MYNT EYE D SDK

1.7.2

制作者 Doxygen 1.8.11

Contents

Chapter 1

MYNT EYE D SDK

- 产品说明
- SDK 安装
- SDK 样例
- SDK 工具
- 工程样例
- API 说明

Chapter 2

产品说明

2.1 产品介绍

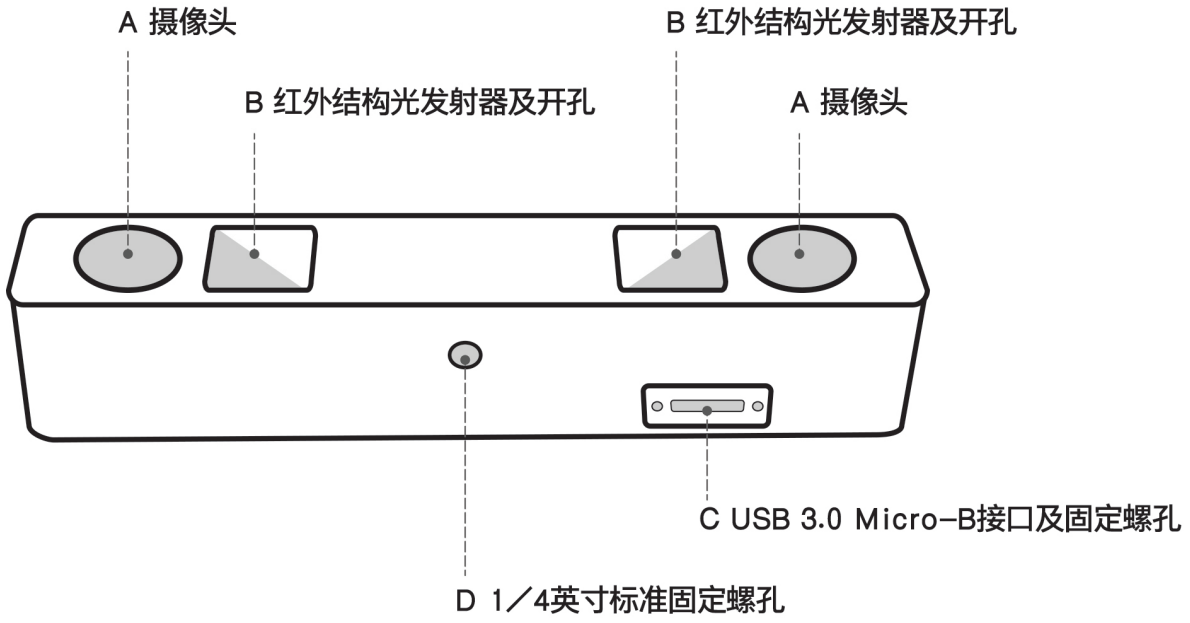
作为基于视觉识别技术的3D传感器，小觅双目摄像头深度版可适用于室内外双重环境。无惧室外强光环境，完全黑暗的室内环境亦可工作。标配的IR主动光，可以完美解决了室内白墙和无纹理物体的识别难题。“双目+IMU”的惯性导航方案，可为VSLAM的应用提供精准的六轴互补数据，并且相较其他单一方案拥有更高精度和鲁棒性。此外，小觅双目摄像头深度版产品（MYNT EYE Depth）还提供丰富的SDK接口和VSLAM开源项目支持，可以帮助客户迅速进行方案集成，加速实现产品研发进程，实现方案的快速产品化和落地。

小觅双目摄像头深度版（MYNT EYE Depth）可广泛应用于视觉定位导航（vSLAM）领域，包括：无人车和机器人的视觉实时定位导航系统、无人机视觉定位系统、无人驾驶避障导航系统、增强现实（AR）、虚拟现实（VR）等；双目也可应用于视觉识别领域，包括：立体人脸识别、三维物体识别、空间运动追踪、三维手势与体感识别等；应用于测量领域，包括：辅助驾驶系统（ADAS）、双目体积计算、工业视觉筛检等。

为保证摄像头产品输出数据质量，产品出厂时，我们已对双目以及IMU进行标定。同时，产品通过富士康实验室的高温高湿持续工作、高温高湿持续操作、低温动态老化、高温工作、低温存储、整机冷热冲击、正弦振动、随机振动等多项产品质量测试，保证品质的稳定和可靠。除了产品和技术的研发，亦可直接应用于产品量产，加速从研发到产品化的过程。

2.2 产品外观

外壳(mm)	PCBA板(mm)
165x31.↩ 5x29.6	149x24



A. 摄像头：摄像头传感器镜头，在使用中请注意保护，以避免成像质量下降。

B. 红外结构光发射器及开孔：通过红外结构光可有效解决白墙等无纹理表面的视觉计算。(非 IR 版，此孔保留，但内部无结构光发射装置)

C. USB Micro-B 接口及固定孔：使用中，插上 USB Micro-B 数据线后，请使用接口端的螺丝紧固接口，以避免使用中损坏接口，也保证数据连接的稳定性。

D. 1/4 英寸标准固定螺孔：用于将双目摄像头固定于摄影三角架等装置。

2.3 图像分辨率支持列表

L'=left rectify, L=left, R'=right rectify, R=right, D=depth	interface	color resolution	color fps	depth resolution	depth fps
L'+D	USB3.0	1280x720	60/30/20/10	1280x720	60/30/20/10
L'+D	USB3.0	640x480	60/30	640x480	60/30
L'+R'+D	USB3.0	2560x720	30	1280x720	30
L'+R'+D	USB3.0	1280x480	60/30	640x480	60/30
L+D	USB3.0	1280x720	60/30/20/10	1280x720	60/30/20/10
L+D	USB3.0	640x480	60/30	640x480	60/30
L+R+D	USB3.0	2560x720	30	1280x720	30
L+R+D	USB3.0	1280x480	60/30	640x480	60/30
L+R	USB3.0	2560x720	60/30	not open	null
L'+R'	USB3.0	2560x720	60/30	not open	null
D	USB3.0	not open	null	1280x720	60/30
D	USB3.0	not open	null	640x480	60/30
L+R	USB2.0	2560x720	5	not open	null
L'+R'	USB2.0	2560x720	5	not open	null
L+R	USB2.0	1280x480	15	not open	null

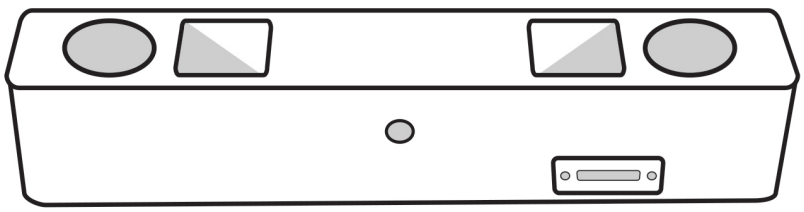
L'=left rectify, L=left, R'=right rectify, R=right, D=depth	interface	color resolution	color fps	depth resolution	depth fps
L'+R'	USB2.0	1280x480	15	not open	null
L'+D	USB2.0	1280x720	5	640x720	5
L'+D	USB2.0	640x480	15	320x480	15
L+D	USB2.0	1280x720	5	640x720	5
L+D	USB2.0	640x480	15	320x480	15
L'	USB2.0	1280x720	5	not open	null
L	USB2.0	1280x720	5	not open	null
D	USB2.0	not open	null	640x720	5
D	USB2.0	not open	null	320x480	15
L+R	USB2.0/MJPG	2560x720	5	not open	null
L+R	USB2.0/MJPG	1280x480	15	not open	null
L	USB2.0/MJPG	1280x720	5	not open	null

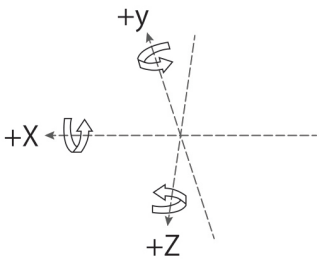
注意：

- L'=left rectify image, L=left image
- R'=right rectify image, R=right image, D=depth image
- 在IR Depth Only模式下，帧率只支持15fps和30fps.

2.4 IMU 坐标系统

IMU 坐标系统为右手系，坐标轴方向如下：





Chapter 3

SDK 安装

3.1 支持平台

SDK是基于CMake构建的，用以Linux，Windows等多个平台。SDK提供两种安装方式：下载安装以及源码安装编译方式。

已测试可用的平台有：

```
* Windows 10
* Ubuntu 18.04/16.04/14.04
* Jetson TX2
* RK3399
```

Tips:

- ubuntu系统仅支持源码编译安装。
- Ubuntu 14.04需要升级至gcc5.

Warning: 由于硬件传输速率要求，请尽量使用USB3.0接口。另外，虚拟机因大多存在USB驱动兼容性问题，不建议使用。

3.2 Linux SDK 用户指南

1. 安装 SDK 依赖

1.1 安装 OpenCV

如果您已经安装了 *opencv* 或者您想要使用 *ROS*，您可以跳过这步.

1.1.1 apt 或者编译安装 OpenCV (选择一个)

1.1.1.1 使用 apt 安装 OpenCV (推荐)

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

1.1.1.2 编译安装 OpenCV

OpenCV 如何编译安装，请见官方文档 Installation in Linux <https://docs.opencv.org/master/d7/d9f/tutorial_linux_install.html>_ 。或参考如下命令：

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

```
1 git clone https://github.com/opencv/opencv.git
2 cd opencv/
3 git checkout tags/3.4.0
4
5 cd opencv/
6 mkdir build
7 cd build/
8
9 cmake ..
10
11 make -j4
12 sudo make install
```

1.2 安装点云例程依赖的 PCL 库 (可选)

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

1.3 建立 libGL.so 软链接用以解决在 TX1/TX2 上的 bug (可选)

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

2. 编译 SDK

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

2.1 初始化 SDK

注意: 因为设备权限的问题，命令执行完成之后，您必须重新拔插设备(这个操作在同一台电脑上，只需要做一次)。

```
1 make init
```

2.2 编译 SDK

```
1 make all
```

3. 运行例程

Note:: 默认打开矫正后的图像。(跑vio时需要使用原图，跑深度或者点云使用矫正后的图像)

1) get_image 显示左右目的图像和彩色深度图

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

2) get_depth 显示左目的图像，16UC1的深度图和鼠标选中的像素的深度值(mm)

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

3) get_points 显示左目的图像，16UC1的深度图和点云

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

4) get_imu 打印 imu 数据

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

5) get_img_params 打印相机参数并保存在文件中

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

6) get_imu_params 打印 imu 参数并保存在文件中

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

7) get_from_callbacks 使用回调方式获取图像和 imu 数据

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

8) get_all_with_options 使用不同参数打开设备

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

4 安装带有 OpenCV 的 ROS

如果您不使用 ROS(The Robot Operation System), 您可以跳过此部分.

4.1 安装 ROS Kinectic 版本

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

ROS Kinetic 会自动安装 OpenCV, JPEG.

4.2 编译 ROS Wrapper

```
1 make ros
```

Core:

```
1 roscore
```

RViz Display:

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

Publish:

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

Subscribe:

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

4.3 编译内测版设备 ROS Wrapper

```
1 make ros
```

Core:

```
1 roscore
```

RViz Display:

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

Publish:

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

Subscribe:

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

5. 打包

如果打包指定版本OpenCV的包：

```
1 cd <sdk>
2 make cleanall
3 export OpenCV_DIR=<install prefix>
4
5 export OpenCV_DIR=/usr/local
6 export OpenCV_DIR=$HOME/opencv-2.4.13.3
```

Packaging:

```
1 cd <sdk>
2 make pkg
```

6. 清理

```
1 cd <sdk>
2 make cleanall
```

3.3 Windows SDK 用户指南

以下源码编译安装过程。如果只需使用预编译好的库，请参考 Windows 预编译 exe 安装 。

1. 安装编译工具

1.1 安装 Visual Studio

从 https://visualstudio.microsoft.com/ 下载并安装

1.2 安装 CMake

从 https://cmake.org/ 下载并安装

1.3 安装 MSYS2

1) 从 http://mirrors.ustc.edu.cn/msys2/distrib/x86_64/ 下载并安装

2) 将 bin 目录的路径添加到系统变量的 PATH 变量列表中

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

3) 安装 make

```
1 pacman -Syu
2 pacman -S make
```

2. 安装 SDK 依赖

2.1 安装 OpenCV

2.1.1 用预先建立的库安装OpenCV (Recommend)

更多信息您可以参考 *OpenCV* 官方文档 (https://docs.opencv.org/3.4.2/d3/d52/tutorial_windows_install.html)

1) 进入 OpenCV 源码页 http://sourceforge.net/projects/opencvlibrary/files/opencv-win/ 2) 下载一个您想要安装的安装包. 例如 3.4.2/opencv-3.4.2-vc14_vc15.exe 3) 使用管理员权限运行安装包 4) 安装完成之后，设置 OpenCV 环境变量并添加到系统的 path 变量中

2.1.2 设置环境变量

开启 cmd, 输入以下命令:

将 "D:\OpenCV" 替换为您自己的解压缩目录

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

将 OpenCV bin 路径添加到系统环境变量的 PATH 变量列表中

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

2.2 安装 libjpeg-turbo

1) 从 https://sourceforge.net/projects/libjpeg-turbo/files/ 下载 libjpeg-turbo 并安装

2) 将 bin 目录的路径添加到系统变量的 PATH 变量列表中

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

2.3 安装点云例程依赖的 PCL 库 (可选)

从 https://github.com/PointCloudLibrary/pcl/releases 下载集成安装程序(PCL + dependencies)

3. 编译 SDK

打开 "x64 Native Tools Command Prompt for VS 2017"(适用于 VS 2017 的 x64 本机工具命令提示) 命令行界面

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

4. 运行例程

Note: 默认打开矫正后的图像。(跑vio时需要使用原图，跑深度或者点云使用矫正后的图像)

1) get_image 显示左右目的图像和彩色深度图

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

2) get_depth 显示左目的图像，16UC1的深度图和鼠标选中的像素的深度值(mm)

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

3) get_points 显示左目的图像，16UC1的深度图和点云

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

4) get_imu 打印 imu 数据

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

5) get_img_params 打印相机参数并保存在文件中

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

6) get_imu_params 打印 imu 参数并保存在文件中

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

7) get_from_callbacks 使用回调方式获取图像和 imu 数据

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

8) get_all_with_options 使用不同参数打开设备

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

5. 清理

```
1 cd <sdk>
2 make cleanall
```

3.4 Windows 预编译 exe 安装

下载地址： mynteye-d-1.7.1-win-x64-opencv-3.4.3.exe Google Drive, 百度网盘

安装完 SDK 的 exe 安装包后，桌面会生成 SDK 根目录的快捷方式。

进入 "<SDK_ROOT_DIR>\bin\samples" 目录，双击 "get_image.exe" 运行，即可看到相机画面。

生成样例工程

首先，安装好 Visual Studio 2017 https://visualstudio.microsoft.com/ 和 CMake https://cmake.org/ 。

接着，进入 "<SDK_ROOT_DIR>\samples" 目录，双击 "generate.bat" 即可生成样例工程 _build\mynteye_samples.sln 。

如何于 Visual Studio 2017 下使用 SDK

进入 "<SDK_ROOT_DIR>\projects\vs2017"，见 "README.md" 说明。

3.5 ROS 安装

4.1 安装 ROS Kinectic 版本

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

> ROS Kinetic 会自动安装 OpenCV, JPEG.

4.2 编译 ROS Wrapper

```
1 make ros
```

Core:

```
1 roscore
```

RViz Display:

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

Publish:

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

Subscribe:

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

4.3 编译内测版设备 ROS Wrapper

```
1 make ros
```

Core:

```
1 roscore
```

RViz Display:

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

Publish:

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

Subscribe:

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

3.6 ROS 如何使用

按照 ROS 安装 ， 编译再运行节点。

`rostopic list` 可以列出发布的节点：

```
1 /mynteye/depth/camera_info
2 /mynteye/depth/image_raw
3 /mynteye/depth/image_raw/compressed
4 /mynteye/depth/image_raw/compressed/parameter_descriptions
5 /mynteye/depth/image_raw/compressed/parameter_updates
6 /mynteye/depth/image_raw/compressedDepth
7 /mynteye/depth/image_raw/compressedDepth/parameter_descriptions
8 /mynteye/depth/image_raw/compressedDepth/parameter_updates
9 /mynteye/depth/image_raw/theora
10 /mynteye/depth/image_raw/theora/parameter_descriptions
11 /mynteye/depth/image_raw/theora/parameter_updates
12 /mynteye/imu/data_raw
13 /mynteye/imu/data_raw_processed
14 /mynteye/left/camera_info
15 /mynteye/left/image_color
16 /mynteye/left/image_color/compressed
17 ...
```

`rostopic hz <topic>` 可以检查是否有数据：

```
1 subscribed to [/mynteye/imu/data_raw]
2 average rate: 202.806
3   min: 0.000s max: 0.021s std dev: 0.00819s window: 174
4 average rate: 201.167
5   min: 0.000s max: 0.021s std dev: 0.00819s window: 374
6 average rate: 200.599
7   min: 0.000s max: 0.021s std dev: 0.00819s window: 574
8 average rate: 200.461
9   min: 0.000s max: 0.021s std dev: 0.00818s window: 774
10 average rate: 200.310
11   min: 0.000s max: 0.021s std dev: 0.00818s window: 974
12   ...
```

`rostopic echo <topic>` 可以打印发布数据等。了解更多，请阅读 `rostopic`。

ROS 封装的文件结构，如下所示：

```
<sdk>/wrappers/ros/
├──src/
   └──mynteye_wrapper_d/
      ├──launch/
      │   ├──display.launch
      │   └──mynteye.launch
      ├──msg/
      ├──rviz/
      ├──src/
       │   ├──mynteye_listener.cc
       │   └──mynteye_wrapper_nodelet.cc
       │   └──mynteye_wrapper_node.cc
       │   └──pointcloud_generatort.cc
       │   └──pointcloud_generator.h
      ├──CMakeLists.txt
      ├──nodelet_plugins.xml
       └──package.xml
```

其中 `mynteye.launch` 里，可以配置发布的 topics 与 frame_ids 、决定启用哪些数据、以及设定控制选项。其中，`gravity` 请配置成当地重力加速度。

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

Chapter 4

SDK 样例

4.1 获取双目图像

API 通过 `DeviceMode::DEVICE_COLOR` 参数获取图像数据，或者 `DeviceMode::DEVICE_ALL` 同时捕获图像和深度数据。

通过 `GetStreamData()` 函数，就能获取想要的数据。

参考代码片段：

```
1 // Device mode, default DEVICE_ALL
2 //   DEVICE_COLOR: IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH n
3 //   DEVICE_DEPTH: IMAGE_LEFT_COLOR n IMAGE_RIGHT_COLOR n IMAGE_DEPTH y
4 //   DEVICE_ALL:   IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH y
5 // Note: y: available, n: unavailable, -: depends on #stream_mode
6 params.dev_mode = DeviceMode::DEVICE_DEPTH;
7
8
9 auto left_color = cam.GetStreamData(ImageType::IMAGE_LEFT_COLOR);
10     if (left_color.img) {
11     cv::Mat left = left_color.img->To(ImageFormat::COLOR_BGR)->ToMat();
12     painter.DrawSize(left, CVPainter::TOP_LEFT);
13     painter.DrawStreamData(left, left_color, CVPainter::TOP_RIGHT);
14     painter.DrawInformation(left, util::to_string(counter.fps()),
15         CVPainter::BOTTOM_RIGHT);
16     cv::imshow("left color", left);
```

完整代码样例，请见get_image.cc 。

4.2 获取深度图像

深度图像，属于上层合成数据。

可以通过设置depth_mode来改变深度图显示。

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

然后使用GetStreamData（）获取。另外，判断不为空后再使用。

参考代码片段：

```
1 auto image_depth = cam.GetStreamData(ImageType::IMAGE_DEPTH);
2 if (image_depth.img) {
3   cv::Mat depth = image_depth.img->To(ImageFormat::DEPTH_RAW)->ToMat();
4
5   cv::setMouseCallback("depth", OnDepthMouseCallback, &depth_region);
6   // Note: DrawRect will change some depth values to show the rect.
7   depth_region.DrawRect(depth);
8   cv::imshow("depth", depth);
9
10   depth_region.ShowElems<ushort>(depth, [](const ushort& elem) {
11     return std::to_string(elem);
12   }, 80, depth_info);
13 }
```

上述代码，用了 OpenCV 来显示图像。选中显示窗口时，按 ESC/Q 就会结束程序。

完整代码样例，请见get_depth.cc 。

4.3 获取点云图像

点云图像，属于上层合成数据。API使用GetStreamData（）获取。另外，判断不为空后再使用。

参考代码片段：

```
1 auto image_color = cam.GetStreamData(ImageType::IMAGE_LEFT_COLOR);
2 auto image_depth = cam.GetStreamData(ImageType::IMAGE_DEPTH);
3 if (image_color.img && image_depth.img) {
4     cv::Mat color = image_color.img->To(ImageFormat::COLOR_BGR)
5         ->ToMat();
6     painter.DrawSize(color, CVPainter::TOP_LEFT);
7     painter.DrawStreamData(color, image_color, CVPainter::TOP_RIGHT);
8     painter.DrawInformation(color, util::to_string(counter.fps()),
9         CVPainter::BOTTOM_RIGHT);
10
11     cv::Mat depth = image_depth.img->To(ImageFormat::DEPTH_RAW)
12         ->ToMat();
13
14     cv::imshow("color", color);
15
16     viewer.Update(color, depth);
17 }
```

上述代码，用了 PCL 来显示点云。关闭点云窗口时，也会结束程序。

完整代码样例，请见get_points.cc。

4.4 获取IMU数据

使用EnableMotionDatas()来启用缓存，才能通过GetMotionDatas()函数来获取到IMU数据。否则，只能通过回调接口得到IMU数据，请参阅（从回调接口获取数据）[]。

参考代码片段：

```
1 auto motion_datas = cam.GetMotionDatas();
2 if (motion_datas.size() > 0) {
3     std::cout << "Imu count: " << motion_datas.size() << std::endl;
4     for (auto data : motion_datas) {
5         if (data.imu) {
6             if (data.imu->flag == MYNTEYE_IMU_ACCEL) {
7                 counter.IncrAccelCount();
8                 std::cout << "[accel] stamp: " << data.imu->timestamp
9                     << ", x: " << data.imu->accel[0]
10                     << ", y: " << data.imu->accel[1]
11                     << ", z: " << data.imu->accel[2]
12                     << ", temp: " << data.imu->temperature
13                     << std::endl;
14             } else if (data.imu->flag == MYNTEYE_IMU_GYRO) {
15                 counter.IncrGyroCount();
16                 std::cout << "[gyro] stamp: " << data.imu->timestamp
17                 << ", x: " << data.imu->gyro[0]
18                 << ", y: " << data.imu->gyro[1]
19                 << ", z: " << data.imu->gyro[2]
20                 << ", temp: " << data.imu->temperature
21                 << std::endl;
22             } else {
23                 std::cerr << "Imu type is unknown" << std::endl;
24             }
25         } else {
26           std::cerr << "Motion data is empty" << std::endl;
27         }
28     }
29     std::cout << std::endl;
30 }
```

上述代码，用了 OpenCV 来显示图像和数据。选中显示窗口时，按 ESC/Q 就会结束程序。

完整代码样例，请见get_imu.cc 。

4.5 从回调接口获取数据

API提供了SetStreamCallback()，SetMotionCallback()函数，来设定各类数据的回调。

参考代码片段：

```
1 cam.SetImgInfoCallback([](const std::shared_ptr<ImgInfo>& info) {
2     std::cout << "  [img_info] fid: " << info->frame_id
3         << ", stamp: " << info->timestamp
4         << ", expos: " << info->exposure_time << std::endl
5         << std::flush;
6 });
7 for (auto&& type : types) {
8     // Set stream data callback
9     cam.SetStreamCallback(type, [](const StreamData& data) {
10     std::cout << "  [" << data.img->type() << "] fid: "
11         << data.img->frame_id() << std::endl
12         << std::flush;
13     });
14 }
15
16 // Set motion data callback
17 cam.SetMotionCallback([](const MotionData& data) {
18     if (data.imu->flag == MYNTEYE_IMU_ACCEL) {
19         std::cout << "[accel] stamp: " << data.imu->timestamp
20         << ", x: " << data.imu->accel[0]
21         << ", y: " << data.imu->accel[1]
22         << ", z: " << data.imu->accel[2]
```

```
23         << ", temp: " << data.imu->temperature
24         << std::endl;
25     } else if (data.imu->flag == MYNTEYE_IMU_GYRO) {
26         std::cout << "[gyro] stamp: " << data.imu->timestamp
27         << ", x: " << data.imu->gyro[0]
28         << ", y: " << data.imu->gyro[1]
29         << ", z: " << data.imu->gyro[2]
30         << ", temp: " << data.imu->temperature
31         << std::endl;
32     }
33     std::cout << std::flush;
34 });
```

上述代码，用了 OpenCV 来显示图像和数据。选中显示窗口时，按 ESC/Q 就会结束程序。

完整代码样例，请见 get_from_callbacks.cc。

4.6 通过设置参数获取不同类型的数据

get_all_with_options 样例可以通过添加参数来设定当前设备的各类控制值。

get_all_with_options -h 参数说明：

```
1 Open device with different options.
2
3 Options:
4   -h, --help            show this help message and exit
5   -m, --imu             Enable imu datas
6
7   Open Params:
8     The open params
9
10    -i INDEX, --index=INDEX
11                        Device index
12    -f RATE, --rate=RATE
13                        Framerate, range [0,60], [30](STREAM_2560x720),
14                        default: 10
15    --dev-mode=MODE       Device mode, default 2 (DEVICE_ALL)
16                          0: DEVICE_COLOR, left y right - depth n
17                          1: DEVICE_DEPTH, left n right n depth y
18                          2: DEVICE_ALL,   left y right - depth y
19                          Note: y: available, n: unavailable, -: depends on
20                        stream mode
21    --cm=MODE           Color mode, default 0 (COLOR_RAW)
22                          0: COLOR_RAW, color raw
23                          1: COLOR_RECTIFIED, color rectified
24    --dm=MODE           Depth mode, default 2 (DEPTH_COLORFUL)
25                          0: DEPTH_RAW
26                          1: DEPTH_GRAY
27                          2: DEPTH_COLORFUL
28    --sm=MODE           Stream mode of color & depth,
29                        default 2 (STREAM_1280x720)
30                          0: STREAM_640x480, 480p, vga, left
31                          1: STREAM_1280x480, 480p, vga, left+right
32                          2: STREAM_1280x720, 720p, hd, left
33                          3: STREAM_2560x720, 720p, hd, left+right
34    --csf=MODE          Stream format of color,
35                        default 1 (STREAM_YUYV)
36                          0: STREAM_MJPG
37                          1: STREAM_YUYV
38    --dsf=MODE          Stream format of depth,
39                        default 1 (STREAM_YUYV)
40                          1: STREAM_YUYV
41    --ae                Enable auto-exposure
42    --awb               Enable auto-white balance
43    --ir=VALUE          IR intensity, range [0,6], default 0
44    --ir-depth          Enable ir-depth-only
45
46   Feature Toggles:
47     The feature toggles
48
49    --proc=MODE         Enable process mode, e.g. imu assembly, temp_drift
50                          0: PROC_NONE
51                          1: PROC_IMU_ASSEMBLY
52                          2: PROC_IMU_TEMP_DRIFT
53                          3: PROC_IMU_ALL
54    --img-info          Enable image info, and sync with image
```

例如 `./samples/_output/bin/get_all_with_options -f 60 --dev-mode=0 --sm=2` 显示的是1280x720的60帧左目未矫正图像。

完整代码样例 get_all_with_options 。

4.7 获取图像标定参数

通过获取API `GetStreamIntrinsics()`,`GetStreamExtrinsics()`函数，可以获取当前打开设备的图像标定参数。

参考代码片段：

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

运行结果保存在当前目录下，参考运行结果：

```
1 VGA Intrinsics left: {width: [640], height: [480], fx: [358.45721435546875000], fy:
       [359.53115844726562500], cx: [311.12109375000000000], cy: [242.63494873046875000]coeffs: [-0.28297042846679688,
       0.06178283691406250, -0.00030517578125000, 0.00218200683593750, 0.00000000000000000]}
2 VGA Intrinsics right: {width: [640], height: [480], fx: [360.13885498046875000], fy:
       [360.89624023437500000], cx: [325.11029052734375000], cy: [251.46371459960937500]coeffs: [-0.30667877197265625,
       0.08611679077148438, -0.00030136108398438, 0.00155639648437500, 0.00000000000000000]}
3 VGA Extrinsics left to right: {rotation: [0.99996054172515869, 0.00149095058441162, 0.00875246524810791,
       -0.00148832798004150, 0.99999880790710449, -0.00030362606048584, -0.00875294208526611, 0.00029063224792480,
       0.99996161460876465], translation: [-120.36341094970703125, 0.00000000000000000, 0.00000000000000000]}
```

完整代码样例，请见get_img_params.cc 。

4.8 获取IMU标定参数

通过API `GetMotionIntrinsics()`,`GetMotionExtrinsics`函数，可以获取当前打开设备的IMU标定参数。

参考代码片段：

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

运行结果保存在当前目录下，参考运行结果：

```
1 Motion Intrinsics: {accel: {scale: [1.00205999990004191, 0.00000000000000000, 0.00000000000000000,
       0.00000000000000000, 1.00622999999999996, 0.00000000000000000, 0.00000000000000000, 0.00000000000000000,
       1.00171999999999994], assembly: [1.00000000000000000, 0.00672262000000000, -0.00364474000000000, 0.00000000000000000,
       1.00000000000000000, 0.00101348000000000, -0.00000000000000000, 0.00000000000000000, 1.00000000000000000,
       1.00000000000000000], drift: [0.00000000000000000, 0.00000000000000000, 0.00000000000000000], noise:
       [0.00000000000000000, 0.00000000000000000, 0.00000000000000000], bias: [0.00000000000000000, 0.00000000000000000,
       0.00000000000000000], x: [0.00856165620000000, -0.00009840052800000], y: [0.05968393300000000,
       -0.00130967680000000], z: [0.01861442050000000, -0.00016033523000000]}, gyro: {scale: [1.00008999999999992,
       0.00000000000000000, 0.00000000000000000, 0.00000000000000000, 0.99617599999999995, 0.00000000000000000, 0.000000000000
       0.00000000000000000, 1.00407000000000002], assembly: [1.00000000000000000, -0.00700362000000000,
       -0.00326206000000000, 0.00549571000000000, 1.00000000000000000, 0.00224867000000000, 0.00236088000000000,
       0.00044507800000000, 1.00000000000000000, 1.00000000000000000], drift: [0.00000000000000000, 0.00000000000000000,
       0.00000000000000000], noise: [0.00000000000000000, 0.00000000000000000, 0.00000000000000000], bias:
       [0.00000000000000000, 0.00000000000000000, 0.00000000000000000], x: [0.18721455299999998, 0.00077411070000000], y:
       [0.60837032000000002, -0.00939702710000000], z: [-0.78549276000000001, 0.02584820200000000]}}
```

完整代码样例，请见get_imu_params.cc 。

4.9 设定打开参数

设定图像分辨率

通过设置 params.stream_mode 参数，就可以设定图像的分辨率。

注意

- 图像分辨率现在支持4种: 单目640X480，1280x720 和双目1280x480， 2560x720

参考代码片段:

```
1 // Stream mode: left color only
2 // params.stream_mode = StreamMode::STREAM_640x480;  // vga
3 // params.stream_mode = StreamMode::STREAM_1280x720;  // hd
4 // Stream mode: left+right color
5 // params.stream_mode = StreamMode::STREAM_1280x480;  // vga
6 params.stream_mode = StreamMode::STREAM_2560x720;  // hd
```

设定图像帧率

通过设置 params.framerate 参数，就可以设定图像的帧率。

注意

- 图像帧率有效值(0-60)
- 分辨率在2560X720时帧率有效值为(30)

参考代码片段:

```
1 // Framerate: 10(default), [0,60], [30](STREAM_2560x720)
2 params.framerate = 30;
```

设定图像模式

通过 params.color_mode 参数，就可以设定图像的模式。

COLOR_RAW 为原图， COLOR_RECTIFIED 为矫正图。

参考代码片段:

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

设定深度图模式

通过 params.depth_mode 参数，就可以设定深度图的模式。

DEPTH_COLORFUL 为着色后的深度图，DEPTH_GRAY 为灰色深度图， DEPTH_GRAY 为原始深度图。

参考代码片段:

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

启用自动曝光及自动白平衡

通过设置 params.state_ae 和 params.state_awb 为 true ，就可以启动自动曝光和自动白平衡。

默认自动曝光和自动白平衡是启用的，如果想关闭，可以设置参数值为 false 。

参考代码片段：

```
1 // Auto-exposure: true(default), false
2 // params.state_ae = false;
3
4 // Auto-white balance: true(default), false
5 // params.state_awb = false;
```

启用IR及其调节

通过设置 params.ir_intensity 参数，就可以设定图像的IR强度。

启用IR，就是设定 params.ir_intensity 大于0的值。值越大，强度越高(最大为10)。

参考代码片段：

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

启用IR Depth Only

通过设置 params.ir_depth_only 参数，就可以设定IR Depth Only功能。默认关闭。

> 注意
>
> - 15帧下此功能不生效

参考代码片段：

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

调整深度图着色值

通过设置 `params.colour_depth_value` 参数, 默认值是 1000。

参考代码片段:

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

以上功能参考运行结果，于 Linux 上：

```
1 Open device: 0, /dev/video1
2
3 D/eSPDI_API: SetPropertyValue control=7 value=0D/eSPDI_API: SetPropertyValue control=7 value=35D/eSPDI_API:
       SetPropertyValue control=7 value=1-- Auto-exposure state: enabled
4 D/eSPDI_API: SetPropertyValue control=7 value=0D/eSPDI_API: SetPropertyValue control=7 value=12D/eSPDI_API:
       SetPropertyValue control=7 value=1-- Auto-white balance state: enabled
5 -- Framerate: 5
6 D/eSPDI_API: SetPropertyValue control=7 value=4 SetDepthDataType: 4
7 -- Color Stream: 1280x720 YUYV
8 -- Depth Stream: 1280x720 YUYV
9
10 D/eSPDI_API: SetPropertyValue control=7 value=0D/eSPDI_API: SetPropertyValue control=7 value=3D/eSPDI_API:
       SetPropertyValue control=7 value=4
11 -- IR intensity: 4
12 D/eSPDI_API: CVideoDevice::OpenDevice 1280x720 fps=5
13
14 Open device success
```

完整代码样例 `get_image` 。

4.10 相机控制参数API

打开或关闭自动曝光

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

打开或关闭自动白平衡

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

设置 IR 强度

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

设置全局增益

注意:: 需要关闭自动曝光

```
1 /** Set global gain [1 - 16]
2  * value -- global gain value
3  * */
4 void SetGlobalGain(const float &value);    see "camera.h"
```

设置曝光时间

注意:: 需要关闭自动曝光

```
1 /** Set exposure time [1ms - 2000ms]
2  * value -- exposure time value
3  * */
4 void SetExposureTime(const float &value);    see "camera.h"
```

Chapter 5

SDK 工具

5.1 分析IMU数据

SDK 提供了 IMU 数据分析工具 imu_analytics.py. 工具的详细信息见 tools/README.md

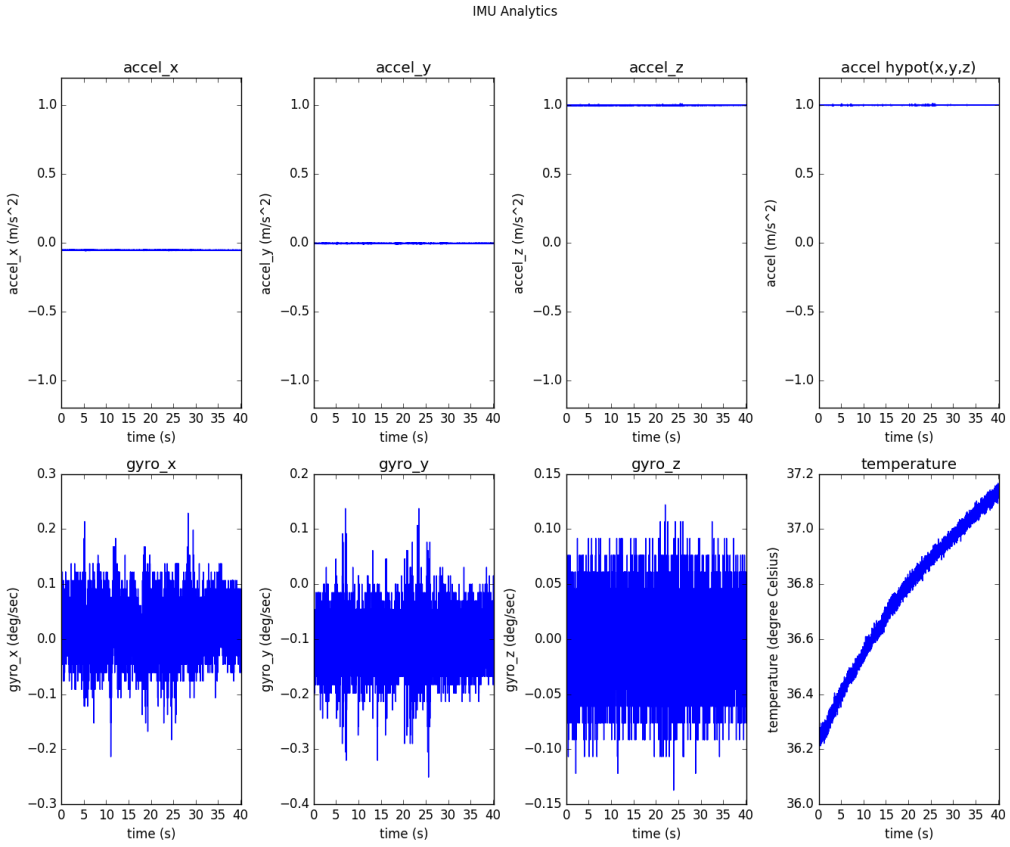
Linux 系统运行命令:

```
1 $ 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=
```

Linux 系统上的结果参考:

```
1 $ 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=
2 imu analytics ...
3   input: dataset
4   outdir: dataset
5   gyro_limits: None
6   accel_limits: [(-1.2, 1.2), (-1.2, 1.2), (-1.2, 1.2), (-1.2, 1.2)]
7   time_unit: None
8   time_limits: None
9   auto: False
10   gyro_show_unit: d
11   gyro_data_unit: d
12   temp_limits: None
13 open dataset ...
14   imu: 20040, temp: 20040
15   timebeg: 4.384450, timeend: 44.615550, duration: 40.231100
16 save figure to:
17   dataset/imu_analytics.png
18 imu analytics done
```

分析结果图保存在 dataset 目录中. 如下:



另外，可以使用 -h 参数查看工具详细参数选项.

```
1 $ python tools/analytics/imu_analytics.py -h
```

Copyright 2018. MYNTEYE

5.2 分析时间戳

SDK 提供了时间戳分析工具 stamp_analytics.py. 工具的详细信息见 tools/README.

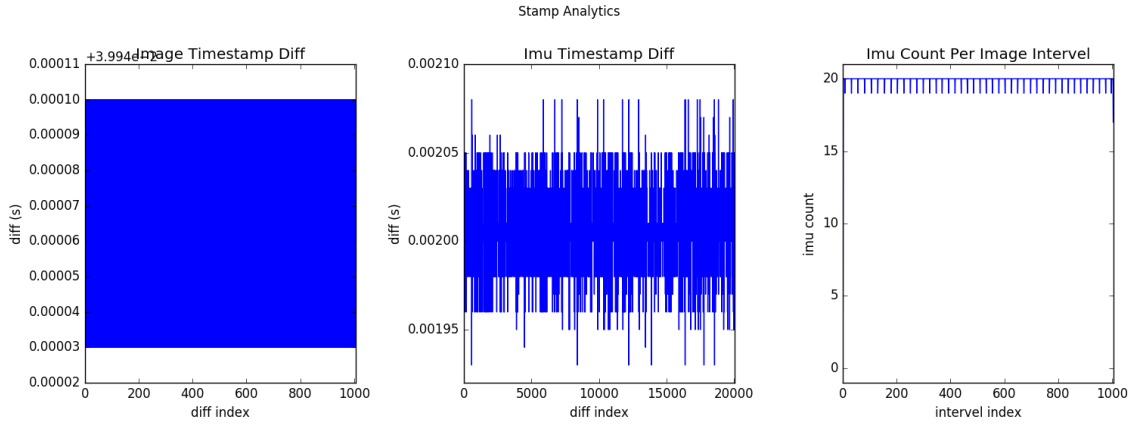
Linux 系统运行命令:

```
1 $ python tools/analytics/stamp_analytics.py -i dataset -c tools/config/mynteye/mynteye_config.yaml
```

Linux 系统上的结果参考:

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

rate (Hz)
  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
```

分析结果图保存在 dataset 目录中. 如下:



另外，可以使用 -h 参数查看工具详细参数选项.

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

Copyright 2018. MYNTEYE

5.3 录制数据集

SDK 提供了录制数据集的工具 record. 工具的详细信息见 tools/README.md

Linux 系统运行命令:

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

Windows 系统运行命令:

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

Linux 系统上的结果参考:

```
1 $ ./tools/_output/bin/dataset/record
2 Saved 1007 imgs, 20040 imus to ./dataset
3 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
4 I0513 21:29:38.608853 11487 record.cc:121] Img count: 1007, fps: 24.9732
5 I0513 21:29:38.608873 11487 record.cc:123] Imu count: 20040, hz: 496.983
```

结果默认保存在 `<workdir>/dataset` 中. 您也可以使用参数指定自定义目录存放结果.

录制结果目录详情:

```
1 <workdir>/
2 └──dataset/
3     ├──left/
4     │   ├──stream.txt  # Image infomation
5     │   ├──000000.png  # Image, index 0
6     │   └──...
7     ├──right/
8     │   ├──stream.txt  # Image information
9     │   ├──000000.png  # Image, index 0
10    │   └──...
11    └──motion.txt  # IMU information
```


5.4 保存设备信息和参数

SDK 提供了保存信息和参数的工具 `save_all_infos` 。

参考运行命令：

```
1 ./tools/_output/bin/writer/save_all_infos
2
3 # Windows
4 .\tools\_output\bin\writer\save_all_infos.bat
```

参考运行结果，于 Linux 上：

```
1 I/eSPDI_API: eSPDI: EtronDI_Init
2 Device descriptors:
3   name: MYNT-EYE-D1000
4   serial_number: 203837533548500F002F0028
5   firmware_version: 1.0
6   hardware_version: 2.0
7   spec_version: 1.0
8   lens_type: 0000
9   imu_type: 0000
10   nominal_baseline: 120
```

默认会保存进 `<workdir>/config` 目录。你也可以加参数，指定保存到其他目录。

保存内容如下：

```
1 <workdir>/
2 └──config/
3     └──SN0610243700090720/
4        ├──device.info
5        └──imu.params
```

完整代码样例 `save_all_infos` 。

5.5 写入IMU标定参数

SDK提供了写入IMU标定参数的工具 `imu_params_writer` 。

有关如何获取, 请阅读 [获取IMU标定参数]()。

参考运行命令:

```
1 ./tools/_output/bin/writer/imu_params_writer tools/writer/config/imu.params
2
3 # Windows
4 .\tools\_output\bin\writer\imu_params_writer.bat tools\writer\config\imu.params
```

其中，`tools/writer/config/imu.params` 是参数文件路径。如果你自己标定了参数，可以编辑此文件，然后执行上述命令写入设备。

> 警告
>
> - 请不要随意覆写参数。另外 `save_all_infos` 工具可帮你备份参数。

完整代码样例 `imu_params_writer` 。

5.6 升级 HID 设备固件

获取固件

最新固件： mynteye-d-hid-firmware-1.0.bin `Google Drive`, 百度网盘

编译 SDK 工具

```
1 cd <sdk>
2 make tools
```

升级固件

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

5.7 升级相机固件

注意:: 此工具不支持内测版设备升级

获取固件

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

获取升级工具

Latest tool: EtronWriter_1.0.6.zip `Google Drive`, `Baidu Pan`

升级固件

注意:: 请严格按照步骤升级固件.(否则可能会丢失相机标定参数)

1, 选择相机设备.

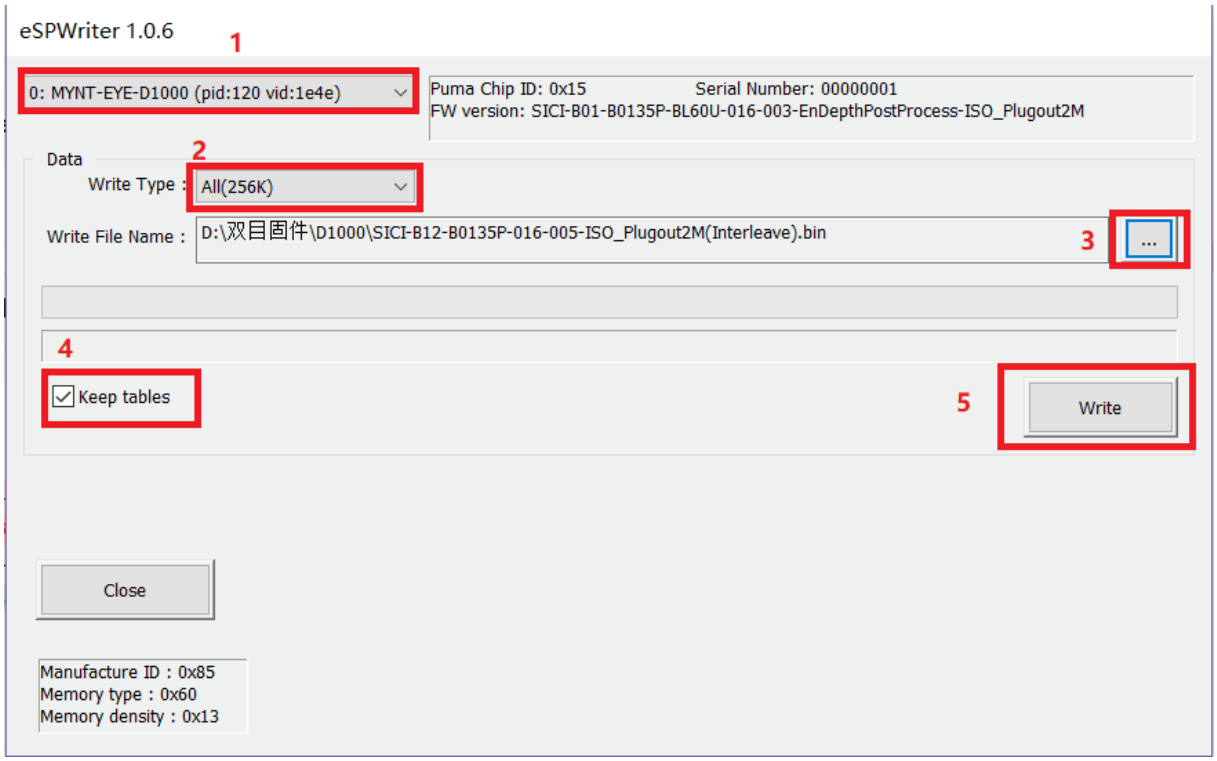
2, 选择数据类型(256KB).

3, 选择相机固件.

4, 选择 `Keep tables` (保留相机标定参数).

5, 点击 `Write`.

参考图示使用工具:



Chapter 6

工程样例

- Visual Studio 2017 如何使用 SDK
- Qt Creator 如何使用 SDK
- CMake 如何使用 SDK

6.1 Visual Studio 2017 如何使用 SDK

本教程将使用 Visual Studio 2017 创建一个项目来使用 SDK 。

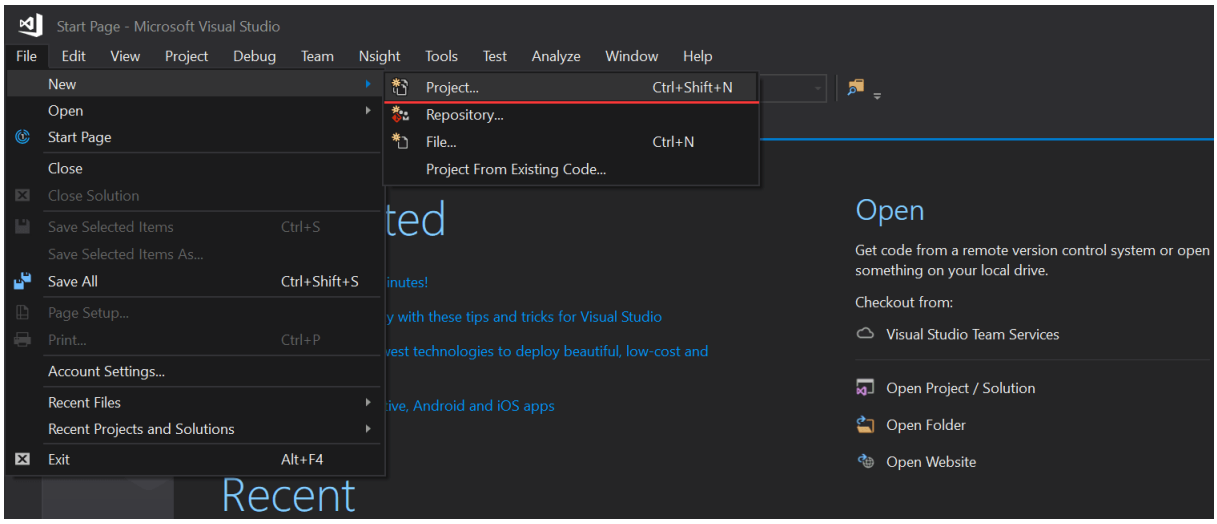
> 你可以在 `<sdk>/platforms/projects/vs2017` 目录下找到工程样例。

准备

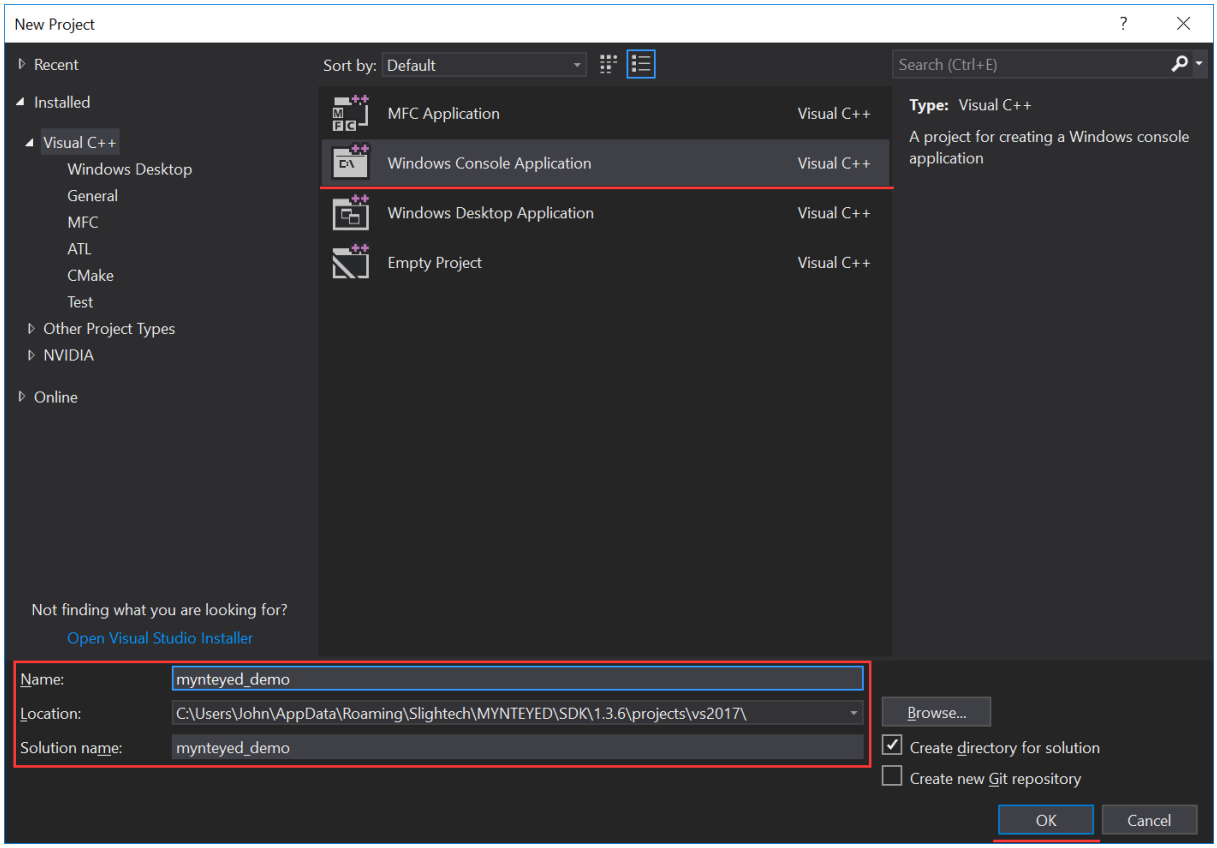
- Windows: 安装 SDK 的 exe 包

创建项目

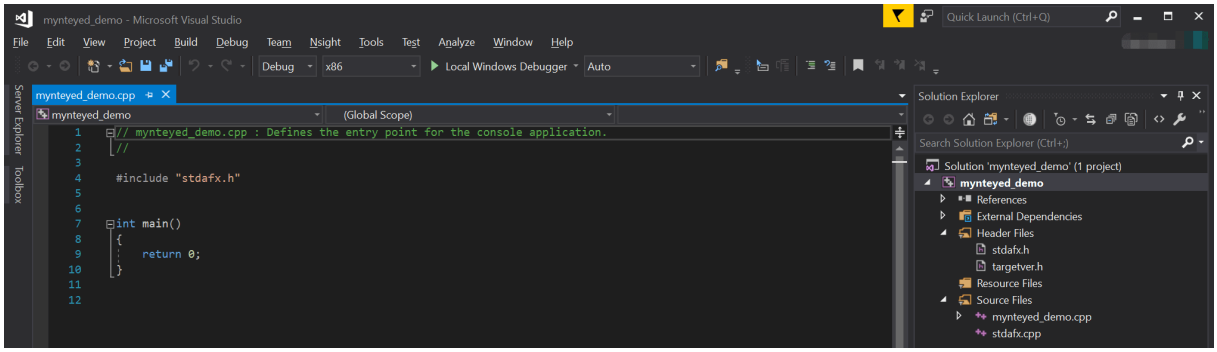
打开 Visual Studio 2017 ， 然后 `File > New > Project`,



选择 "Windows Console Application"，设置项目位置和名字，点击 "OK"，

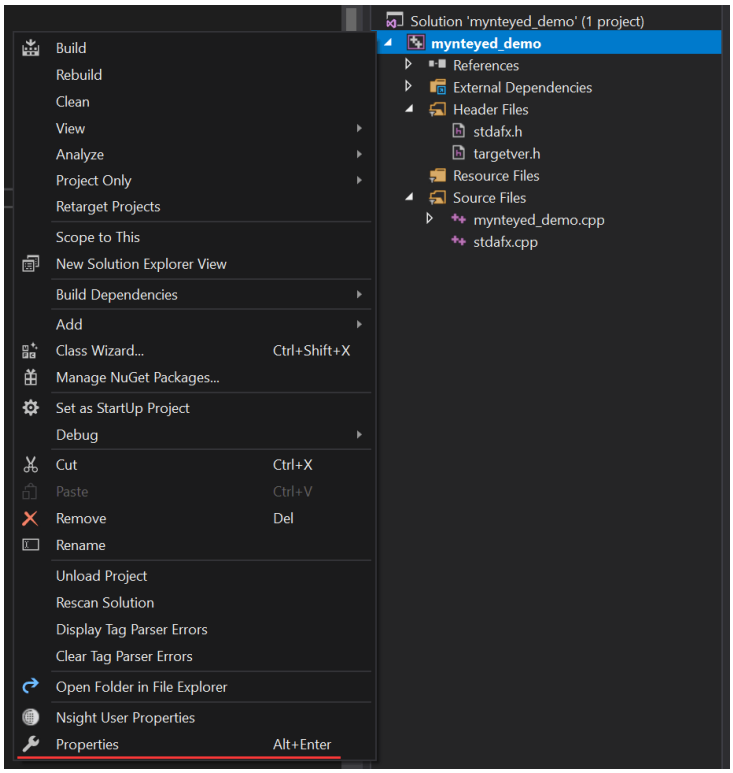


最后，你可以看到一个新的项目被创建，



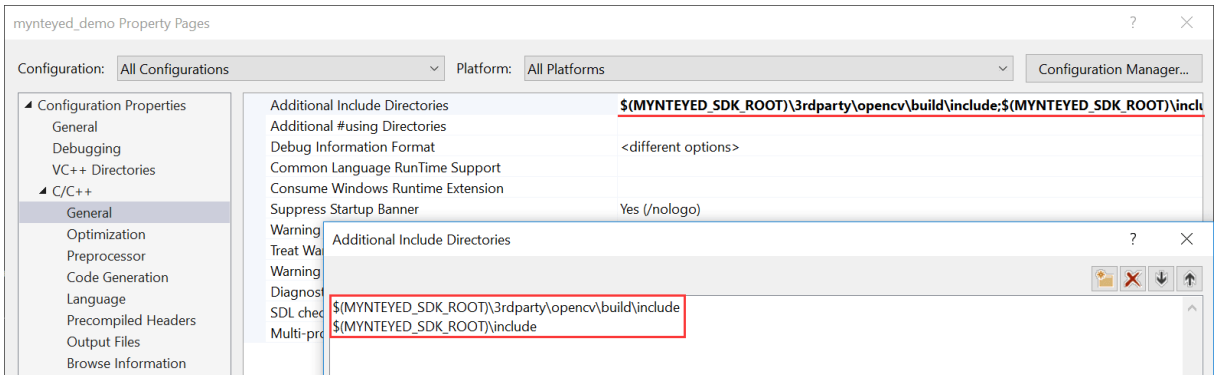
配置项目

右键点击该项目，打开 "Properties" 窗口，



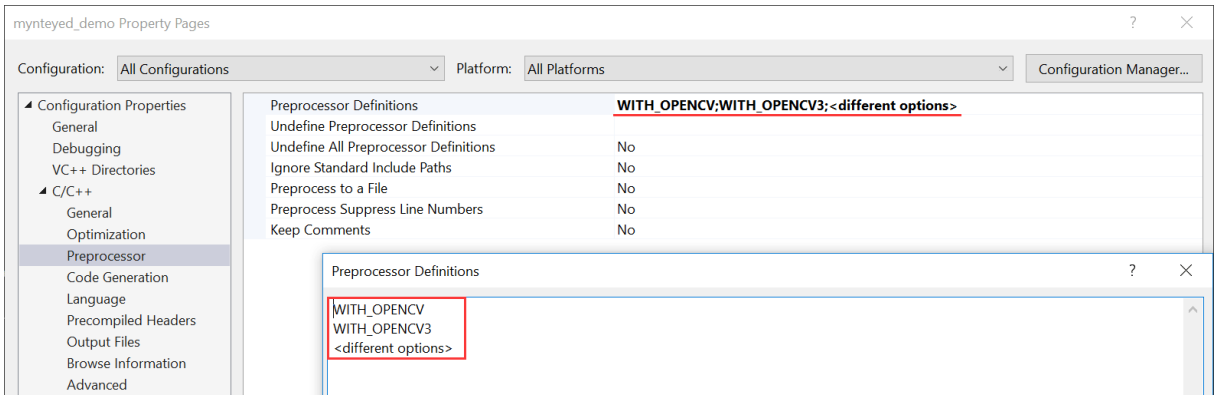
将 "Configuration" 更改为 "All Configurations"， 然后添加以下路径到 "Additional Include Directories"，

```
1 $(MYNTEYED_SDK_ROOT)\include
2 $(MYNTEYED_SDK_ROOT)\3rdparty\opencv\build\include
```



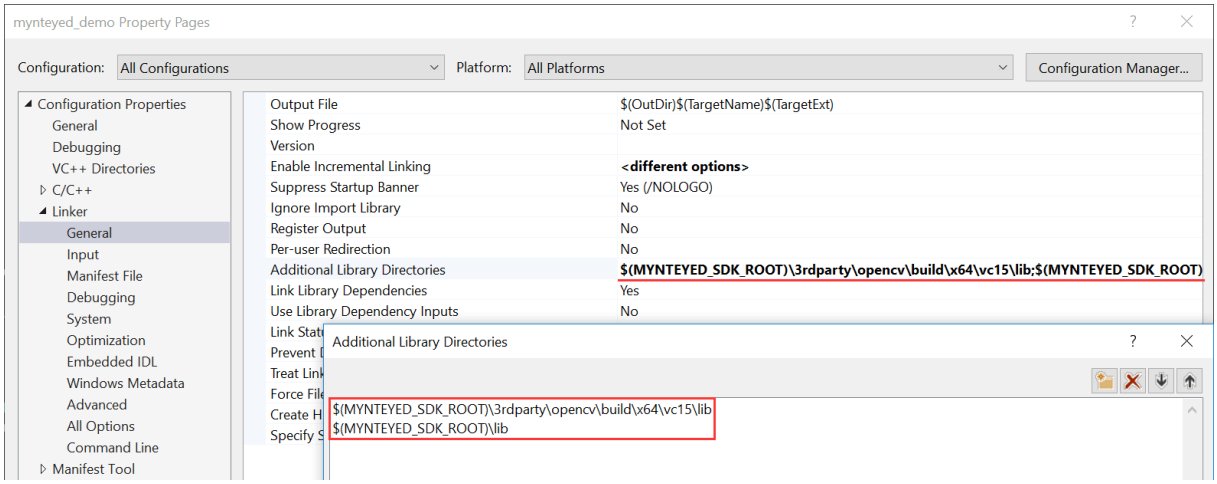
添加以下定义到 "Preprocessor Definitions",

```
1 WITH_OPENCV
2 WITH_OPENCV3
```



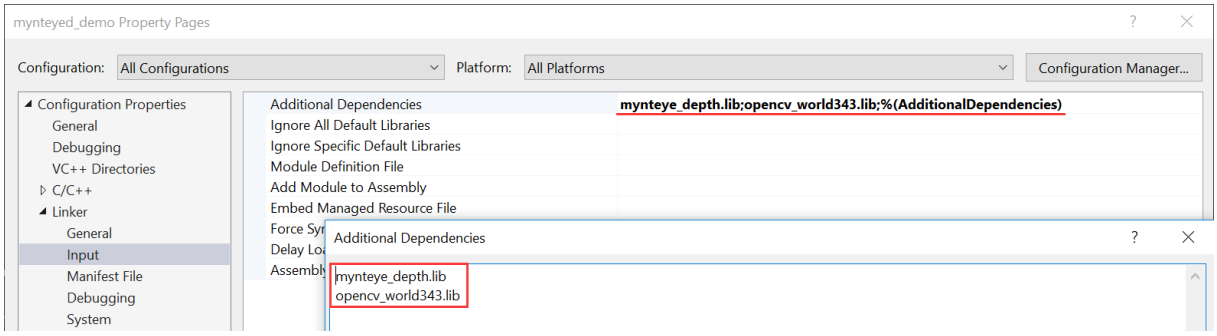
添加以下路径到 "Additional Library Directories"，

```
1 $(MYNTEYED_SDK_ROOT)\lib
2 $(MYNTEYED_SDK_ROOT)\3rdparty\opencv\build\x64\vc15\lib
```



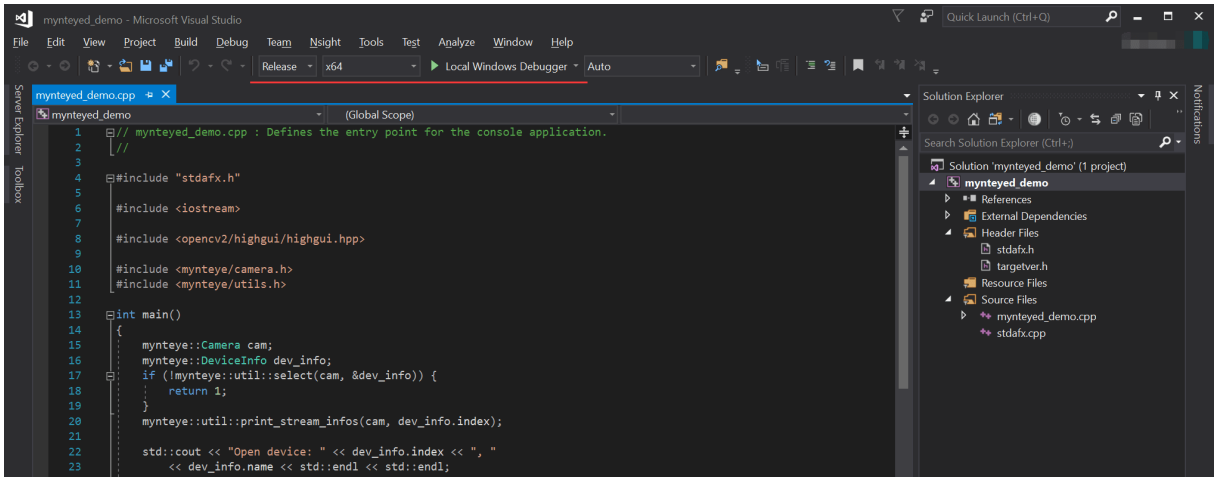
添加以下库到 "Additional Dependencies"，

```
1 mynteye_depth.lib
2 opencv_world343.lib
```



使用SDK

添加头文件和使用 API，



选择 "Release x64" 来运行项目。

6.2 Qt Creator 如何使用 SDK

该教程将会使用 Qt creator 创建 Qt 项目来运行 SDK 。

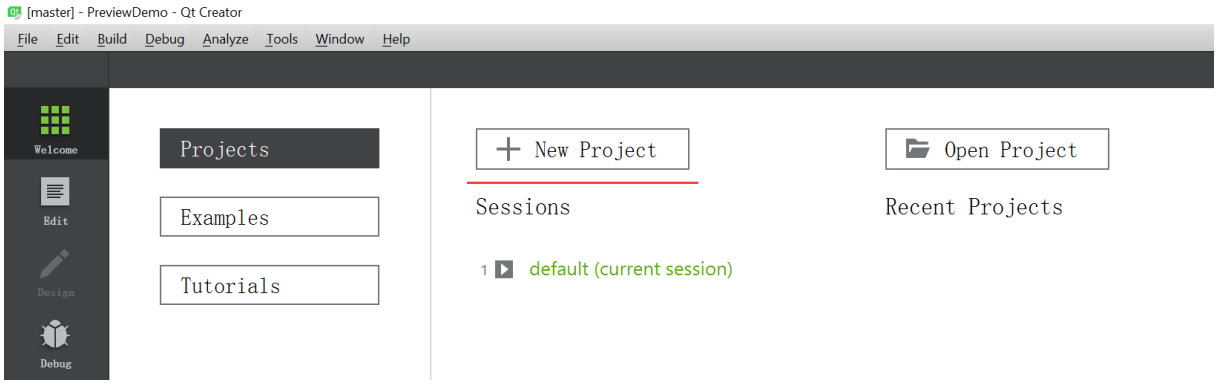
> 你可以在 <sdk>/platforms/projects/qtcreator 目录下找到工程样例。

准备

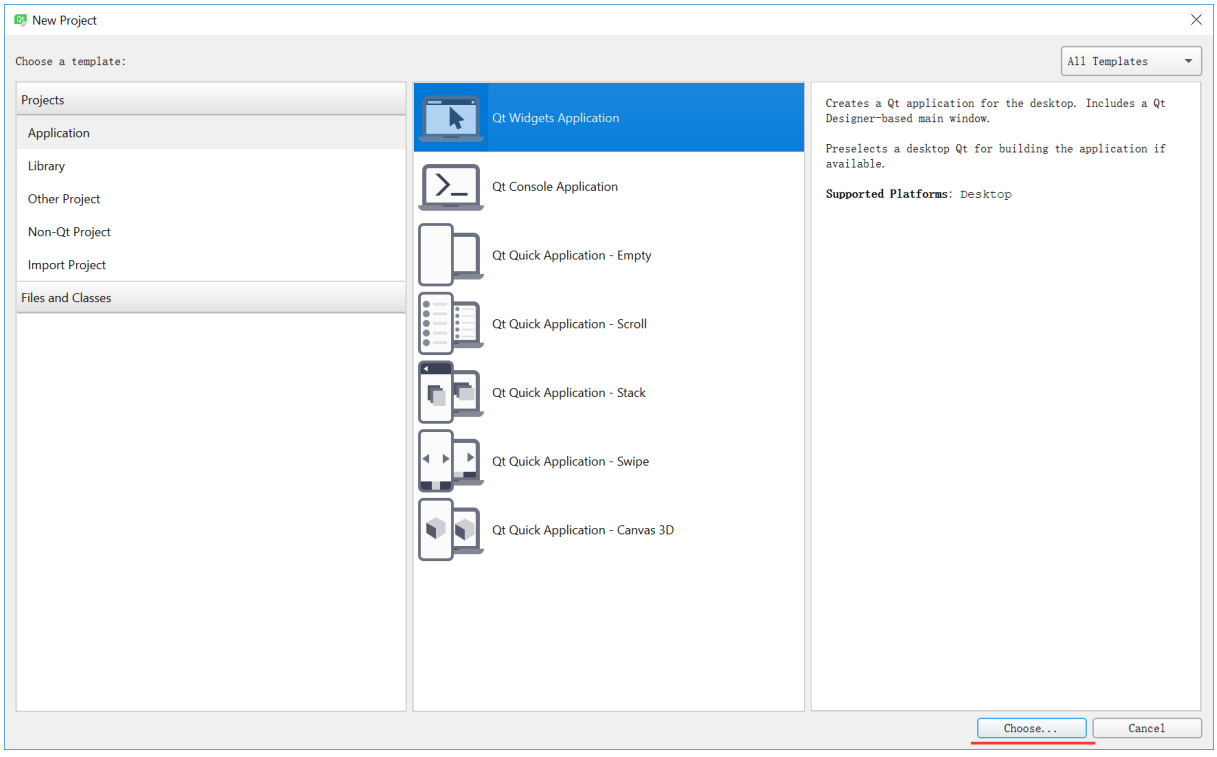
- Windows: 安装 SDK 的 exe 包
- Linux: 使用源代码编译和 make install

创建项目

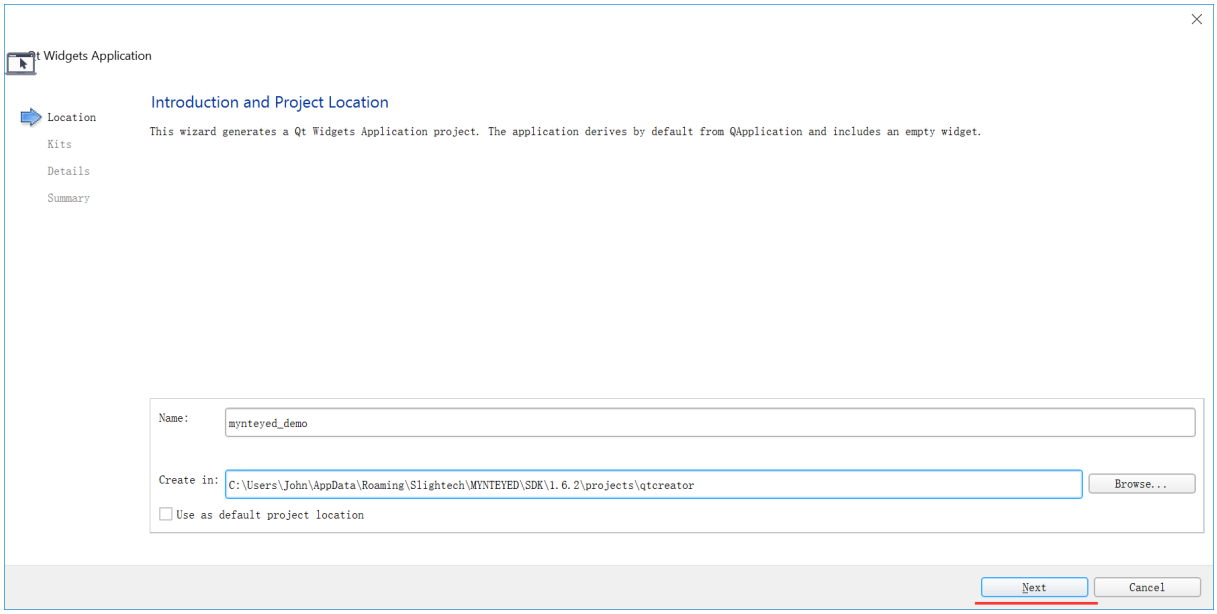
打开 Qt Creator ， 然后 New Project，



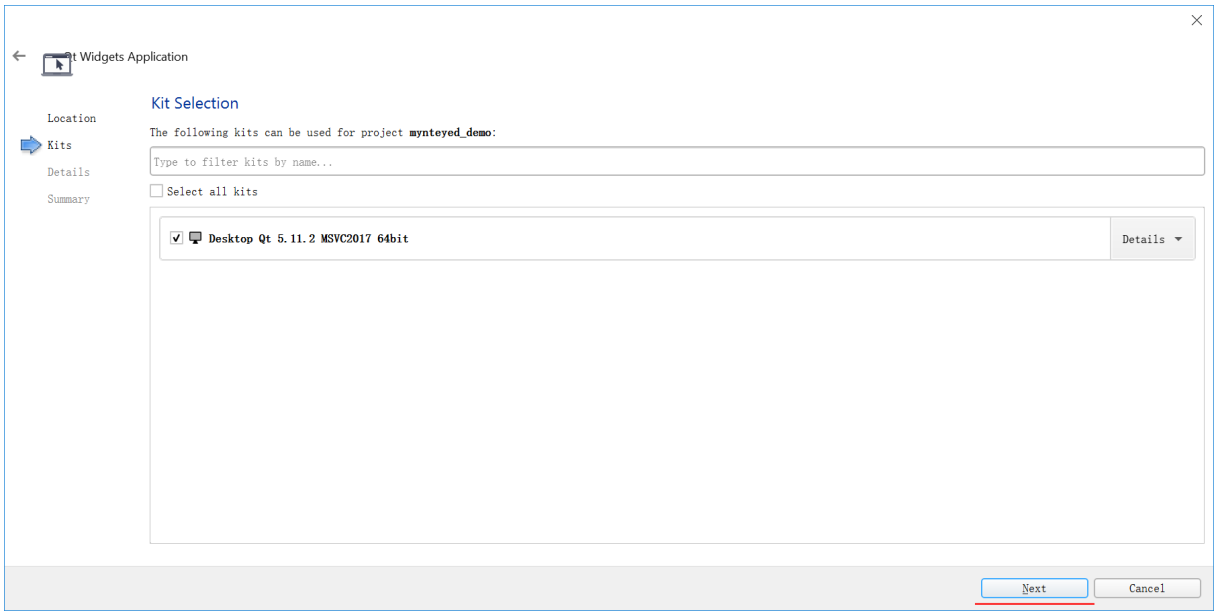
选择 Qt Widgets Application，



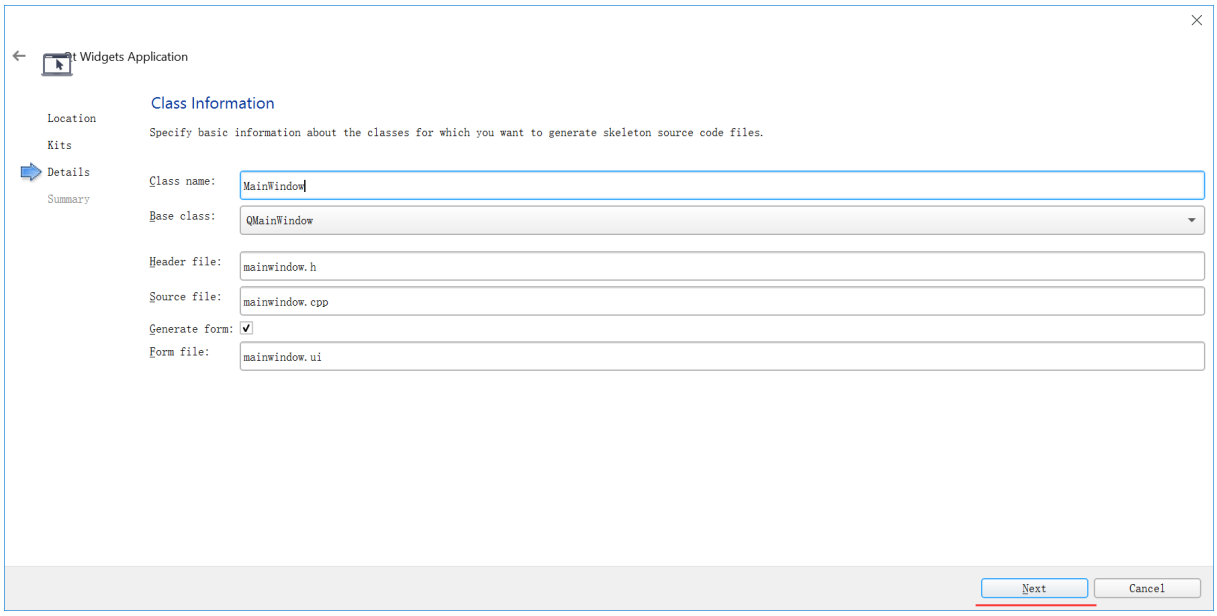
设置项目位置和名字，

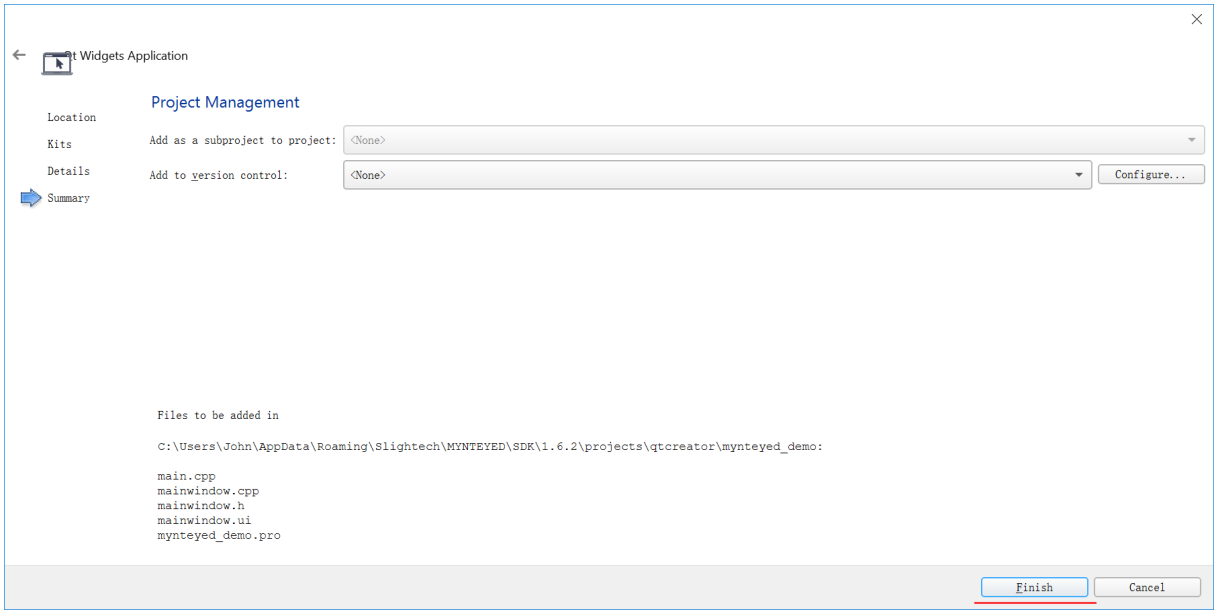


选择 build kits，

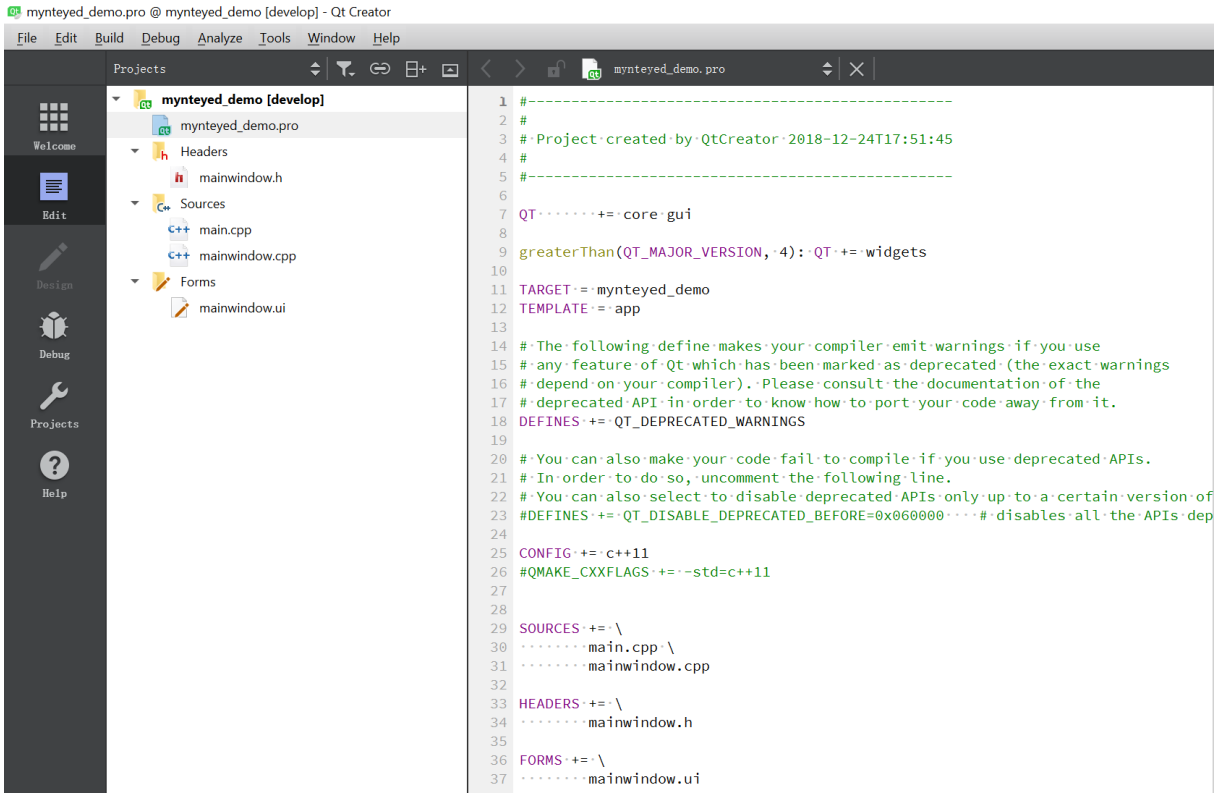


然后他将会生成框架源文件，





最后，你将会看到这样的新项目工程，



配置项目

添加 INCLUDEPATH 和 LIBS 到 mynteyed_demo.pro 。

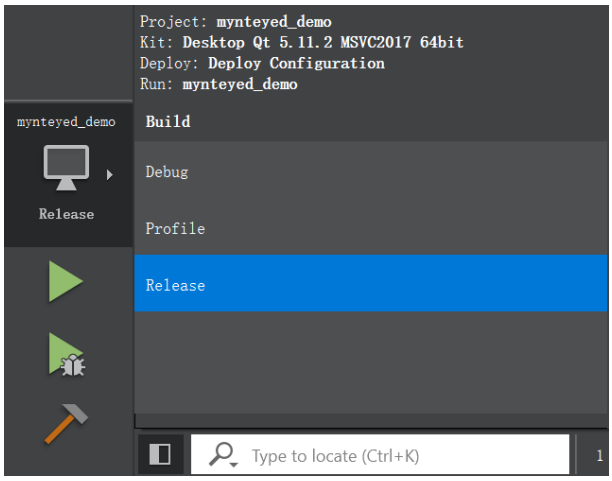
```
win32 {
    SDK_ROOT = "$$(MYNTEYED_SDK_ROOT)"
    isEmpty(SDK_ROOT) {
        error( "MYNTEYED_SDK_ROOT not found, please install SDK firstly" )
    }
    message("SDK_ROOT: $$SDK_ROOT")

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

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

使用SDK

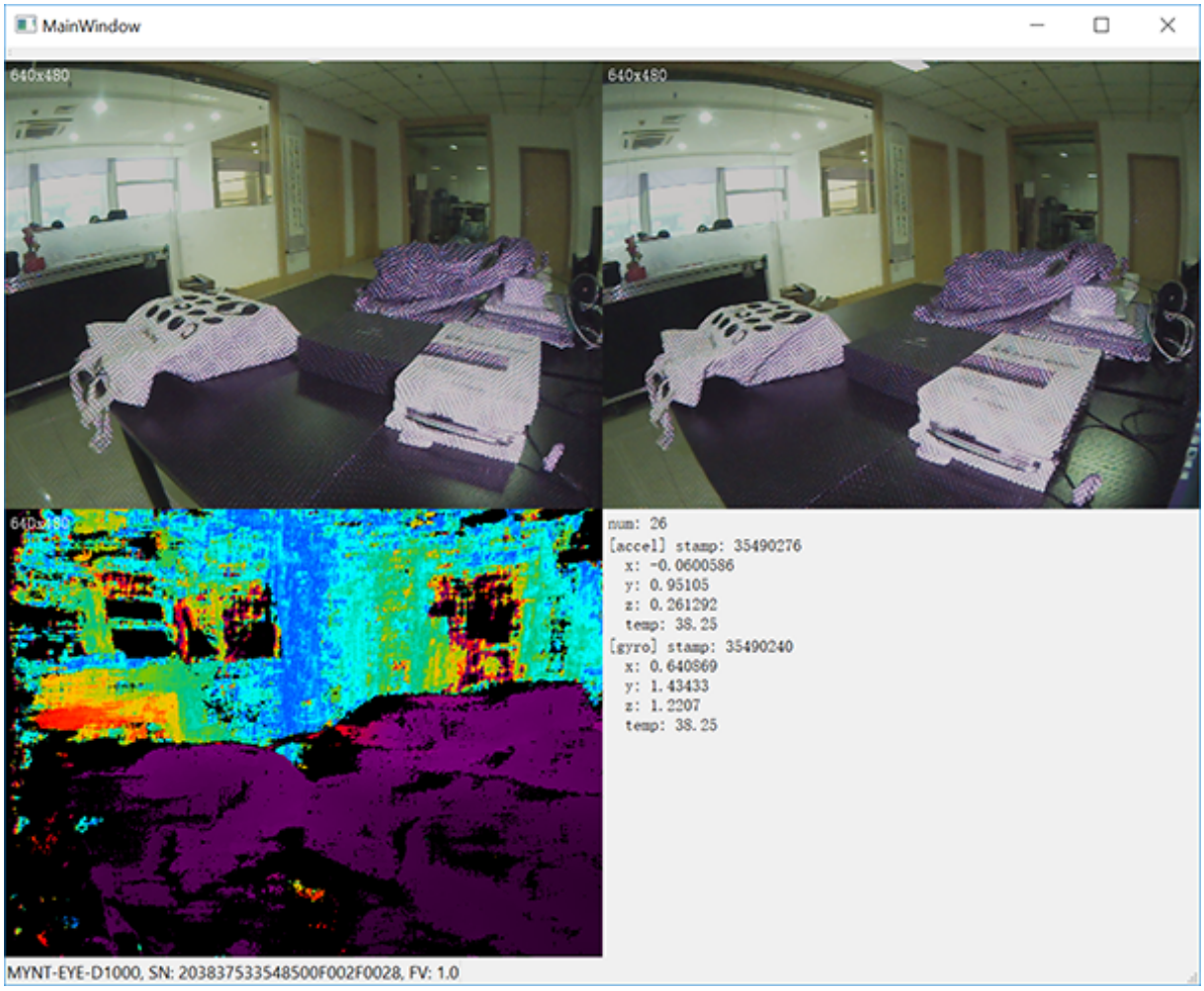
可以参考工程样例添加头文件和使用 API 。

Windows

选择 "Release" 来运行项目。

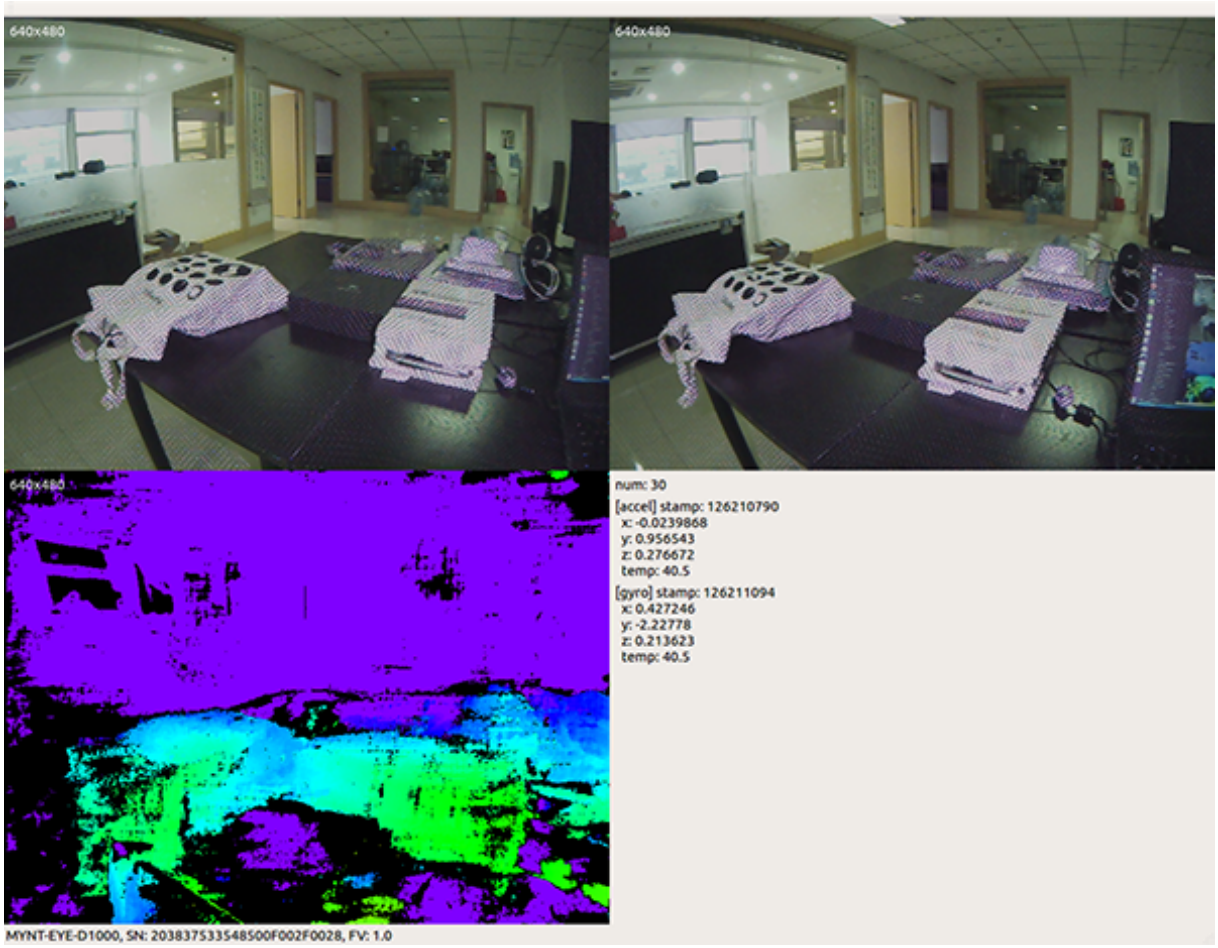


然后你将看到主窗口，



Linux

运行项目，你将看到主窗口，



6.3 CMake 如何使用 SDK

本教程将使用 CMake 创建一个项目来使用 SDK 。

> 你可以在 `<sdk>/platforms/projects/cmake` 目录下找到工程样例。

准备

- Windows: 安装 SDK 的 exe 包
- Linux: 使用源代码编译和 `make install`

创建项目

添加 `CMakeLists.txt` 和 `mynteyed_demo.cc` 文件，

```
1 cmake_minimum_required(VERSION 3.0)
2
3 project(mynteyed_demo VERSION 1.0.0 LANGUAGES C CXX)
4
5 # flags
6
7 set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wall -O3")
8 set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -O3")
9
10 set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -std=c++11 -march=native")
11 set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11 -march=native")
12
13 ## mynteyed_demo
14
15 add_executable(mynteyed_demo mynteyed_demo.cc)
```

配置项目

增加 mynteyed 和 OpenCV 到 CMakeLists.txt ，

```
1 # packages
2
3 if(MSVC)
4   set(SDK_ROOT "$ENV{MYNTEYED_SDK_ROOT}")
5   if(SDK_ROOT)
6     message(STATUS "MYNTEYED_SDK_ROOT: ${SDK_ROOT}")
7     list(APPEND CMAKE_PREFIX_PATH
8       "${SDK_ROOT}/lib/cmake"
9       "${SDK_ROOT}/3rdparty/opencv/build"
10     )
11   else()
12     message(FATAL_ERROR "MYNTEYED_SDK_ROOT not found, please install SDK firstly")
13   endif()
14 endif()
15
16 ## mynteyed
17
18 find_package(mynteyed REQUIRED)
19 message(STATUS "Found mynteye: ${mynteyed_VERSION}")
20
21 # When SDK build with OpenCV, we can add WITH_OPENCV macro to enable some
22 # features depending on OpenCV, such as ToMat().
23 if(mynteyed_WITH_OPENCV)
24   add_definitions(-DWITH_OPENCV)
25 endif()
26
27 ## OpenCV
28
29 # Set where to find OpenCV
30 #set(OpenCV_DIR "/usr/share/OpenCV")
31
32 # When SDK build with OpenCV, we must find the same version here.
33 find_package(OpenCV REQUIRED)
34 message(STATUS "Found OpenCV: ${OpenCV_VERSION}")
```

将 include_directories 和 target_link_libraries 添加到 mynteyed_demo 目标，

```
1 # targets
2
3 include_directories(
4   ${OpenCV_INCLUDE_DIRS}
5 )
6
7 ## mynteyed_demo
8
9 add_executable(mynteyed_demo mynteyed_demo.cc)
10 target_link_libraries(mynteyed_demo mynteye_depth ${OpenCV_LIBS})
```

使用SDK

可以参考工程样例添加头文件和使用 API 。

Windows

可以参考 Quick Start Guide for Windows 安装编译工具。

然后打开 "x64 Native Tools Command Prompt for VS 2017" 命令行来编译和运行，

```
1 mkdir _build
2 cd _build
3
4 cmake -G "Visual Studio 15 2017 Win64" ..
5
6 msbuild.exe ALL_BUILD.vcxproj /property:Configuration=Release
7
8 .\Release\mynteyed_demo.exe
```

Linux

打开命令行来编译和运行，

```
1 mkdir _build
2 cd _build/
3
4 cmake ..
5
6 make
7
8 ./mynteyed_demo
```

Chapter 7

继承关系索引

7.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

Chapter 8

类索引

8.1 类列表

这里列出了所有类、结构、联合以及接口定义等，并附带简要说明:

Chapter 9

类说明

9.1 mynteyed::Camera类 参考

Public 成员函数

- std::vector< DeviceInfo > GetDeviceInfos () const

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

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

  Get all stream infos.
- ErrorCode Open ()

  Open camera.
- ErrorCode Open (const OpenParams &params)

  Open camera with params.
- bool IsOpened () const

  Whethor camera is opened or not.
- std::shared_ptr< device::Descriptors > GetDescriptors () const

  Get all device descriptors.
- std::string GetDescriptor (const Descriptor &desc) const

  Get one device descriptor.
- StreamIntrinsics GetStreamIntrinsics (const StreamMode &stream_mode) const

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

  Get the intrinsics of camera.
- StreamExtrinsics GetStreamExtrinsics (const StreamMode &stream_mode) const

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

  Get the extrinsics of camera.
- bool WriteCameraCalibrationBinFile (const std::string &filename)

  Write camera calibration bin file.
- MotionIntrinsics GetMotionIntrinsics () const

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

  Get the intrinsics of motion.

- MotionExtrinsics GetMotionExtrinsics () const

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

  Get the extrinsics from left to motion.
- bool IsWriteDeviceSupported () const

  Whethor write device supported or not.
- bool WriteDeviceFlash (device::Descriptors *desc, device::ImuParams *imu_params, Version *spec_version=nullptr)

  Write device flash.
- void EnableProcessMode (const ProcessMode &mode)

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

  Enable process mode, e.g.
- bool IsImageInfoSupported () const

  Whethor image info supported or not.
- void EnableImageInfo (bool sync)

  Enable image infos.
- void DisableImageInfo ()

  Disable image info.
- bool IsImageInfoEnabled () const

  Whethor image info enabled or not.
- bool IsImageInfoSynced () const

  Whethor image info synced or not.
- bool IsStreamDataEnabled (const ImageType &type) const

  Whethor stream data of certain image type enabled or not.
- bool HasStreamDataEnabled () const

  Has any stream data enabled.
- StreamData GetStreamData (const ImageType &type)

  Get latest stream data.
- std::vector< StreamData > GetStreamDatas (const ImageType &type)

  Get cached stream datas.
- bool IsMotionDatasSupported () const

  Whethor motion datas supported or not.
- void EnableMotionDatas (std::size_t max_size=std::numeric_limits< std::size_t >::max())

  Enable motion datas.
- void DisableMotionDatas ()

  Disable motion datas.
- bool IsMotionDatasEnabled () const

  Whethor motion datas enabled or not.
- std::vector< MotionData > GetMotionDatas ()

  Get cached motion datas.
- void SetImgInfoCallback (img_info_callback_t callback, bool async=true)

  Set image info callback.
- void SetStreamCallback (const ImageType &type, stream_callback_t callback, bool async=true)

  Set stream data callback.
- void SetMotionCallback (motion_callback_t callback, bool async=true)

  Set motion data callback.
- void Close ()

  Close the camera.
- bool HidFirmwareUpdate (const char *filepath)

  Update hid device firmware.

- void SetExposureTime (const float &value)

  Set exposure time [1ms - 2000ms] value – exposure time value.
- void GetExposureTime (float &value)

  Get exposure time value – return exposure time value.
- void SetGlobalGain (const float &value)

  Set global gain [1 - 16] value – global gain value.
- void GetGlobalGain (float &value)

  Get global gain value – return global gain value.
- void SetIRIntensity (const std::uint16_t &value)

  set infrared(IR) intensity [0, 10] default 4
- bool AutoExposureControl (bool enable)

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

  Auto-white-balance enabled or not default enabled.

9.1.1 成员函数说明

9.1.1.1 void mynteyed::Camera::DisableImageInfo ( )

Disable image info.

9.1.1.2 void mynteyed::Camera::DisableMotionDatas ( )

Disable motion datas.

9.1.1.3 void mynteyed::Camera::EnableImageInfo (bool *sync*)

Enable image infos.

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

9.1.1.4 void mynteyed::Camera::EnableMotionDatas (std::size_t *max_size* = `std::numeric_limits< std::size_t >::max()`)

Enable motion datas.

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

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

9.1.1.5 void mynteyed::Camera::EnableProcessMode (const ProcessMode & *mode*)

Enable process mode, e.g.

imu assembly, temp_drift

9.1.1.6 void mynteyed::Camera::EnableProcessMode (const std::int32_t & *mode*)

Enable process mode, e.g.

imu assembly, temp_drift

9.1.1.7 std::vector<MotionData> mynteyed::Camera::GetMotionDatas ( )

Get cached motion datas.

Besides, you can also get them from callback

9.1.1.8 void mynteyed::Camera::SetImgInfoCallback (img_info_callback_t *callback*, bool *async* = `true`)

Set image info callback.

9.1.1.9 void mynteyed::Camera::SetMotionCallback (motion_callback_t *callback*, bool *async* = `true`)

Set motion data callback.

9.1.1.10 void mynteyed::Camera::SetStreamCallback (const ImageType & *type*, stream_callback_t *callback*, bool *async* = `true`)

Set stream data callback.

9.2 mynteyed::CameraIntrinsics结构体 参考

Camera intrinsics: size, coeffs and camera matrix.

Public 属性

- std::uint16_t width
  The width of the image in pixels.
- std::uint16_t height
  The height of the image in pixels.
- double fx
  The focal length of the image plane, as a multiple of pixel width.
- double fy
  The focal length of the image plane, as a multiple of pixel height.
- double cx
  The horizontal coordinate of the principal point of the image.
- double cy
  The vertical coordinate of the principal point of the image.
- double coeffs [5]
  The distortion coefficients: k1,k2,p1,p2,k3.
- double p [12]
  3x4 projection matrix in the (rectified) coordinate systems left: fx' cx' fy' cy' 1 right: fx' cx' tx fy' cy' 1

9.2.1 详细描述

Camera intrinsics: size, coeffs and camera matrix.

9.3 mynteyed::device::Descriptors结构体 参考

Device descriptors.

9.3.1 详细描述

Device descriptors.

9.4 mynteyed::DeviceInfo结构体 参考

Device information.

Public 属性

- std::int32_t index
  The device index.
- std::string name
  The device name.
- std::uint16_t type
  The device type.
- std::uint16_t pid
  The product id.
- std::uint16_t vid
  The vendor id.
- std::uint16_t chip_id
  The chip id.
- std::string fw_version
  The firmware version.

9.4.1 详细描述

Device information.

9.5 mynteyed::Extrinsics结构体 参考

Extrinsics, represent how the different datas are connected.

Public 成员函数

- Extrinsics Inverse () const

  Inverse this extrinsics.

Public 属性

- double rotation [3][3]

  Rotation matrix left camera to right camera.
- double translation [3]

  Translation vector left camera to right camera.

9.5.1 详细描述

Extrinsics, represent how the different datas are connected.

9.5.2 成员函数说明

9.5.2.1 Extrinsics mynteyed::Extrinsics::Inverse ( ) const [inline]

Inverse this extrinsics.

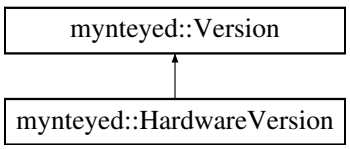
返回

the inversed extrinsics.

9.6 mynteyed::HardwareVersion类 参考

Hardware version.

类 mynteyed::HardwareVersion 继承关系图:

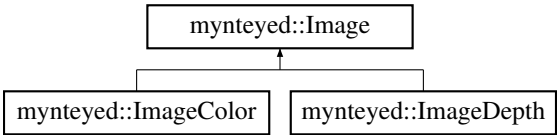


9.6.1 详细描述

Hardware version.

9.7 mynteyed::Image类 参考

类 mynteyed::Image 继承关系图:



9.8 mynteyed::ImageColor类 参考

类 mynteyed::ImageColor 继承关系图:

9.9 mynteyed::ImageDepth类 参考

类 mynteyed::ImageDepth 继承关系图:

9.10 mynteyed::ImgInfo结构体 参考

Image information.

Public 属性

- std::uint16_t frame_id

  Image frame id.
- std::uint32_t timestamp

  Image timestamp.
- std::uint16_t exposure_time

  Image exposure time.

9.10.1 详细描述

Image information.

9.11 mynteyed::ImuData结构体 参考

Imu data.

Public 属性

- std::uint8_t flag

  Data type MYNTEYE_IMU_ACCEL: accelerometer MYNTEYE_IMU_GYRO: gyroscope.
- std::uint64_t timestamp

  Imu gyroscope or accelerometer or frame timestamp.
- double temperature

  temperature
- double accel [3]

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

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

9.11.1 详细描述

Imu data.

9.11.2 类成员变量说明

9.11.2.1 double mynteyed::ImuData::accel[3]

Imu accelerometer data for 3-axis: X, Y, X.

9.11.2.2 double mynteyed::ImuData::gyro[3]

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

9.12 mynteyed::ImuIntrinsics结构体 参考

IMU intrinsics: scale, drift and variances.

Public 属性

- double scale [3][3]

  Scale matrix.
- double assembly [3][3]

  Assembly error [3][3].
- double noise [3]

  Noise density variances.
- double bias [3]

  Random walk variances.
- double x [2]

  Temperature drift.

9.12.1 详细描述

IMU intrinsics: scale, drift and variances.

9.12.2 类成员变量说明

9.12.2.1 double mynteyed::ImuIntrinsics::scale[3][3]

Scale matrix.

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

9.12.2.2 double mynteyed::ImuIntrinsics::x[2]

Temperature drift.

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

9.13 mynteyed::device::ImuParams结构体 参考

Device imu paramters.

9.13.1 详细描述

Device imu paramters.

9.14 mynteyed::MotionData结构体 参考

Motion data.

Public 属性

- std::shared_ptr< ImuData > imu
  ImuData.

9.14.1 详细描述

Motion data.

9.14.2 类成员变量说明

9.14.2.1 std::shared_ptr<ImuData> mynteyed::MotionData::imu

ImuData.

9.15 mynteyed::MotionIntrinsics结构体 参考

Motion intrinsics, including accelerometer and gyroscope.

Public 属性

- ImuIntrinsics accel
  Accelerometer intrinsics.
- ImuIntrinsics gyro
  Gyroscope intrinsics.

9.15.1 详细描述

Motion intrinsics, including accelerometer and gyroscope.

9.16 mynteyed::OpenParams结构体 参考

Device open parameters.

Public 成员函数

- OpenParams ()

  Constructor.
- ~OpenParams ()

  Destructor.

Public 属性

- std::int32_t dev_index

  Device index.
- std::int32_t framerate

  Framerate, range [0,60], [0,30](STREAM_2560x720), default 10.
- DeviceMode dev_mode

  Device mode, default DEVICE_ALL.
- ColorMode color_mode

  Color mode, default COLOR_RAW.
- DepthMode depth_mode

  Depth mode, default DEPTH_COLORFUL.
- StreamMode stream_mode

  Stream mode of color & depth, default STREAM_1280x720.
- StreamFormat color_stream_format

  Stream format of color, default STREAM_YUYV.
- StreamFormat depth_stream_format

  Stream format of depth, default STREAM_YUYV.
- bool state_ae

  Auto-exposure, default true.
- bool state_awb

  Auto-white balance, default true.
- std::uint8_t ir_intensity

  IR (Infrared), range [0,10], default 0.
- bool ir_depth_only

  IR Depth Only mode, default false.
- float colour_depth_value

  Colour depth image, default 5000.

9.16.1 详细描述

Device open parameters.

9.16.2 构造及析构函数说明

9.16.2.1 mynteyed::OpenParams::OpenParams ( )

Constructor.

9.16.2.2 mynteyed::OpenParams::~OpenParams ( )

Destructor.

9.16.3 类成员变量说明

9.16.3.1 float mynteyed::OpenParams::colour_depth_value

Colour depth image, default 5000.

[0, 16384]

9.16.3.2 DeviceMode mynteyed::OpenParams::dev_mode

Device mode, default DEVICE_ALL.

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

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

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

9.16.3.3 bool mynteyed::OpenParams::ir_depth_only

IR Depth Only mode, default false.

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

9.17 mynteyed::Rate类 参考

9.18 mynteyed::StreamData结构体 参考

Stream data.

Public 属性

- std::shared_ptr< Image > img

  Image data.
- std::shared_ptr< ImgInfo > img_info

  Image information.

9.18.1 详细描述

Stream data.

9.19 mynteyed::StreamInfo结构体 参考

Stream information.

Public 属性

- std::int32_t index
  The stream index.
- std::int32_t width
  The stream width.
- std::int32_t height
  The stream height.
- StreamFormat format
  The stream format.

9.19.1 详细描述

Stream information.

9.20 mynteyed::StreamIntrinsics结构体 参考

Camera intrinsics: size, coeffs and camera matrix.

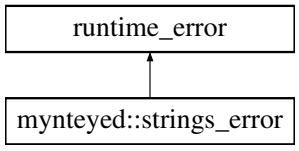
9.20.1 详细描述

Camera intrinsics: size, coeffs and camera matrix.

9.21 mynteyed::strings_error类 参考

The strings error.

类 mynteyed::strings_error 继承关系图:



9.21.1 详细描述

The strings error.

9.22 mynteyed::Type类 参考

Type.

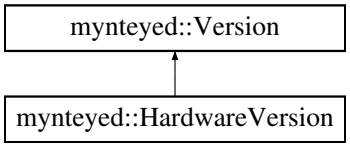
9.22.1 详细描述

Type.

9.23 mynteyed::Version类 参考

Version.

类 mynteyed::Version 继承关系图:



9.23.1 详细描述

Version.

Index