date.
- Download and install latest version of `CUDA Toolkit`.

```
$ brew tap caskroom/cask
$ brew cask install cuda
$ brew cask info cuda
```

Install SDK

```
$ cd <sdk directory>
$ ./install.sh
$ source ~/.bash_profile
```

If your CUDA is not installed in `/usr/local/cuda/lib`, before running `./install.sh`, you should change the following variables in `./install.sh`:

```
RPATH_CUDA=/usr/local/cuda/lib
```

Then, the apps in SDK could detect the CUDA runtime correctly.

Note: It is not recommended that install OpenCV or CUDA in a different location. Because the `install.sh` not change the SDK libraries' rpath of them now. If you run the binary executables on Terminal, you could easily specify OpenCV and CUDA library search paths in `~/.bash_profile` like this:

```
CUDA_LIBDIR=/usr/local/cuda/lib
OPENCV_LIBDIR=/usr/local/lib
export DYLD_LIBRARY_PATH=$CUDA_LIBDIR:$DYLD_LIBRARY_PATH
export DYLD_LIBRARY_PATH=$OPENCV_LIBDIR:$DYLD_LIBRARY_PATH
```

Check Installation

After `./install.sh`, you could check the environment variables:

```
$ echo $MYNTEYE_SDK_ROOT
<Should be the sdk path>
```

Build SDK Samples

```
$ cd samples/
$ mkdir build
$ cd build/
$ cmake -DCMAKE_BUILD_TYPE=Release ..
$ make
```

Then you could run the sample like this:

```
$ ./samples/bin/camera [name]
```

Please see "samples/README.md" to learn more about the samples.

About SDK Tools

Please see "tools/README.md".

Open SDK Documentation

```
$ open doc/api/html/index.html
```

FAQ

If have issues when installing, you could make an attempt to find it in FAQ.

2.3 Getting Started on Windows (MSVC)

Install CUDA (Optional)

If you have Nvidia's graphic card, you could enable GPU compute capability as follows:

- Ensure your graphic card drivers are up to date.
- Download and install latest version of CUDA Toolkit 9.x.

Install SDK

Using Executable

Run the executable named `*.exe` to install.

Using Archive

Extract the archive named `*.tar.gz`.

Run "Command Prompt (cmd)" as administrator, then:

```
> cd <sdk directory>
> install.bat
```

> Note: Must open a new "Command Prompt (cmd)" as administrator, then execute `install.bat` to setup SDK. Otherwise, the environment variables won't be correct. Besides, you'd better reopen a "Command Prompt (cmd)" to "Check Installation" and "Build SDK Samples".

Check Installation

After installation, you could check the following environment variables in system:

- MYNTEYE_SDK_ROOT=<Should be the sdk path>
- MYNTEYE_SDK_PATH=<sdk>\bin;<sdk>\bin\3rdparty;<sdk>\apps;<sdk>\apps\3rdparty;<sdk>\deps\opencv\x64\vc14\b
- PATH=%MYNTEYE_SDK_PATH%;..
  - Ensure `%MYNTEYE_SDK_PATH%` in `PATH` for searching the libraries.
  - Ensure `%MYNTEYE_SDK_PATH%` is the first entry to avoid searching the libraries from other paths.
- MYNTEYE_SDK_DATA=<Where data will be stored, Optional>
  - Must be the path that the user has write privilege.

> If dll not found, you could reboot your computer to ensure PATH takes effect. As we know, it must be reboot on Windows 7.

How to solve issues when run `Previewer`

- Error: `opencv*.dll`, `mynteye*.dll`, `Qt5*.dll` not found
  - Check installation, ensure `%MYNTEYE_SDK_PATH%` in `PATH` and reboot.
- Error: `api-ms-win-crt-runtime-l1-1-0.dll` not found
  - Download and install `Universal CRT`, see `here`.
  - Or, download and install `this (Chinese)`.
- Error: `nppi64_*.dll` not found
  - Please install `CUDA Toolkit`.
  - E.g. `nppi64_80.dll` needs "CUDA Toolkit 8.0".

Build SDK Samples

Prerequisites

Windows Development Kits:

- Visual Studio
  - Visual Studio 2015
  - Note: CUDA Samples v8.0 only support vs2010-2015 now.

Add the following environment variables to PATH:

```
C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\bin
C:\Program Files (x86)\MSBuild\14.0\Bin
```

Build Essential Tools:

- CMake

Note: You'd better reopen a "Command Prompt (cmd)" after you change the PATH.

Build

```
> cd samples
> mkdir build
> cd build

> cmake -DCMAKE_BUILD_TYPE=Release ^
-G "Visual Studio 14 2015 Win64" ^
..

> msbuild.exe ALL_BUILD.vcxproj /property:Configuration=Release
# Or, open "<sdk>\samples\build\mynteye_sdk_samples.sln" with Visual Studio to build.
#   Please select "Configuration: Release" "Platform: x64" in solution configuration properties.
#
# If run with "Debug x64" in Visual Studio, you should open project properties and setup:
#   C/C++ > Code Generation, Runtime Library = Multi-threaded DLL (/MD)
```

Then you could run the sample like this:

```
> .\samples\bin\camera.exe
```

Please see "samples/README.md" to learn more about the samples.

About SDK Tools

Please see "tools/README.md".

Open SDK Documentation

```
> start doc\api\html\index.html
```

How about using different version of Visual Studio

The version must be greater than or equal to 2010, which support C++11 features.

You should add the corresponding environment variables to `PATH`:

```
C:\Program Files (x86)\Microsoft Visual Studio <?>\VC\bin
C:\Program Files (x86)\MSBuild\<?>\Bin
```

And change the following command like this:

```
> cmake -DCMAKE_BUILD_TYPE=Release ^
-G "Visual Studio <?> <?> Win64" ^
..
```

You can also run `cmake -h` to learn more.

However, it is recommended that use Visual Studio 2015, see FAQ 6.

FAQ

If have issues when installing, you could make an attempt to find it in `FAQ`.

2.4 Getting Started on Tegra (TX1, TX2)

> Note: no apps

Install `JetPack`

Download and install `JetPack`, please follow this `Install Guide`.

```
# On the host machine running Ubuntu, create a new directory to store installation packages.
$ mkdir JetPack
$ cd JetPack/

# Download JetPack-${VERSION}.run into the new directory on the host Ubuntu machine.
$ chmod a+x ./JetPack-L4T-3.1-linux-x64.run
```

Prerequisites

```
# Install build tools
$ sudo apt-get install build-essential cmake git

# Install ssl for https, v4l for video
$ sudo apt-get install libssl-dev libv4l-dev
```

Install SDK

Go to the SDK directory and edit `sdk.cfg` to set your OpenCV install directory etc., then install:

```
$ cd <sdk directory>
$ ./install.sh
$ source ~/.bashrc
```

Check Installation

After `./install.sh`, you could check the environment variables:

```
$ echo $MYNTEYE_SDK_ROOT
<Should be the sdk path>
```

Build SDK Samples

```
$ cd samples/
$ mkdir build
$ cd build/
$ cmake -DCMAKE_BUILD_TYPE=Release ..
$ make
```

Then you could run the sample like this:

```
$ ./samples/bin/camera [name]
```

Please see "samples/README.md" to learn more about the samples.

About SDK Tools

Please see "tools/README.md".

Open SDK Documentation

```
$ xdg-open doc/api/html/index.html
```

FAQ

If have issues when installing, you could make an attempt to find it in FAQ.

Chapter 3

Tutorials

Calibration

- How to calibrate camera with OpenCV
- How to calibrate camera and IMU with Kalibr

Firmware

Firmwares are here.

- How to upgrade firmware

3.1 How to calibrate camera with OpenCV

When using MYNT EYE camera first time, the SDK will download a calibration file for this camera to "settings/S↩
N∗.conf".

Besides, the SDK also provides a `stereo_calib` tool to help calibrate camera with OpenCV by yourself.

stereo_calib

```
$ cd $MYNTEYE_SDK_ROOT
$ ./tools/stereo_calib -help
Stereo calibration with square chessboard.

  Calibrate the stereo camera and display rectified results along with the computed disparity images.

  Chessboard pattern which has a size of 9 X 6: http://docs.opencv.org/master/pattern.png.

Usage:

  ./stereo_calib -w=9 -h=6 -s=1.0 [-nr] [-n=12 -i=1] <image list>

Options:
                     Default    Description
  -help                         show help message

  -w=<board_width>         9    number of inner corners per a chessboard row
  -h=<board_height>        6    number of inner corners per a chessboard column
  -s=<square_size>       1.0    the square side length in real measurement
                                    scale the vectors of the calibration pattern points
  -nr                           not display rectified results

  -n=<image_number>        0    capture images manually if greater then 0
                                    stored into `./images` directory
  -i=<device_index>        0    device index, but default is MYNTEYE on Windows
                                    `./tools/list_devices` to see indexes on Linux/Mac

  <image list>  ./stereo_calib.xml  given images in a XML/YML file
```

Calibration

1. Prepare a calibration board with square chessboard.
   - Could print a chessboard pattern and paste it on a board which had better be flat and hard.
2. Plug the camera in a USB 3.0 port.
3. Run `stereo_calib` to calibrate camera.

```
```
Manually capture `12` images of device `1`, then calibrate.
$ ./tools/stereo_calib -w=9 -h=6 -s=25 -n=12 -i=1

Using captured images in `images/imagelist.yaml` to calibrate.
$ ./tools/stereo_calib -w=9 -h=6 -s=25 images/imagelist.yaml
```
```

   - If not sure which index is MYNT EYE, you could run `./tools/list_devices`.
   - If not `-nr` option, you will preview rectified results after calibration done.
     - Press `space` to next or `esc` to exit.
4. The calibration file will be saved to "<MYNTEYE_SDK_ROOT>/settings/SN*.conf".

References

- Camera calibration With OpenCV

3.2 How to calibrate camera and IMU with Kalibr

TODO.

> However, you could see `cameraimu_calibration_guidebook.md` in MYNT-EYE-OKVIS-Sample now.

3.3 How to upgrade firmware

> Languages: 简中

Application Introduction

Application Package: `setup.exe`.

1. This application is dedicated to MYNT EYE and is used usually for upgrading firmware. You can also use it to preview images and IMU data.
2. This application is only available on Windows 7, Windows 8 and Windows 10.
3. This application may not be able to preview images on the computers which don't have some VS runtime libraries. However, upgrading firmware still works.
   - You can install the common VS runtime libraries to solve it, see here.

Firmware Upgrade Instructions

1. Plug the MYNT EYE camera in a USB 3.0 port, then open the application.
2. Click "Options" on menu bar, and select "FirmwareUpdate". And a upgrade window will pop up.
3. On the upgrade window, confirm "Device" is selected as "MYNTEYE". Then click "Update" on right side, and select the "mynteye_∗.img" we offerred to upgrade.
    - During upgrading, do not disconnect the MYNT EYE camera.
    - On upgrade window, "Status" shows the message. After done, the message will be "Program Succeeded, disconnect the camera".
4. After a firmware is upgraded, please replug the MYNT EYE camera.
    - **Before using MYNT EYE for the first time after upgraded, please keep it stable on the table for 6 seconds.** The reason is that:
    - When MYNT EYE works for the first time, there is a IMU zero drift compensation process. You can reopen the application and see IMU data changes at the bottom-left of the application interface.

Other Instructions:

1. On the upgrade window, if "Device" is "Cypress FX3 USB BootLoader Device" instead of "MYNTEYE", you can also click "Update" on right side to upgrade.
2. On the upgrade window, if you click "Erase" on right side and erase the firmware, you need to reopen the application. After entering the upgrade window again, you will see "Device" is "Cypress FX3 USB BootLoader Device", then click "Update" on right side to upgrade.
3. After entering the upgrade window again, if "Device" is not "Cypress FX3 USB BootLoader Device" or is not found, you need manually install the driver. The steps are as follows:
    - Open "Device Manager" in Windows, find the device named "WestBridge_driver", and right click it to update the driver.
    - Go to "<Installation Directory>/WestBridge_driver", select the folder corresponding to current system (win7 above select "wlh"), select "x86" or "x64" corresponding to the bits, finally select the driver to upgrade.
    - After the driver is installed, refresh the "Device Manager". You will find "Cypress FX3 USB BootLoader Device" now. If it fails, try to replug the camera.

Chapter 4

API Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

4.2 Modules

Here is a list of all modules:

Chapter 5

Class Documentation

5.1 mynteye::CalibrationParameters Struct Reference

Intrinsic parameters of each cameras and extrinsic (translation and rotation).

Public Member Functions

- CalibrationParameters ()

  Constructor.
- CalibrationParameters (const cv::Mat &M1, const cv::Mat &D1, const cv::Mat &M2, const cv::Mat &D2, const cv::Mat &R, const cv::Mat &T)

  Constructor.
- ~CalibrationParameters ()

  Destructor.
- ErrorCode Load (const std::string &filepath)

  Loads the calibration parameters from a file.
- ErrorCode Load (const std::string &intrinsic_filepath, const std::string &extrinsic_filepath)

  Loads the calibration parameters from two different file.
- ErrorCode Save (const std::string &filepath)

  Saves the calibration parameters into a file.

Public Attributes

- cv::Mat M1

  Input/output left camera matrix.
- cv::Mat D1

  Input/output lens distortion coefficients for left camera.
- cv::Mat M2

  Input/output right camera matrix.
- cv::Mat D2

  Input/output lens distortion coefficients for right camera.
- cv::Mat R

  Output rotation matrix between left and right camera coordinate systems.
- cv::Mat T

  Output translation vector between the coordinate systems of the cameras.

5.1.1 Detailed Description

Intrinsic parameters of each cameras and extrinsic (translation and rotation).

5.1.2 Constructor & Destructor Documentation

5.1.2.1 CalibrationParameters() [1/2]

```
mynteye::CalibrationParameters::CalibrationParameters ( )
```

Constructor.

5.1.2.2 CalibrationParameters() [2/2]

```
mynteye::CalibrationParameters::CalibrationParameters (
            const cv::Mat & M1,
            const cv::Mat & D1,
            const cv::Mat & M2,
            const cv::Mat & D2,
            const cv::Mat & R,
            const cv::Mat & T )
```

Constructor.

5.1.2.3 ~CalibrationParameters()

```
mynteye::CalibrationParameters::~CalibrationParameters ( )
```

Destructor.

5.1.3 Member Function Documentation

5.1.3.1 Load() [1/2]

```
ErrorCode mynteye::CalibrationParameters::Load (
            const std::string & filepath )
```

Loads the calibration parameters from a file.

Parameters

filepath	the file path from which the parameters will be loaded.

Returns

SUCCESS if ok, otherwise error.

5.1.3.2 Load() [2/2]

```
ErrorCode mynteye::CalibrationParameters::Load (
            const std::string & intrinsic_filepath,
            const std::string & extrinsic_filepath )
```

Loads the calibration parameters from two different file.

Parameters

intrinsic_filepath	the file path from which the intrinsic parameters will be loaded.
extrinsic_filepath	the file path from which the extrinsic parameters will be loaded.

Returns

SUCCESS if ok, otherwise error.

5.1.3.3 Save()

```
ErrorCode mynteye::CalibrationParameters::Save (
            const std::string & filepath )
```

Saves the calibration parameters into a file.

Parameters

filepath	the file path in which the parameters will be stored.

Returns

SUCCESS if ok, otherwise error.

5.1.4 Member Data Documentation

5.1.4.1 D1

`cv::Mat mynteye::CalibrationParameters::D1`

Input/output lens distortion coefficients for left camera.

See also

distCoeffs1 parameter of `stereoCalibrate()`

5.1.4.2 D2

`cv::Mat mynteye::CalibrationParameters::D2`

Input/output lens distortion coefficients for right camera.

See also

distCoeffs2 parameter of `stereoCalibrate()`

5.1.4.3 M1

`cv::Mat mynteye::CalibrationParameters::M1`

Input/output left camera matrix.

See also

cameraMatrix1 parameter of `stereoCalibrate()`

5.1.4.4 M2

`cv::Mat mynteye::CalibrationParameters::M2`

Input/output right camera matrix.

See also

cameraMatrix1 parameter of `stereoCalibrate()`

5.1.4.5 R

`cv::Mat mynteye::CalibrationParameters::R`

Output rotation matrix between left and right camera coordinate systems.

See also

R parameter of `stereoCalibrate()`

5.1.4.6 T

`cv::Mat mynteye::CalibrationParameters::T`

Output translation vector between the coordinate systems of the cameras.

See also

T parameter of `stereoCalibrate()`

5.2 mynteye::Camera Class Reference

The main class to use the MYNT EYE camera.

Public Member Functions

- Camera ()
  Constructor.
- ~Camera ()
  Destructor.
- void SetMode (const Mode &mode)
  Sets mode.
- void SetScale (float scale)
  Sets scale factor to the grabbed images.
- ErrorCode Open (const InitParameters &params)
  Opens the MYNE EYE camera.
- bool IsOpened ()
  Tests if the camera is opened.
- void ActivateAsyncGrabFeature (bool keep_imu_frequency=false)
  Activates async grap feature.
- void DeactivateAsyncGrabFeature ()
  Deactivates async grap feature.
- bool IsAsyncGrabFeatureActivated ()
  Tests if the async grap feature is activated.
- void ActivateDepthMapFeature ()
  Activates depth map feature.

- void DeactivateDepthMapFeature ()

  Deactivates depth map feature.
- bool IsDepthMapFeatureActivated ()

  Tests if the depth map feature is activated.
- void ActivatePointCloudFeature ()

  Activates point cloud feature.
- void DeactivatePointCloudFeature ()

  Deactivates point cloud feature.
- bool IsPointCloudFeatureActivated ()

  Tests if the point cloud feature is activated.
- void SetGrabErrorCallback (GrabErrorCallback callback)

  Sets grab error callback.
- void SetGrabProcessCallbacks (GrabPreProcessCallback pre_callback, GrabPostProcessCallback post_↩ callback)

  Sets process grab callbacks.
- void SetRectifyProcessCallbacks (RectifyPreProcessCallback pre_callback, RectifyPostProcessCallback post_callback)

  Sets process rectification callbacks.
- void SetDepthMapProcessCallbacks (DepthMapPreProcessCallback pre_callback, DepthMapPost↩ ProcessCallback post_callback)

  Sets process depth map callbacks.
- void SetPointCloudProcessCallbacks (PointCloudPreProcessCallback pre_callback, PointCloudPost↩ ProcessCallback post_callback)

  Sets process point cloud callbacks.
- ErrorCode Grab ()

  Grabs the new image, and process it.
- ErrorCode RetrieveImage (cv::Mat &mat, const View &view=View::VIEW_LEFT)

  Retrieves a image of wanted type.
- ErrorCode RetrieveIMUData (std::vector< IMUData > &imudatas, std::uint32_t &timestamp)

  Retrieves the IMU datas.
- Resolution GetResolution ()

  Returns the current image size.
- CameraInformation GetCameraInformation ()

  Returns the camera informations.
- CalibrationParameters GetCalibrationParameters ()

  Returns the calibration parameters.
- std::uint64_t GetDroppedCount (const Process &proc)

  Returns the dropped count in some process.
- void Close ()

  Closes the camera and free the memory.
- void ActivatePlugin (const std::string &libpath)

  Activates plugin.
- void DeactivatePlugin ()

  Deactivates plugin.
- bool IsPluginActivated ()

  Tests if plugin is activated.
- bool IsFirmwareLatest ()

  Tests if firmware is latest.

Static Public Member Functions

- static std::string GetSDKRoot ()

  Returns the SDK root directory.
- static std::string GetSDKVersion ()

  Returns the SDK version.

5.2.1 Detailed Description

The main class to use the MYNT EYE camera.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Camera()

```
mynteye::Camera::Camera ( )
```

Constructor.

5.2.2.2 ~Camera()

```
mynteye::Camera::~Camera ( )
```

Destructor.

5.2.3 Member Function Documentation

5.2.3.1 ActivateAsyncGrabFeature()

```
void mynteye::Camera::ActivateAsyncGrabFeature (
            bool keep_imu_frequency = false )
```

Activates async grap feature.

Parameters

keep_imu_frequency	whether keep imu frequency or not.

5.2.3.2 ActivatePlugin()

```
void mynteye::Camera::ActivatePlugin (
            const std::string & libpath )
```

Activates plugin.

Parameters

libpath	the plugin library path.

5.2.3.3 GetCalibrationParameters()

```
CalibrationParameters mynteye::Camera::GetCalibrationParameters ( )
```

Returns the calibration parameters.

Returns

the calibration parameters.

5.2.3.4 GetCameraInformation()

```
CameraInformation mynteye::Camera::GetCameraInformation ( )
```

Returns the camera informations.

Returns

the camera informations.

5.2.3.5 GetDroppedCount()

```
std::uint64_t mynteye::Camera::GetDroppedCount (
            const Process & proc )
```

Returns the dropped count in some process.

Returns

the dropped count in some process.

5.2.3.6 GetResolution()

```
Resolution mynteye::Camera::GetResolution ( )
```

Returns the current image size.

Returns

the current image size.

5.2.3.7 GetSDKRoot()

```
static std::string mynteye::Camera::GetSDKRoot ( )  [static]
```

Returns the SDK root directory.

Returns

the SDK root directory.

5.2.3.8 GetSDKVersion()

```
static std::string mynteye::Camera::GetSDKVersion ( )  [static]
```

Returns the SDK version.

Returns

the SDK version.

5.2.3.9 Grab()

```
ErrorCode mynteye::Camera::Grab ( )
```

Grabs the new image, and process it.

Note

The grabbing function is typically called in the main loop.
If ActivateAsyncGrabFeature(), you should SetGrabErrorCallback() to listen the grab error. And this method only return ERROR_CAMERA_GRAB_FAILED if there is not a new frame (it will block a moment to wait for grabbing).

Returns

SUCCESS if ok, otherwise error.

5.2.3.10 IsAsyncGrabFeatureActivated()

```
bool mynteye::Camera::IsAsyncGrabFeatureActivated ( )
```

Tests if the async grap feature is activated.

Returns

`true` if activated.

5.2.3.11 IsDepthMapFeatureActivated()

```
bool mynteye::Camera::IsDepthMapFeatureActivated ( )
```

Tests if the depth map feature is activated.

Returns

`true` if activated.

5.2.3.12 IsFirmwareLatest()

```
bool mynteye::Camera::IsFirmwareLatest ( )
```

Tests if firmware is latest.

Returns

`true` if latest.

5.2.3.13 IsOpened()

```
bool mynteye::Camera::IsOpened ( )
```

Tests if the camera is opened.

Returns

`true` if opened.

5.2.3.14 IsPluginActivated()

```
bool mynteye::Camera::IsPluginActivated ( )
```

Tests if plugin is activated.

Returns

`true` if activated.

5.2.3.15 IsPointCloudFeatureActivated()

```
bool mynteye::Camera::IsPointCloudFeatureActivated ( )
```

Tests if the point cloud feature is activated.

Returns

`true` if activated.

5.2.3.16 Open()

```
ErrorCode mynteye::Camera::Open (
            const InitParameters & params )
```

Opens the MYNE EYE camera.

Returns

SUCCESS if ok, otherwise error.

5.2.3.17 RetrieveImage()

```
ErrorCode mynteye::Camera::RetrieveImage (
            cv::Mat & mat,
            const View & view = View::VIEW_LEFT )
```

Retrieves a image of wanted type.

Parameters

mat	the Mat to store the image.
view	defines the image type wanted, see View.

Returns

SUCCESS if ok, otherwise error.

Note

The retrieve function should be called after the function Camera::Grab.

5.2.3.18 RetrieveIMUData()

```
ErrorCode mynteye::Camera::RetrieveIMUData (
            std::vector< IMUData > & imudatas,
            std::uint32_t & timestamp )
```

Retrieves the IMU datas.

Parameters

imudatas	the given IMU datas.
timestamp	the timestamp which elapsed after camera working with unit 0.1ms.

Returns

SUCCESS if ok, otherwise error.

Note

timestamp_ms = timestamp / 10

5.2.3.19 SetMode()

```
void mynteye::Camera::SetMode (
            const Mode & mode )
```

Sets mode.

default: MODE_AUTO_DETECT

Parameters

mode	the compute mode.

See also

Mode

5.2.3.20 SetScale()

```
void mynteye::Camera::SetScale (
            float scale )
```

Sets scale factor to the grabbed images.

Parameters

scale	the scale factor.

5.3 mynteye::CameraInformation Struct Reference

Camera information.

Public Member Functions

- ErrorCode Load (const std::string &filepath)

  Loads the camera information from a file.
- ErrorCode Save (const std::string &filepath)

  Saves the camera information into a file.

Public Attributes

- std::string serial

  The serial number.
- std::string version

  The firmware version.
- std::string product

  The product name.
- std::string manufacturer

  The manufacturer name.
- uint16_t product_id

  The product id.
- uint16_t vendor_id

  The vendor id.

5.3.1 Detailed Description

Camera information.

5.3.2 Member Function Documentation

5.3.2.1 Load()

```
ErrorCode mynteye::CameraInformation::Load (
            const std::string & filepath )
```

Loads the camera information from a file.

Parameters

filepath	the file path from which the information will be loaded.

Returns

SUCCESS if ok, otherwise error.

5.3.2.2 Save()

```
ErrorCode mynteye::CameraInformation::Save (
            const std::string & filepath )
```

Saves the camera information into a file.

Parameters

filepath	the file path in which the information will be stored.

Returns

SUCCESS if ok, otherwise error.

5.4 mynteye::IMUData Struct Reference

IMU (inertial measurement unit) data.

Public Attributes

- std::uint32_t time

  Time which elapsed after camera working with unit 0.1ms.
- std::int16_t time_offset

  Time offset before retrieve IMU data with unit 0.1ms.

- float accel_x

  Accelerometer data for 3-axis: x, y, z.
- float accel_y

  Accelerometer data for 3-axis: x, y, z.
- float accel_z

  Accelerometer data for 3-axis: x, y, z.

- float gyro_x

  Gyroscope data for 3-axis: x, y, z.
- float gyro_y

  Gyroscope data for 3-axis: x, y, z.
- float gyro_z

  Gyroscope data for 3-axis: x, y, z.

5.4.1 Detailed Description

IMU (inertial measurement unit) data.

5.4.2 Member Data Documentation

5.4.2.1 accel_x

```
float mynteye::IMUData::accel_x
```

Accelerometer data for 3-axis: x, y, z.

See also

accel_x, accel_y, accel_z

5.4.2.2 accel_y

```
float mynteye::IMUData::accel_y
```

Accelerometer data for 3-axis: x, y, z.

See also

accel_x, accel_y, accel_z

5.4.2.3 accel_z

```
float mynteye::IMUData::accel_z
```

Accelerometer data for 3-axis: x, y, z.

See also

accel_x, accel_y, accel_z

5.4.2.4 gyro_x

```
float mynteye::IMUData::gyro_x
```

Gyroscope data for 3-axis: x, y, z.

See also

gyro_x, gyro_y, gyro_z

5.4.2.5 gyro_y

```
float mynteye::IMUData::gyro_y
```

Gyroscope data for 3-axis: x, y, z.

See also

gyro_x, gyro_y, gyro_z

5.4.2.6 gyro_z

```
float mynteye::IMUData::gyro_z
```

Gyroscope data for 3-axis: x, y, z.

See also

gyro_x, gyro_y, gyro_z

5.4.2.7 time

```
std::uint32_t mynteye::IMUData::time
```

Time which elapsed after camera working with unit 0.1ms.

Note

time = time_retrieve + time_offset
time_ms = time / 10

See also

Camera::RetrieveIMUData()

5.4.2.8 time_offset

```
std::int16_t mynteye::IMUData::time_offset
```

Time offset before retrieve IMU data with unit 0.1ms.

Note

time_offset_ms = time_offset / 10

5.5 mynteye::InitParameters Struct Reference

Parameters for MYNT EYE initialization.

Public Member Functions

- InitParameters ()

  Constructor.
- InitParameters (const std::string &name, const std::int32_t &framerate=60, const CalibrationParameters *clib_params=nullptr)

  Constructor.
- ~InitParameters ()

  Destructor.
- ErrorCode Load (const std::string &filename)

  Loads the values of the parameters contained in a file.
- ErrorCode Save (const std::string &filename)

  Saves the current bunch of parameters into a file.

Public Attributes

- std::string name

  Camera name or index.
- std::int32_t framerate

  Camera framerate.
- const CalibrationParameters * clib_params

  Calibration parameters.

5.5.1 Detailed Description

Parameters for MYNT EYE initialization.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 InitParameters() [1/2]

```
mynteye::InitParameters::InitParameters ( )
```

Constructor.

5.5.2.2 InitParameters() [2/2]

```
mynteye::InitParameters::InitParameters (
            const std::string & name,
            const std::int32_t & framerate = 60,
            const CalibrationParameters * clib_params = nullptr )
```

Constructor.

5.5.2.3 ~InitParameters()

```
mynteye::InitParameters::~InitParameters ( )
```

Destructor.

5.5.3 Member Function Documentation

5.5.3.1 Load()

```
ErrorCode mynteye::InitParameters::Load (
            const std::string & filename )
```

Loads the values of the parameters contained in a file.

Parameters

filename	the path to the file from which the parameters will be loaded.

Returns

SUCCESS if ok, otherwise error.

5.5.3.2 Save()

```
ErrorCode mynteye::InitParameters::Save (
            const std::string & filename )
```

Saves the current bunch of parameters into a file.

Parameters

filename	the path to the file in which the parameters will be stored.

Returns

SUCCESS if ok, otherwise error.

5.5.4 Member Data Documentation

5.5.4.1 clib_params

```
const CalibrationParameters* mynteye::InitParameters::clib_params
```

Calibration parameters.

If not specified, we will download from our server by serial number.

default: nullptr

5.5.4.2 framerate

```
std::int32_t mynteye::InitParameters::framerate
```

Camera framerate.

default: 60

5.5.4.3 name

```
std::string mynteye::InitParameters::name
```

Camera name or index.

default values:

- Windows: MYNTEYE
- Mac: 0
- Linux: 0

5.6 mynteye::Plugin Class Reference

The plugin which could implement processing by yourself.

Public Member Functions

- virtual void OnCreate (Camera *camera)

  Called when plugin created.
- virtual void OnOpen (const ErrorCode &code)

  Called when camera opened.
- virtual void OnGrab (const ErrorCode &code)

  Called when camera grabbed.
- virtual void OnRetrieveImage (cv::Mat &mat, const View &view, const ErrorCode &code)

  Called when camera retrieved image.
- virtual void OnRetrieveIMUData (std::vector< IMUData > &imudatas, std::uint32_t &timestamp, const ErrorCode &code)

  Called when camera retrieved IMU datas.
- virtual void OnClose ()

  Called when camera closed.
- virtual void OnProcessGrab (cv::Mat &left_raw, cv::Mat &right_raw)

  Called before process grabbed images.
- virtual bool OnProcessRecify (const cv::Mat &left_raw, const cv::Mat &right_raw, cv::Mat &left_rectified, cv::Mat &right_rectified)

  Called before process rectification.
- virtual bool OnProcessDepthMap (const cv::Mat &left_rectified, const cv::Mat &right_rectified, cv::Mat &depthmap)

  Called before process depth map.
- virtual bool OnProcessPointCloud (const cv::Mat &depthmap, cv::Mat &pointcloud)

  Called before process point cloud.

5.6.1 Detailed Description

The plugin which could implement processing by yourself.

5.6.2 Member Function Documentation

5.6.2.1 OnCreate()

```
virtual void mynteye::Plugin::OnCreate (
            Camera * camera )  [inline], [virtual]
```

Called when plugin created.

Parameters

camera	the MYNT EYE camera.

5.6.2.2 OnGrab()

```
virtual void mynteye::Plugin::OnGrab (
            const ErrorCode & code )  [inline], [virtual]
```

Called when camera grabbed.

Parameters

code	SUCCESS if ok, otherwise error.

5.6.2.3 OnOpen()

```
virtual void mynteye::Plugin::OnOpen (
            const ErrorCode & code )  [inline], [virtual]
```

Called when camera opened.

Parameters

code	SUCCESS if ok, otherwise error.

5.6.2.4 OnProcessDepthMap()

```
virtual bool mynteye::Plugin::OnProcessDepthMap (
            const cv::Mat & left_rectified,
```

```
            const cv::Mat & right_rectified,
            cv::Mat & depthmap )  [inline], [virtual]
```

Called before process depth map.

Note

This method is called in a standalone processing thread.

Returns

true if processed by yourself. Then it won't process depth map any more in SDK.

5.6.2.5 OnProcessGrab()

```
virtual void mynteye::Plugin::OnProcessGrab (
            cv::Mat & left_raw,
            cv::Mat & right_raw )  [inline], [virtual]
```

Called before process grabbed images.

Note

This method is called in a standalone processing thread.

5.6.2.6 OnProcessPointCloud()

```
virtual bool mynteye::Plugin::OnProcessPointCloud (
            const cv::Mat & depthmap,
            cv::Mat & pointcloud )  [inline], [virtual]
```

Called before process point cloud.

Note

This method is called in a standalone processing thread.

Returns

true if processed by yourself. Then it won't process point cloud any more in SDK.

5.6.2.7 OnProcessRecify()

```
virtual bool mynteye::Plugin::OnProcessRecify (
            const cv::Mat & left_raw,
            const cv::Mat & right_raw,
            cv::Mat & left_rectified,
            cv::Mat & right_rectified )  [inline], [virtual]
```

Called before process rectification.

Note

This method is called in a standalone processing thread.

Returns

true if processed by yourself. Then it won't process rectification any more in SDK.

5.6.2.8 OnRetrieveImage()

```
virtual void mynteye::Plugin::OnRetrieveImage (
            cv::Mat & mat,
            const View & view,
            const ErrorCode & code )  [inline], [virtual]
```

Called when camera retrieved image.

Parameters

mat	the Mat to store the image.
view	defines the image type wanted, see View.
code	SUCCESS if ok, otherwise error.

5.6.2.9 OnRetrieveIMUData()

```
virtual void mynteye::Plugin::OnRetrieveIMUData (
            std::vector< IMUData > & imudatas,
            std::uint32_t & timestamp,
            const ErrorCode & code )  [inline], [virtual]
```

Called when camera retrieved IMU datas.

Parameters

imudatas	the given IMU datas.
timestamp	the timestamp which elapsed after MYNTEYE working with unit 0.1ms.
code	SUCCESS if ok, otherwise error.

Note

time_elapsed_ms = timestamp / 10

5.7 mynteye::Resolution Struct Reference

Image resolution about width and height.

Public Member Functions

- Resolution (const std::int32_t &w=0, const std::int32_t &h=0)

  Constructor.
- ~Resolution ()

  Destructor.
- std::int32_t Area () const

  Returns the area of the image.
- bool operator== (const Resolution &that) const

  Tests if the given Resolution has the same properties.
- bool operator!= (const Resolution &that) const

  Tests if the given Resolution has different properties.

Public Attributes

- std::int32_t width

  image width in pixels
- std::int32_t height

  image height in pixels

5.7.1 Detailed Description

Image resolution about width and height.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Resolution()

```
mynteye::Resolution::Resolution (
            const std::int32_t & w = 0,
            const std::int32_t & h = 0 )
```

Constructor.

5.7.2.2 ~Resolution()

```
mynteye::Resolution::~Resolution ( )
```

Destructor.

5.7.3 Member Function Documentation

5.7.3.1 Area()

```
std::int32_t mynteye::Resolution::Area ( ) const
```

Returns the area of the image.

Returns

The number of pixels of the image.

5.7.3.2 operator"!=()

```
bool mynteye::Resolution::operator!= (
            const Resolution & that ) const
```

Tests if the given Resolution has different properties.

Parameters

that	the given Resolution.

See also

operator==()

Returns

True if same, false different.

5.7.3.3 operator==()

```
bool mynteye::Resolution::operator== (
            const Resolution & that ) const
```

Tests if the given Resolution has the same properties.

Parameters

that	the given Resolution.

See also

operator!=()

Returns

True if same, false different.

Chapter 6

Module Documentation

6.1 Public enumeration types

Enumerations

- enum mynteye::ErrorCode {
  mynteye::SUCCESS = 0, mynteye::ERROR_FAILURE, mynteye::ERROR_FILE_OPEN_FAILED, mynteye↩
  ::ERROR_CAMERA_OPEN_FAILED,
  mynteye::ERROR_CAMERA_NOT_OPENED, mynteye::ERROR_CAMERA_GRAB_FAILED, mynteye::E↩
  RROR_CAMERA_RETRIEVE_FAILED, mynteye::ERROR_CAMERA_RETRIEVE_NOT_READY,
  mynteye::ERROR_FEATURE_NOT_ACTIVATED, mynteye::ERROR_LAST }

  List error codes in the MYNT EYE SDK.
- enum mynteye::Mode { mynteye::MODE_AUTO_DETECT, mynteye::MODE_CPU, mynteye::MODE_GPU,
  mynteye::MODE_LAST }

  List available modes.
- enum mynteye::View {
  mynteye::VIEW_LEFT, mynteye::VIEW_RIGHT, mynteye::VIEW_LEFT_UNRECTIFIED, mynteye::VIEW_↩
  RIGHT_UNRECTIFIED,
  mynteye::VIEW_DEPTH_MAP, mynteye::VIEW_DEPTH_MAP_UNNORMALIZED, mynteye::VIEW_POIN↩
  T_CLOUD, mynteye::VIEW_LAST }

  List available views.
- enum mynteye::Process {
  mynteye::PROC_GRAB, mynteye::PROC_RECTIFY, mynteye::PROC_DEPTH_MAP, mynteye::PROC_P↩
  OINT_CLOUD,
  mynteye::PROC_LAST }

  List process names.
- enum mynteye::Gravity {
  mynteye::TOP_LEFT, mynteye::TOP_RIGHT, mynteye::BOTTOM_LEFT, mynteye::BOTTOM_RIGHT,
  mynteye::GRAVITY_LAST }

  List gravity values.

6.1.1 Detailed Description

6.1.2 Enumeration Type Documentation

6.1.2.1 ErrorCode

`enum mynteye::ErrorCode`

List error codes in the MYNT EYE SDK.

Enumerator

SUCCESS	Standard code for successful behavior.
ERROR_FAILURE	Standard code for unsuccessful behavior.
ERROR_FILE_OPEN_FAILED	File cannot be opened for not exist, not a regular file or any other reason.
ERROR_CAMERA_OPEN_FAILED	Camera cannot be opened for not plugged or any other reason.
ERROR_CAMERA_NOT_OPENED	Camera is not opened now.
ERROR_CAMERA_GRAB_FAILED	Camera grab the image failed.
ERROR_CAMERA_RETRIEVE_FAILED	Camera retrieve the image failed.
ERROR_CAMERA_RETRIEVE_NOT_READY	Camera retrieve the image not ready (in computing).
ERROR_FEATURE_NOT_ACTIVATED	Feature is not activiated now.
ERROR_LAST	Last guard.

6.1.2.2 Gravity

`enum mynteye::Gravity`

List gravity values.

Enumerator

TOP_LEFT	Top left.
TOP_RIGHT	Top right.
BOTTOM_LEFT	Bottom left.
BOTTOM_RIGHT	Bottom right.
GRAVITY_LAST	Last guard.

6.1.2.3 Mode

`enum mynteye::Mode`

List available modes.

Enumerator

MODE_AUTO_DETECT	Auto detect.
MODE_CPU	CPU.
MODE_GPU	GPU.
MODE_LAST	Last guard.

6.1.2.4 Process

enum mynteye::Process

List process names.

Enumerator

PROC_GRAB	Process grab.
PROC_RECTIFY	Process rectification.
PROC_DEPTH_MAP	Process depth map.
PROC_POINT_CLOUD	Process point cloud.
PROC_LAST	Last guard.

6.1.2.5 View

enum mynteye::View

List available views.

Enumerator

VIEW_LEFT	Left image, rectified.
VIEW_RIGHT	Right image, rectified.
VIEW_LEFT_UNRECTIFIED	Raw left image, unrectified.
VIEW_RIGHT_UNRECTIFIED	Raw right image, unrectified.
VIEW_DEPTH_MAP	Depth map, normalized [0,255].
VIEW_DEPTH_MAP_UNNORMALIZED	Depth map, unnormalized.
VIEW_POINT_CLOUD	Point cloud with OpenCV coordinate system. See here: http://docs.opencv.org/master/d9/d0c/group_↩ _calib3d.html#details
VIEW_LAST	Last guard.

Chapter 7

FAQ & Issues

1) How to run apps directly in Terminal/Command Prompt?

```
# Linux
$ ./apps/Previewer

# macOS
$ ./apps/Previewer.app/Contents/MacOS/Previewer

# Windows
> .\apps\Previewer.exe
```

2) Running apps or samples crashes due to the CUDA driver is not proper.

If you run `./apps/Previewer` or `./samples/bin/camera2 <index>` crashes and see this information:

```
CUDA Error: CUDA driver version is insufficient for CUDA runtime version
```

It seems that your CUDA driver is not proper for CUDA runtime.

You could select to install CUDA driver when install `CUDA Toolkit`.

3) Install the NVIDIA display driver failed on Ubuntu.

If you install the NVIDIA display driver failed like these:

```
Installing the NVIDIA display driver...
It appears that an X server is running. Please exit X before installation. If you're sure that X is not runnin
...
Driver:   Installation Failed

A system reboot is required to continue installation. Please reboot then run the installer again. An attmept h
...
Driver:   Reboot required to continue
```

You could install the driver as follows:

```
$ sudo apt-cache search nvidia-
$ sudo apt-get install nvidia-375
$ sudo reboot
```

Or, see this: `How can I install CUDA on Ubuntu 16.04?`.

4) Running apps in SDK (Previewer, etc.) doesn't detect the CUDA runtime correctly?

You could run apps directly in Terminal, and confirm "Detect CUDA support" or not by the output message.

However, run apps in Terminal is not the same as run them through click to start. You should not export the library search paths as follows:

```
# Linux
$ vi ~/.bashrc
export CUDA_HOME=/usr/local/cuda-8.0
export PATH=$CUDA_HOME/bin:$PATH
export LD_LIBRARY_PATH=$CUDA_HOME/lib64:$LD_LIBRARY_PATH

# macOS
$ vi ~/.bash_profile
export PATH=/usr/local/cuda-8.0/bin:$PATH
export PATH=$CUDA_HOME/bin:$PATH
export DYLD_LIBRARY_PATH=$CUDA_HOME/lib:$DYLD_LIBRARY_PATH
```

Otherwise, it may "Detect CUDA support" when run in Terminal, but "Detect CUDA not support" when click to start. Because when click to start apps, they won't know the search paths that you added for Terminal.

To ensure the apps could detect the CUDA runtime correctly, you should add library search paths to system.

On Linux, add the paths to "/etc/ld.so.conf":

```
# Add library directory at end: /usr/local/cuda-8.0/lib64
$ sudo vi /etc/ld.so.conf

# Rebuild the cache
$ sudo ldconfig

# Check if library directory in cache
$ ldconfig -p | grep cuda
```

On macOS, link the CUDA libs to "/usr/local/lib":

```
$ CUDA_LIBPATH=/usr/local/cuda/lib; \
for lib in libcuda.dylib libnppc.dylib libnppi.dylib; do \
    libpath="$CUDA_LIBPATH/$lib"; \
    if [ -e "$libpath" ]; then \
        ln -fhs "$libpath" "/usr/local/lib"; \
    fi \
done
```

5) What is "Warning: Failed to get the camera's infomation"?

On Windows, if you run a executable using SDK when you has plugged the camera for a while, you may see the following warning:

```
Warning: Failed to get the camera's infomation.
Warning: sn is empty.
```

Then, the SDK will fetch a default calibration file to "settings/SN.conf" and continue processing. Thus, it will lead to bad depthmap because of the unmatched calibration parameters.

You could replug the camera and run any executable using SDK to solve it.

The reason is the camera will be in low power state on Windows when you has plugged the camera for a while. And the SDK can not get the camera's infomation in this state.

About the camera infomation, the SDK will cache it to a temporary file "setting/mynteye.info.temp" once you got it. When you failed to get the infomation, the SDK will read it from the temporary file if exists. Besides, you could create a file "setting/mynteye.info" to specify the camera infomation.

6) "error LNK1104: cannot open file 'ws2_32.lib'" in using Visual Studio 2013

It is recommended that use Visual Studio 2015 to build "mynteye_sdk_samples.sln".

Alternatively, you could try explicitly linking 'ws2_32.lib' to the executable binaries in `<sdk>/samples/CMakeLists.txt`.

```
if(MSVC)
  target_link_libraries(<executable> wsock32 ws2_32)
endif()
```

7) "error while loading shared libraries: .. file too short" on Linux

```
lib/libmynteye_core.so: error while loading shared libraries: /home/cc/mynt/mynteye-1.5-linux-x64-gcc5-opencv-
```

It means the shared library size is incorrect. Please download the SDK and extract it again.

Besides, you could check the archive's MD5 checksum like that:

```
$ md5sum xxx.tar.gz
```

8) "error adding symbols: File in wrong format" on Linux

```
.../lib/libmynteye_core.so: error adding symbols: File in wrong format
```

It means the shared library format is incorrect. Check your system's architecture like that:

```
$ uname -a
Linux john-ubuntu 4.10.0-35-generic #39~16.04.1-Ubuntu SMP Wed Sep 13 09:02:42 UTC 2017 x86_64 x86_64
       x86_64 GNU/Linux
```

Arch	Archive Name
x86, i686, i386	-
x86_64	x64
arm	-
aarch64	aarch64

The "Archive Name" marked with − means not supported.

Then, download the SDK archive with the same architecture and install.

Known Issues

1) Open MYNT EYE camera failed on macOS.

It may hang when open the camera, after you open and close it several times on macOS.

Reconnecting the MYNT EYE camera will solve this issue.

Chapter 8

Release Notes

MYNT EYE SDK 1.5

- Support different IMU Hz firmware.
- Add exe package for Windows.
- Add sdk.cfg to help find OpenCV.
- Update api doc and generate pdf.

MYNT EYE SDK 1.4

- Add an option to keep IMU frequency when use async grab.
- Add cuda features with compute 5.3 for TX1.
- Fix some issues of samples on Tegra.
- Fix illegal instruction on some computers.
- Improve how to detect OpenCV with CMake.

MYNT EYE SDK 1.3

- Add async grab api.
- Add imu time offset detection.
- Add how to save dataset in samples.
- Fix a issue in using plugin.
- Improve install readme.

MYNT EYE SDK 1.2

- Add tutorials in doc.
- Add depth map unnormalized api.
- Complete TX1/TX2 support.
- Complete Linux AArch64 support.
- Release a new firmware version.

MYNT EYE SDK 1.1

- Add faq in doc.
- Add a firmware api.
- Fix a conflict issue.
- Improve app previewer.
- Complete Windows support.

MYNT EYE SDK 1.0

- Initial release.

Chapter 9

Appendix: 简中文档

教程

- 如何升级 MYNT EYE 的固件

9.1 如何升级 MYNT EYE 的固件

固件升级应用说明

应用安装包：`setup.exe`。

1. 该应用为 MYNT EYE 专用，主要用于升级固件。也可以预览图像及 IMU 数据。
2. 该应用仅适用于 Windows 7、Windows 8、Windows 10。
3. 部分电脑如果没有安装某些 VS 运行库将无法显示图像（但不影响升级固件和预览 IMU 数据）。
   - 下载安装常用 VS 运行库合集或能解决该问题，参考这里。

固件升级操作步骤

1. 将 MYNT EYE 插入 USB 3.0 口，然后打开应用。
2. 点击菜单栏的 Options，选择 FirmwareUpdate，会弹出升级窗口。
3. 在升级窗口里，确认 Device 选中为 MYNTEYE。然后点击右侧的 Update，选择我们提供的 mynteye_*.img 进行升级。
   - 升级过程中请勿拔插 MYNT EYE。
   - 升级状态显示在 Status，完成后提示“Program Succeeded, disconnect the camera”。
4. 固件升级完成后，请重插拔 MYNT EYE 再使用。
   - **MYNT EYE 固件升级后初次使用，请保持其静置在桌面 6 秒。** 原因如下：
   - MYNT EYE 初次工作会有一个 IMU 零漂补偿过程。你可以重开该应用，在左下方能看到 IMU 数据变动。

其他说明：

1. 在升级窗口里，如果 Device 不是 MYNTEYE ， 而是“Cypress FX3 USB BootLoader Device”，一样可以点击右侧的 Update 进行升级。
2. 在升级窗口里，如果点击了右侧的 Erase ，擦除了固件，你需要重开该应用。再次进入升级窗口后，你可以看到 Device 为“Cypress FX3 USB BootLoader Device”，继续点击右侧的 Update 进行升级。
3. 如果 Erase 后再次进入升级窗口时，Device 名称不是“Cypress FX3 USB BootLoader Device”或找不到，说明你需要手动安装驱动。步骤如下：
   - 打开系统的设备管理器，找到名称为“WestBridge_driver”的设备，然后右键更新驱动。
   - 选择“<应用安装目录>/WestBridge_driver”路径下，相应系统文件夹（win7以上选择wlh）下 x86/x64 （根据系统位数选择）内的驱动文件进行安装。
   - 驱动安装成功之后，刷新设备管理器，将找到“Cypress FX3 USB BootLoader Device”。如果失败，请尝试重插拔摄像头。

Index