00A0 2203∃ 2200∀ 2286⊆ 2713x 27FA⟺ 221A$\surd$ 221B$\sqrt[3]{}$ 2295⊕ 2297⊗ UTF8gbsn

IMSEE SDK Documentation

发布 *1.4.2*

INDEMIND

2022 年 02 月 16 日

Contents

CHAPTER 1

产品

1.1 简介

INDEMIND 成立于 2017 年, 专注于计算机视觉技术与嵌入式计算平台研发, 公司核心技术团队成员均来自计算机视觉领域的顶级技术人员。自创立至今,INDEMIND 已成功发售 INDEMIND 双目视觉惯性模组, 并基于该模组打造了扫地机器人解决方案、商用服务机器人解决方案、穿戴计算解决方案, 可为多行业提供计算机视觉技术服务支持。

INDEMIND 双目视觉惯性模组运用摄像头 +IMU 多传感器融合架构, 使摄像头与 IMU 传感器优势互补, 实现位姿精度更高、环境适应性更强、动态性能更稳定、成本更低的 SLAM 研究硬件方案, 并设计有高精度时间同步机制, 实现时间同步精度微妙级, 进一步保障图像及 IMU 数据精度, 可为视觉 SLAM 研究、智能机器人、AR/VR/MR、无人机避障、室内外导航定位等产品研发或技术研究提供高精度、低成本的数据采集支持。

结合 INDEMIND 自研的 Vi-SLAM 算法, 实现定位稳定性 1-2mm (RMS)、绝对定位精度 < 1%、姿态稳定性 0.1 度 (RMS)、绝对姿态精度小于 1° 等一系列超行业主流水平的定位效果, 可以满足视觉 SLAM 研究、智能机器人、无人机避障、室内外导航定位等使用需求, 有效的节约算法开发周期及成本, 让开发者可以迅速调试及部署。

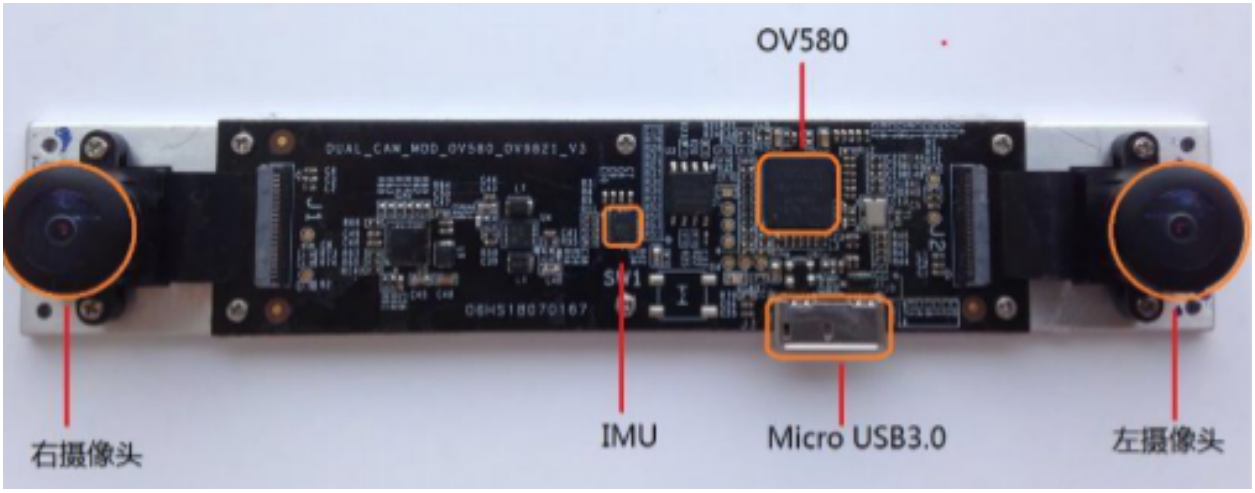
INDEMIND 双目视觉惯性模组内置最高频率 1000Hz 的 6 轴 IMU 传感器及两枚全局快门的 1280*800 高清摄像头, 可提供 1280x800@50fps 、1280x800@100fps、640x400@100fps 、640x400@200fps 的图像采集能力, 结合水平 120°、垂向 75° 的视场角, 可在高速机动机下精准获取图像及 IMU 数据, 足开发者的数据采集需求, 极大降低相应算法追踪难度, 并可满足高速 SLAM 算法 (如车载) 的定位和建图数据需求, 为算法开发工作提供强有力前端数据采集能力。

1.2 尺寸结构

尺寸结构如表所示：

总体尺寸（mm）	板子尺寸（mm）	基线长度（mm）
140×25	95×25	120

模组实物图（拆去壳体后）：



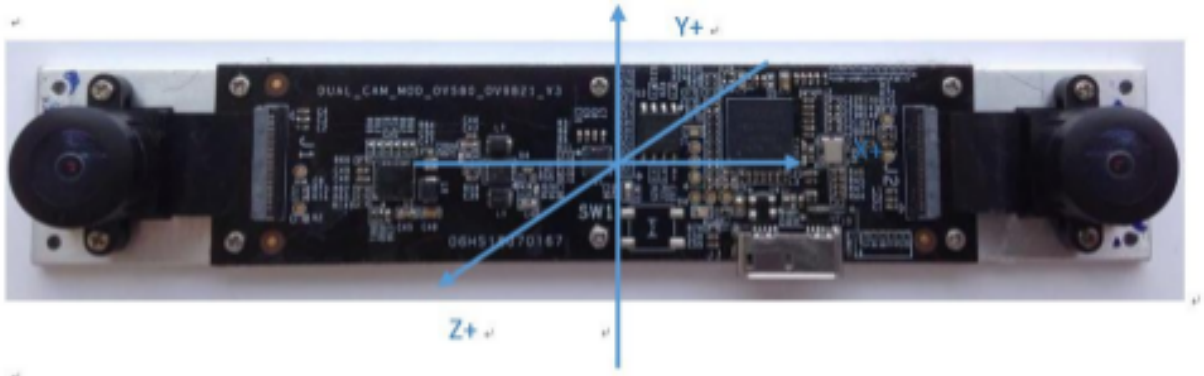
左、右摄像头: 左、右摄像头为传感器镜头, 使用中请注意保护, 以避免成像质量下降。

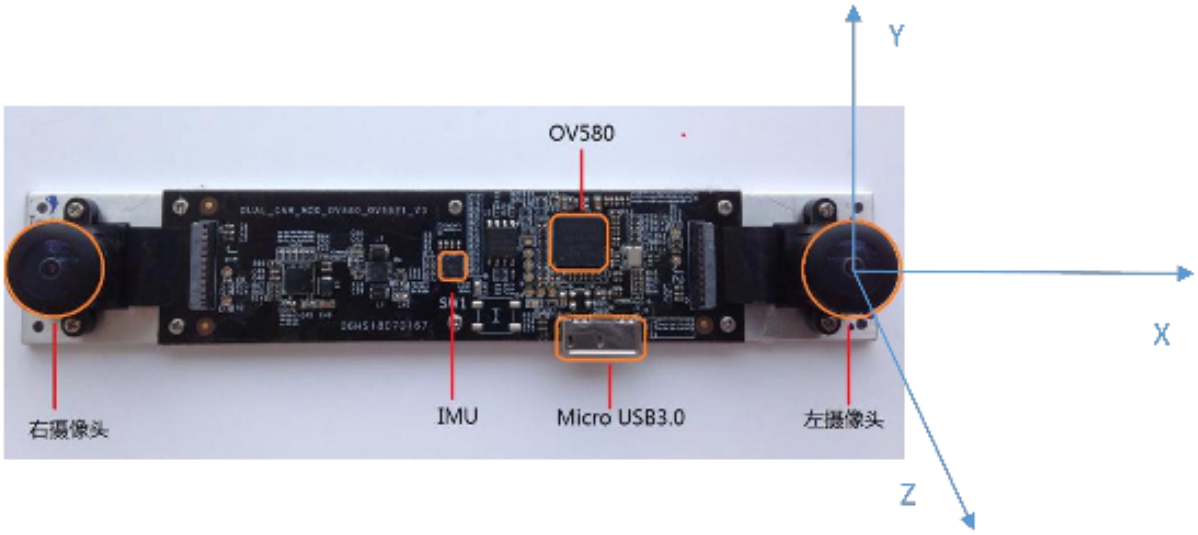
IMU: 高频率 IMU 硬件, 结合双目摄像头, 实现精准定位, 相机与 IMU 安装精密, 切勿拆卸。

OV580: 图像、IMU 数据获取芯片。

Micro USB3.0: 使用中, 插上 Micro USB3.0 数据线后, 保持连接处自然受力, 不弯折, 以避免使用中损坏接口, 或导致数据连接不稳定。

IMU Coordinate System(IMU 坐标系统):IMU 坐标系统为右手系, 坐标轴方向如图所示:





世界坐标系: 世界坐标系为当地地理水平坐标系,X、Z 轴沿水平,Y 轴朝天向, 方位。0 度为模组初始化 Z 轴指向, 坐标系原点为初始化时模值左目镜头位置。

输出数据:X、Y、Z 位置, 姿态四元数, 其中位置为 IMU 在世界坐标系下的位置 (IMU 位于双目模组中间位置), 其中姿态为 IMU 在世界坐标系下的姿态。

1.3 产品数据

1.3.1 产品规格

型号	IMSEE 双目惯性模组
PCB 尺寸	95mm*25mm
实物尺寸	140mm*25mm*30mm
帧率与帧率	1280×800@50FPS、1280×800@100FPS 640×400@100FPS、640×400@200FPS
深度分辨率	Based on CPU/GPU Up to 640*400@25FPS
像素尺寸	3.0 x 3.0μm(1280*800)
快门速度	5 ms
基线	120.0mm
镜头	Replacable Standard M12
视角	D:140° H:120° V:75°
焦距	2.09mm
运动感知	6 Axis IMU
色彩模式	Monochrome
扫描模式	Global Shutter
功耗	1~2.7W@5V DC from USB
同步精度	<1ms (up to 0.05ms)
IMU 频率	1000Hz
输出数据格式	Raw data
接口	USB3.0
重量	52g
UVC MODE	Yes

1.3.2 传感器选型

器件	型号	参数
摄像头	OV9281	50FPS –1280×800
IMU	ICM20602	1000Hz

1.3.3 软件

支持操作系统	Windows 10、Ubuntu 16.04/18.04、ROS kinetic
SDK 地址	https://github.com/INDEMIND/IMSEE-SDK
开发者支持	SDK

CHAPTER 2

SDK

2.1 平台支持

SDK 是基于 CMake 构建的，用以跨 Linux, Windows 平台。

已测试可用的平台有：

- Windows 10
- Ubuntu 18.04.1 / 16.04.6
- Jetson TX2 / Xavier
- firefly RK3399 / RK3328

警告： 为了不影响您的正常使用与开发，请使用 USB 3.0 接口。

2.2 SDK 安装

2.2.1 Ubuntu 安装

Ubuntu 16.04	Ubuntu 18.04
✓	✓

小技巧： 预安装项如下：

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

获取代码

```
sudo apt-get install git
git clone https://github.com/indemind/IMSEE-SDK.git
```

准备依赖

```
cd <IMSEE-SDK>  # <IMSEE-SDK> 为SDK具体路径
make init
```

- OpenCV

小技巧:

- 如果需要安装 ROS，可以不用安装 OpenCV/PCL, 以防兼容性问题。SDK 默认基于 OpenCV3.3.1(ROS Kinetic 自带 OpenCV 版本) 编译。同时我们也提供基于 OpenCV3.4.3 编译的库，在 <IMSEE-SDK>/lib/others 文件夹中。如果客户想基于该版本编译，请在相应文件夹下替换。详细可咨询技术支持。
- OpenCV 如何编译安装，请见官方文档 Installation in Linux 。或参考如下命令:

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

$ git clone https://github.com/opencv/opencv.git
$ cd opencv/
$ git checkout tags/3.4.3

$ 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 \
..
```

(下页继续)

(续上页)

```
$ make -j4
$ sudo make install
```

编译代码

小技巧： 如果 OpenCV 安装到了自定义目录或想指定某一版本，编译前可如下设置路径, 或者直接把该变量写到 ~/.bashrc：

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

不然，CMake 会提示找不到 OpenCV 。

- MNN

小技巧： MNN 依赖 protobuf（使用 3.0 或以上版本），如未安装过 protobuf，请参考如下命令：

```
[required] sudo apt-get install autoconf automake libtool

$ git clone https://github.com/google/protobuf.git
$ cd protobuf
$ git submodule update --init --recursive
$ ./autogen.sh
$ ./configure
$ make
$ make check
$ sudo make install
$ sudo ldconfig # refresh shared library cache.
$ protoc --version # 若安装成功，将显示protoc版本
```

至此，可以开始编译安装 MNN 了：

```
$ git clone https://github.com/alibaba/MNN.git
$ cd MNN
$ ./schema/generate.sh
$ mkdir build $$ cd build
$ cmake ..
$ make -j4
$ sudo make install
```

安装好 OpenCV 与 MNN 后，可以开始编译 SDK 中的 demo 了

编译样例

```
```

$ cd <IMSEE-SDK> # <IMSEE-SDK> 为 SDK 具体路径　　$ make demo

运行样例：

```
$ ./demo/output/bin/get_image
```

教程样例，请阅读*SDK 数据样例*。

结语

工程要引入 SDK 的话，CMake 可参考 `demo/CMakeLists.txt` 里的配置。不然，就是直接引入安装目录里的头文件和动态库。

2.2.2 Windows 源码安装

Windows 10
✓

软件准备

- CMake(3.0 以上)(需要支持 vs2019)
- Visual Studio 2019
- opencv3.3.1
- IMSEE-SDK

配置准备

- 将 CMake 安装路径下 bin 目录路径添加到系统环境变量”PATH” 中。例如：”C:\Program Files\CMake\bin”
- 设置”.sln” 文件的默认打开方式为”Microsoft Visual Studio 2019”。
- 右键”.sln” 文件，选择” 打开方式” -> ” 选择其他应用”，弹出打开应用小窗口。
- 选中”Visual Studio 2019”，勾选” 始终使用此应用打开.sln 文件”，点击” 确定” 按钮。

编译 OpenCV

- 双击”opencv-3.3.1-vc14.exe”，解压文件到指定目录下。例如：解压目录为：”D:\opencv331”。
- 启动 CMake，在”Where is the source code” 中输入 opencv 源码路径。例如：源码路径为：”D:\opencv331\opencv\sources”。
- 在”Where to build the binaries” 中输入”opencv 构建目录”(即，”sources” 所在路径)。例如：构建目录为：”D:\opencv331\opencv\build-win10-x64-vs19”。
- 点击”Configure” 按钮，选择编译工具为”Visual Studio 16 2019”，选择编译目标平台为”x64”。点击”Finish” 按钮，进行第一次配置。

- 第一次配置”Configuring done” 后，在配置窗口中，选中”BUILD_opencv_world”，取消选中”BUILD_DOCS”、”BUILD_EXAMPLES”、”BUILD_TESTS”(取消选中可减少 opencv 编译时间)。再点击”Configure” 按钮，进行第二次编译配置。第二次配置”Configuring done” 后，点击”Generate” 按钮，进行 win 项目生成..。”Generating done” 后，点击”Open Project” 按钮 (或用 vs2019 打开构建目录下的 OpenCV.sln)。
- 在 vs2019 窗口，选择编译目标平台为”Release”，点击” 生成”->” 生成解决方案”，开始编译。编译成功后，生成的文件在”opencv 构建目录\bin\Release” 下。
- 将”opencv 构建目录 binRelease” 添加到环境变量”PATH”。例如：”D:\opencv331\opencv\build-win10-x64-vs19\bin\Release”。
- 新建系统变量”OpenCV_DIR”，值为”opencv 构建目录”。例如：”D:\opencv331\opencv\build-win10-x64-vs19”。

编译 Demo

- 双击文件”IMSEE-SDK\demo\build-demo.bat”，会自动打开”cmake-gui.exe”。注意：不要手动关闭”build-demo.bat”，它会自动关闭。
- 在 CMake 窗口，点击”Configure” 按钮，选择编译工具为”Visual Studio 16 2019”，选择编译目标平台为”x64”。点击”Finish” 按钮，此时 CMake 进行编译配置。
- ”Configuring done” 后，点击”Generate” 按钮，进行 win 项目生成。
- ”Generating done” 后，关闭 CMake 窗口。脚本自动用 vs2019 打开”build/indemind_demos.sln”。注意：vs2019 打开后，”build-demo.bat” 会自动关闭
- 在 vs2019 中选择”Release” 版本，点击” 生成”->” 生成解决方案”，开始编译 demo。生成的文件在”IMSEE-SDK\demo\output\bin” 下。

运行 Demo

- 转到目录”IMSEE-SDK\demo\output\bin”，运行目录下的各个 demo。

小技巧:

- 编译目标平台要指定为”x64”。
- 使用”Visual Studio 2015” 或其他编译工具时，与上面步骤相同。

2.2.3 ROS Wrapper 安装

ROS Kinetic

✓

环境准备

- ROS

ROS Kinetic (Ubuntu 16.04)

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

编译代码

```
$ cd <IMSEE-SDK>  # <IMSEE-SDK> 为SDK具体路径
$ make ros
```

运行节点

```
$ sudo su #开启权限模式
$ source ros/devel/setup.bash
$ roslaunch imsee_ros_wrapper start.launch
```

如果想通过 RViz 预览：

```
$ sudo su #开启权限模式
$ source ros/devel/setup.bash
$ roslaunch imsee_ros_wrapper display.launch
```

2.3 SDK 数据样例

2.3.1 获取设备信息

通过 SDK 的 “GetModuleParams“及 “GetModuleInfo“分别可获取模组的参数信息及硬件信息。

参考代码片段：

```
auto m_pSDK = new CIMRSDK();
MRCONFIG config = {0};
config.bSlam = false;
config.imgResolution = IMG_640;
config.imgFrequency = 50;
config.imuFrequency = 1000;

// m_pSDK->Init(config);
indem::MoudleAllParam param = m_pSDK->GetModuleParams();
indem::ModuleInfo info = m_pSDK->GetModuleInfo();

std::cout << "Module info: " << std::endl;
info.printInfo();
std::cout << "Left param: " << std::endl;
```

(下页继续)

(续上页)

```
param._left_camera[RESOLUTION::RES_640X400].printInfo();
std::cout << "Right param: " << std::endl;
param._right_camera[RESOLUTION::RES_640X400].printInfo();
std::cout << "Imu param: " << std::endl;
param._imu.printInfo();
return 0;
```

2.3.2 获取原始图像

获取图像前，需要初始化 SDK, 然后直接通过回调获取。

参考代码片段：

```
auto m_pSDK = new CIMRSDK();
MRCONFIG config = {0};
config.bSlam = false;
config.imgResolution = IMG_640;
config.imgFrequency = 50;
config.imuFrequency = 1000;

m_pSDK->Init(config);

std::queue<cv::Mat> image_queue;
int img_count = 0;
double last_img_time = -1.0;
m_pSDK->RegistImgCallback([&last_img_time, &img_count, &image_queue](
                              double time, cv::Mat left, cv::Mat right) {
  if (!left.empty() && !right.empty()) {
    cv::Mat img;
    cv::hconcat(left, right, img);
    if (last_img_time >= 0) {
      std::ostringstream ss;
      double fps = 1.0 / (time - last_img_time);
      ss << std::fixed << "time stamp: " << std::setprecision(2) << time
         << ", fps: " << fps;
      cv::putText(img, ss.str(), cv::Point(25, 25), FONT_FACE, FONT_SCALE,
                  FONT_COLOR, THICKNESS);
    }
    last_img_time = time;
    image_queue.push(img);
    ++img_count;
  }
});
auto &&time_beg = times::now();
while (true) {
  if (!image_queue.empty()) {
    cv::imshow("image", image_queue.front());
    image_queue.pop();
  }
  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') { // ESC/Q
    break;
  }
}
delete m_pSDK;
```

(下页继续)

(续上页)

```
return 0;
```

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_image.cpp 。

2.3.3 获取 imu 数据

获取 imu 前，需要初始化 SDK, 然后直接通过回调获取。

参考代码片段:

```
auto m_pSDK = new CIMRSDK();
MRCONFIG config = {0};
config.bSlam = false;
config.imgResolution = IMG_640;
config.imgFrequency = 50;
config.imuFrequency = 1000;

m_pSDK->Init(config);

std::queue<cv::Mat> image_queue;

int img_count = 0;
m_pSDK->RegistImgCallback(
    [&img_count, &image_queue](double time, cv::Mat left, cv::Mat right) {
      if (!left.empty() && !right.empty()) {
        cv::Mat img;
        cv::hconcat(left, right, img);
        image_queue.push(img);
        ++img_count;
      }
    });
int imu_count = 0;
m_pSDK->RegistModuleIMUCallback([&imu_count](ImuData imu) {
  if (imu_count % 100 == 0) {
    LOG(INFO) << "imu timestamp: " << imu.timestamp
              << ", accel x: " << imu.accel[0]
              << ", accel y: " << imu.accel[1]
              << ", accel z: " << imu.accel[2] << ", gyro x: " << imu.gyro[0]
              << ", gyro y: " << imu.gyro[1] << ", gyro z: " << imu.gyro[2];
  }
  ++imu_count;
});
auto &&time_beg = times::now();
while (true) {
  if (!image_queue.empty()) {
    cv::imshow("image", image_queue.front());
    image_queue.pop();
  }
  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') { // ESC/Q
    break;
  }
}
```

(下页继续)

(续上页)

```
delete m_pSDK;
return 0;
```

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_imu.cpp 。

2.3.4 获取矫正图像

矫正图像为 SDK 处理图像，获取矫正图像需要 EnableRectifyProcessor 开启深度计算，然后通过回调获取。参考代码片段：

```
auto m_pSDK = new CIMRSDK();
MRCONFIG config = {0};
config.bSlam = false;
config.imgResolution = IMG_640;
config.imgFrequency = 50;
config.imuFrequency = 1000;

m_pSDK->Init(config);
std::queue<cv::Mat> rectified_queue;
int rectified_img_count = 0;
if (m_pSDK->EnableRectifyProcessor()) {
  m_pSDK->RegistRectifiedImgCallback(
      [&rectified_img_count, &rectified_queue](double time, cv::Mat left,
                                               cv::Mat right) {
        if (!left.empty() && !right.empty()) {
          cv::Mat img;
          cv::hconcat(left, right, img);
          rectified_queue.push(img);
          ++rectified_img_count;
        }
      });
}
auto &&time_beg = times::now();
while (true) {
  if (!rectified_queue.empty()) {
    cv::imshow("rectified_img", rectified_queue.front());
    rectified_queue.pop();
  }
  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') { // ESC/Q
    break;
  }
}
delete m_pSDK;
return 0;
```

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_rectified_img.cpp 。

2.3.5 获取视差图像

视差图像为 SDK 处理图像，获取视差图像需要 `EnableDisparityProcessor` 开启视差计算，然后通过回调获取。

参考代码片段：

```
auto m_pSDK = new CIMRSDK();
MRCONFIG config = {0};
config.bSlam = false;
config.imgResolution = IMG_640;
config.imgFrequency = 50;
config.imuFrequency = 1000;

m_pSDK->Init(config);
std::queue<cv::Mat> disparity_queue;
int disparity_count = 0;
if (m_pSDK->EnableDisparityProcessor()) {
  m_pSDK->EnableLRConsistencyCheck();
  // m_pSDK->SetDepthCalMode(DepthCalMode::HIGH_ACCURACY);
  m_pSDK->RegistDisparityCallback(
      [&disparity_count, &disparity_queue](double time, cv::Mat disparity) {
        if (!disparity.empty()) {
          disparity.convertTo(disparity, CV_8U, 255. / (16 * 64));
          cv::applyColorMap(disparity, disparity, cv::COLORMAP_JET);
          disparity.setTo(0, disparity == -16);
          disparity_queue.push(disparity);
          ++disparity_count;
        }
      });
}
auto &&time_beg = times::now();
while (true) {
  if (!disparity_queue.empty()) {
    cv::imshow("disparity", disparity_queue.front());
    disparity_queue.pop();
  }
  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') { // ESC/Q
    break;
  }
}
delete m_pSDK;
return 0;
```

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_disparity.cpp 。

2.3.6 获取左右一致性检测视差图像

视差图像为 SDK 处理图像，获取视差图像需要 `EnableDisparityProcessor` 开启视差计算，再通过“EnableLRConsistencyCheck“使能左右一致性检测（默认为不检测），然后通过回调获取。

参考代码片段：

```
auto m_pSDK = new CIMRSDK();
MRCONFIG config = {0};
config.bSlam = false;
config.imgResolution = IMG_640;
config.imgFrequency = 50;
config.imuFrequency = 1000;

m_pSDK->Init(config);
std::queue<cv::Mat> disparity_queue;
int disparity_count = 0;
if (m_pSDK->EnableDisparityProcessor()) {
  m_pSDK->EnableLRConsistencyCheck();
  // m_pSDK->SetDepthCalMode(DepthCalMode::HIGH_ACCURACY);
  m_pSDK->RegistDisparityCallback(
      [&disparity_count, &disparity_queue](double time, cv::Mat disparity) {
        if (!disparity.empty()) {
          disparity.convertTo(disparity, CV_8U, 255. / (16 * 64));
          cv::applyColorMap(disparity, disparity, cv::COLORMAP_JET);
          disparity.setTo(0, disparity == -16);
          disparity_queue.push(disparity);
          ++disparity_count;
        }
      });
}
auto &&time_beg = times::now();
while (true) {
  if (!disparity_queue.empty()) {
    cv::imshow("disparity", disparity_queue.front());
    disparity_queue.pop();
  }
  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') { // ESC/Q
    break;
  }
}
delete m_pSDK;
return 0;
```

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_disparity_with_lr_check.cpp 。

小技巧： 开启左右一致性检测会影响视差计算的速度，用户酌情使用。

2.3.7 获取高精度模式视差图像

视差图像为 SDK 处理图像，获取视差图像需要 `EnableDisparityProcessor` 开启视差计算，再通过“SetDepthCalMode“设置高精度模式（默认为低精度模式），然后通过回调获取。

参考代码片段：

```
auto m_pSDK = new CIMRSDK();
MRCONFIG config = {0};
config.bSlam = false;
config.imgResolution = IMG_640;
config.imgFrequency = 50;
config.imuFrequency = 1000;

m_pSDK->Init(config);
std::queue<cv::Mat> disparity_queue;
int disparity_count = 0;
if (m_pSDK->EnableDisparityProcessor()) {
  m_pSDK->EnableLRConsistencyCheck();
  m_pSDK->SetDepthCalMode(DepthCalMode::HIGH_ACCURACY);
  m_pSDK->RegistDisparityCallback(
      [&disparity_count, &disparity_queue](double time, cv::Mat disparity) {
        if (!disparity.empty()) {
          disparity.convertTo(disparity, CV_8U, 255. / (16 * 64));
          cv::applyColorMap(disparity, disparity, cv::COLORMAP_JET);
          disparity.setTo(0, disparity == -16);
          disparity_queue.push(disparity);
          ++disparity_count;
        }
      });
}
auto &&time_beg = times::now();
while (true) {
  if (!disparity_queue.empty()) {
    cv::imshow("disparity", disparity_queue.front());
    disparity_queue.pop();
  }
  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') { // ESC/Q
    break;
  }
}
delete m_pSDK;
return 0;
```

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_disparity_with_high_accuracy.cpp 。

小技巧： 开启高精度模式会影响视差计算的速度，用户酌情使用。

2.3.8 获取深度图像

深度图像为 SDK 处理图像，获取深度图像需要 `EnableDepthProcessor` 开启深度计算，然后通过回调获取。

参考代码片段：

```
auto m_pSDK = new CIMRSDK();
MRCONFIG config = {0};
config.bSlam = false;
config.imgResolution = IMG_640;
config.imgFrequency = 50;
config.imuFrequency = 1000;

m_pSDK->Init(config);
std::queue<cv::Mat> depth_queue;
int depth_count = 0;
if (m_pSDK->EnableDepthProcessor()) {
  m_pSDK->RegistDepthCallback(
      [&depth_count, &depth_queue](double time, cv::Mat depth) {
        if (!depth.empty()) {
          depth.convertTo(depth, CV_16U, 1000.0);
          depth_queue.push(depth);
          ++depth_count;
        }
      });
}
auto &&time_beg = times::now();
while (true) {
  if (!depth_queue.empty()) {
    cv::imshow("depth", depth_queue.front());
    depth_queue.pop();
  }
  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') { // ESC/Q
    break;
  }
}
delete m_pSDK;
return 0;
```

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.cpp 。

小技巧： 预览深度图某区域的值，请见 get_depth_with_region.cpp 。

2.3.9 获取物体检测图像

物体检测图像为 SDK 处理图像，获取物体检测像需要 `EnableDetectorProcessor` 开启计算，然后通过回调获取。

参考代码片段：

```
auto m_pSDK = new CIMRSDK();
MRCONFIG config = {0};
config.bSlam = false;
config.imgResolution = IMG_640;
config.imgFrequency = 50;
config.imuFrequency = 1000;

m_pSDK->Init(config);
std::queue<cv::Mat> detector_queue;
int detector_count = 0;
std::string name[10] = {
    "BG",    "PERSON", "PET_CAT",   "PET_DOG", "SOFA",
    "TABLE", "BED",    "EXCREMENT", "WIRE",    "KEY",
};
m_pSDK->EnableDetectorProcessor();
m_pSDK->RegistDetectorCallback(
    [&detector_count, &detector_queue, &name](DetectorInfo info) {
      if (!info.img.empty()) {
        detector_queue.push(info.img);
        ++detector_count;
        for (int i = 0; i < info.finalBoxInfo.size(); ++i) {
          BoxInfo &obj = info.finalBoxInfo[i];
        }
      }
    });
auto &&time_beg = times::now();
while (true) {
  if (!detector_queue.empty()) {
    cv::imshow("detector", detector_queue.front());
    detector_queue.pop();
  }
  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') { // ESC/Q
    break;
  }
}

delete m_pSDK;
return 0;
```

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_detector.cpp 。

2.3.10 获取点云图像

点云图像深度图像为 SDK 处理图像，获取深度图像需要 `EnablePointProcessor` 开启点云计算，然后通过回调获取

参考代码片段：

```
auto m_pSDK = new CIMRSDK();
MRCONFIG config = {0};
config.bSlam = false;
config.imgResolution = IMG_640;
config.imgFrequency = 50;
config.imuFrequency = 1000;

m_pSDK->Init(config);
std::queue<cv::Mat> points_queue;
int points_count = 0;
if (m_pSDK->EnablePointProcessor()) {
  // m_pSDK->EnableLRConsistencyCheck();
  // m_pSDK->SetDepthCalMode(DepthCalMode::HIGH_ACCURACY);
  m_pSDK->RegistPointCloudCallback(
      [&points_count, &points_queue](double time, cv::Mat points) {
        if (!points.empty()) {
          points_queue.push(points);
          ++points_count;
        }
      });
}
PCViewer pcviewer;
auto &&time_beg = times::now();
while (true) {
  if (!points_queue.empty()) {
    pcviewer.Update(points_queue.front());
    points_queue.pop();
  }
  char key = static_cast<char>(cv::waitKey(1));
  if (key == 27 || key == 'q' || key == 'Q') { // ESC/Q
    break;
  }
  if (pcviewer.WasStopped()) {
    break;
  }
}
delete m_pSDK;
return 0;
```

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.cpp 。

注意： 准备好了 PCL 库，编译教程样例时才会有此例子。如果 PCL 库安装到了自定义目录，那么请打开 demo/CMakeLists.txt ，找到 `find_package(PCL)` ，把 `PCLConfig.cmake` 所在目录添加进 `CMAKE_PREFIX_PATH` 。

CHAPTER 3

ROS 接口

3.1 ROS 如何使用

按照*ROS Wrapper 安装*，编译再运行节点。

小技巧: 使用下面命令前需要先在另一个命令行窗口运行 ROS 节点

`rostopic list` 可以列出发布的节点:

```
$ rostopic list
/imsee/camera_info
/imsee/depth
/imsee/depth/compressed
/imsee/depth/compressed/parameter_descriptions
/imsee/depth/compressed/parameter_updates
/imsee/depth/compressedDepth
/imsee/depth/compressedDepth/parameter_descriptions
/imsee/depth/compressedDepth/parameter_updates
/imsee/depth/theora
/imsee/depth/theora/parameter_descriptions
/imsee/depth/theora/parameter_updates
/imsee/disparity
/imsee/disparity/compressed
/imsee/disparity/compressed/parameter_descriptions
/imsee/disparity/compressed/parameter_updates
/imsee/disparity/compressedDepth
/imsee/disparity/compressedDepth/parameter_descriptions
/imsee/disparity/compressedDepth/parameter_updates
/imsee/disparity/theora
/imsee/disparity/theora/parameter_descriptions
/imsee/disparity/theora/parameter_updates
/imsee/image/camera_info
```

(下页继续)

(续上页)

```
/imsee/image/detector
/imsee/image/detector/compressed
/imsee/image/detector/compressed/parameter_descriptions
/imsee/image/detector/compressed/parameter_updates
/imsee/image/detector/compressedDepth
/imsee/image/detector/compressedDepth/parameter_descriptions
/imsee/image/detector/compressedDepth/parameter_updates
/imsee/image/detector/theora
/imsee/image/detector/theora/parameter_descriptions
/imsee/image/detector/theora/parameter_updates
/imsee/image/left
/imsee/image/left/compressed
/imsee/image/left/compressed/parameter_descriptions
/imsee/image/left/compressed/parameter_updates
/imsee/image/left/compressedDepth
/imsee/image/left/compressedDepth/parameter_descriptions
/imsee/image/left/compressedDepth/parameter_updates
/imsee/image/left/theora
/imsee/image/left/theora/parameter_descriptions
/imsee/image/left/theora/parameter_updates
/imsee/image/rectified/camera_info
/imsee/image/rectified/left
/imsee/image/rectified/left/compressed
/imsee/image/rectified/left/compressed/parameter_descriptions
/imsee/image/rectified/left/compressed/parameter_updates
/imsee/image/rectified/left/compressedDepth
/imsee/image/rectified/left/compressedDepth/parameter_descriptions
/imsee/image/rectified/left/compressedDepth/parameter_updates
/imsee/image/rectified/left/theora
/imsee/image/rectified/left/theora/parameter_descriptions
/imsee/image/rectified/left/theora/parameter_updates
/imsee/image/rectified/right
/imsee/image/rectified/right/compressed
/imsee/image/rectified/right/compressed/parameter_descriptions
/imsee/image/rectified/right/compressed/parameter_updates
/imsee/image/rectified/right/compressedDepth
/imsee/image/rectified/right/compressedDepth/parameter_descriptions
/imsee/image/rectified/right/compressedDepth/parameter_updates
/imsee/image/rectified/right/theora
/imsee/image/rectified/right/theora/parameter_descriptions
/imsee/image/rectified/right/theora/parameter_updates
/imsee/image/right
/imsee/image/right/compressed
/imsee/image/right/compressed/parameter_descriptions
/imsee/image/right/compressed/parameter_updates
/imsee/image/right/compressedDepth
/imsee/image/right/compressedDepth/parameter_descriptions
/imsee/image/right/compressedDepth/parameter_updates
/imsee/image/right/theora
/imsee/image/right/theora/parameter_descriptions
/imsee/image/right/theora/parameter_updates
/imsee/imu
/imsee/points
/rosout
/rosout_agg
/tf_static
...
```

小技巧: 除左右目原始图像及 imu 以外的节点，只有主动订阅才会发布，这是为了节省运算量。

`rostopic hz <topic>` 可以检查是否有数据：

```
$  rostopic hz /imsee/imu
subscribed to [/imsee/imu]
average rate: 991.195
  min: 0.000s max: 0.003s std dev: 0.00014s window: 970
average rate: 989.429
  min: 0.000s max: 0.005s std dev: 0.00016s window: 1959
average rate: 986.974
  min: 0.000s max: 0.008s std dev: 0.00024s window: 2944
average rate: 987.813
  min: 0.000s max: 0.008s std dev: 0.00023s window: 3938
...
```

`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：

```
<IMSEE>/ros/
├─src/
|   └─imsee_ros_wrapper
|   ├── CMakeLists.txt
|   ├── config
|   |   └── settings.yaml
|   ├── launch
|   |   ├── display.launch
|   |   └── start.launch
|   ├── nodelet_plugins.xml
|   ├── package.xml
|   ├── rviz
|   |   └── imsee.rviz
|   └── src
|       ├── wrapper_node.cc
|       └── wrapper_nodelet.cc
```

其中 `config/settings` 里，可以配置图像的帧率分辨率与 imu 的频率。

CHAPTER 4

API DOCS

4.1 API

4.1.1 API

class `indem::`**`CIMRSDK`**
This is SDK interface.

Public Functions

bool **`Init`**(MRCONFIG config = {0})
start all thread to get driver data and pose with config

参数 **`config`** -- if config is not use, all default config will be set

void **`Release`**()
release all resources which SDK used

int **`GetCapbility`**(*MRCapbility* cap)
depreciate

ModuleInfo **`GetModuleInfo`**()
query module informations

MoudleAllParam **`GetModuleParams`**()
query calibration of module

void **`RegistModulePoseCallback`**(DataCallback cb, void *param)
add callback function to get pose data

参数

- **`cb`** -- callback function to regist
- **`param`** -- pass to callback function

void **RegistModuleCameraCallback**(ModuleImageCallback cb, void *param)
add callback function to get image data

参数

- **cb** -- callback function to regist
- **param** -- pass to callback function

void **RegistImgCallback**(ImgCallback cb)
add callback function to get rectified image data

参数 **cb** -- callback function to regist

void **RegistRectifiedImgCallback**(RectifiedImgCallback cb)
add callback function to get rectified image data

参数 **cb** -- callback function to regist

void **RegistDisparityCallback**(DisparityCallback cb)
add callback function to get disparity data

参数 **cb** -- callback function to regist

void **RegistDepthCallback**(DepthCallback cb)
add callback function to get depth data

参数 **cb** -- callback function to regist

void **RegistPointCloudCallback**(PointCloudCallback cb)
add callback function to get point cloud

参数 **cb** -- callback function to regist

void **RegistDetectorCallback**(DetectorImgCallback cb)
add callback function to get detector image

参数 **cb** -- callback function to regist

void **RegistModuleIMUCallback**(ModuleIMUCallback cb)
add callback function to get imu data

参数

- **cb** -- callback function to regist
- **param** -- pass to callback function

void **BeforeSlamInit**(void *inParam)
data callback function

参数 **inParam** -- pass params to slam before it inited

bool **ImportMap**(const char *fullpath)
import map to slam module

参数 **fullpath** -- fullpath of the map

返回 true if success, else false.

bool **ExportMap**(const char *fullpath)
export map from slam module

参数 **fullpath** -- fullpath of the map which wante to save

返回 true if success, else false.

void **BeforePluginInit**(const char *pluginName, void *param)
some plugin may be init with params.

This interface pass params to *Init()* function of plugins.

参数

- **pluginName** -- which plugin should be given.
- **param** -- params to be pass.

int **LoadPlugin**(const char *pluginName)
load plugin manually.

参数 **pluginName** -- which plugin should be loaded.

int **AddPluginCallback**(const char *pluginName, const char *callbackName, PluginCallback cb, void *param)
some plugin may have outputs.

Here give a way to get plugin output data.

参数

- **pluginName** -- which plugin should be add.
- **callbackName** -- which callback function should be add or replace.
- **cb** -- pointer of callback function.
- **param** -- which will be passed to callback function.

返回 true if success

int **InvokePluginMethod**(const char *pluginName, const char *methodName, void *inParam, void *outParam)
some plugin may have its own method.

Here give a way to invoke these method.

参数

- **pluginName** -- which plugin should be added.
- **methodName** -- which method should be called.
- **inParam** -- pointer passed in.
- **outParam** -- pointer to retrieved.

返回 false if fail

void **ReleasePlugin**(const char *pluginName)
release plugin manually.

参数 **pluginName** -- which plugin should be release.

bool **EnableRectifyProcessor**()
enable rectify process.

返回 if success return true.

bool **EnableDisparityProcessor**()
enable disparity process.

返回 if success return true.

bool **EnableDepthProcessor**()
enable depth process.

返回 if success return true.

bool **EnablePointProcessor**()
enable point cloud process.

返回 if success return true.

bool **EnableDetectorProcessor**()
enable detector process.

返回 if success return true.

bool **DisableAllProcessors**()
disanable all processors.

返回 if success return true.

bool **EnableLRConsistencyCheck**()
enable L-R Consistency check.

返回 if success return true.

bool **DisableLRConsistencyCheck**()
disanable L-R Consistency check.

返回 if success return true.

bool **SetDepthCalMode**(indem::*DepthCalMode* mode)
Set depth calculate mode.

返回 if success return true.

Public Static Functions

static void **ListPluginsInfo**(int *pluginNum, char **pluginsName)
display all plugins loaded currently

参数

- **pluginNum** -- plugin's count retrieved
- **pluginName** -- plugin's names retrieved

static void **ListPluginInfo**(const char *pluginsName, int *major, int *minor, char *developer)
display all plugin's information

参数

- **pluginName** -- which plugin information to be query
- **major.....plugin's** -- major version
- **minor.....plugin's** -- minor version
- **developer.....plugin's** -- developer name

4.2 Types

4.2.1 ImuData

struct `indem::ImuData`

Public Members

float **accel**[3]
IMU accelerometer data for 3-axis: X, Y, Z.

float **gyro**[3]
IMU gyroscope data for 3-axis: X, Y, Z.

4.2.2 CameraParameter

struct `indem::CameraParameter`

Public Functions

inline *CameraParameter* **resize**(double ratio) const
The distortion parameters, size depending on the distortion model.

For us, the 4 parameters are: (k1, k2, t1, t2).

Public Members

int **_width**
4X4 matrix from camera to imu

int **_height**
width

double **_focal_length**[2]
height

double **_principal_point**[2]
fx,fy

double **_R**[9]
cx,cy

double **_P**[12]
Rectification matrix (stereo cameras only) A rotation matrix aligning the camera coordinate system to the ideal stereo image plane so that epipolar lines in both stereo images are parallel.

double **_K**[9]
Projection/camera matrix [fx' 0 cx' Tx] P = [0 fy' cy' Ty] [0 0 1 0].

double **_D**[4]
Intrinsic camera matrix for the raw (distorted) images.

[fx 0 cx] K = [0 fy cy] [0 0 1]

4.2.3 IMUParameter

struct `indem::IMUParameter`

Public Functions

inline void **printInfo**()
Compensation parameters of gyroscope.

Public Members

double **_g_max**
a_max

double **_sigma_g_c**
g_max

double **_sigma_a_c**
sigma_g_c

double **_sigma_bg**
sigma_a_c

double **_sigma_ba**
sigma_bg

double **_sigma_gw_c**
sigma_ba

double **_sigma_aw_c**
sigma_gw_c

double **_tau**
sigma_aw_c

double **_g**
tau

double **_a0**[4]
g

double **_T_BS**[16]
a0

double **_Acc**[12]
T_BS.

double **_Gyr**[12]
Compensation parameters of accelerometer.

4.2.4 ModuleInfo

struct `indem::ModuleInfo`

Public Functions

inline void **printInfo**()
baseline

Public Members

char **_designer**[32]
id

char **_fireware_version**[32]
designer

char **_hardware_version**[32]
fireware version

char **_lens**[32]
_hardware version

char **_imu**[32]
lens

char **_viewing_angle**[32]
imu

char **_baseline**[32]
viewing angle

4.2.5 SlamParameter

struct `indem::SlamParameter`

Public Members

int **_numImuFrames**
关键帧数量 (默认为 5)

int **_ceres_minIterations**
相邻 imu 帧数量 (默认为 3)

int **_ceres_maxIterations**
最小优化次数 (默认为 3)

double **_ceres_timeLimit**
最大优化次数 (默认为 4)

int **detection_threshold**
ceres time limit(默认为 3.5000000000000003e-02)

int **detection_octaves**
角点阈值 (默认为 30)

int **detection_maxNoKeypoints**
detection octaves(默认为 0)

bool **displayImages**
最大角点数量 (默认为 100)

4.2.6 MoudleAllParam

struct `indem::MoudleAllParam`

Public Members

std::map<*RESOLUTION*, *CameraParameter*> **_right_camera**
CameraParameter of left cameras.

int **_camera_channel** = 1
CameraParameter of right cameras.

double **_baseline** = 0.12
camera channel

IMUParameter **_imu**
baseline

ModuleInfo **_device**
IMU parameter.

SlamParameter **_slam**
device info

4.2.7 ModuleParameters

struct `indem::ModuleParameters`

Public Members

IMUParameter **_imu**
CameraParameter of cameras.

ModuleInfo **_device**
IMU parameter.

SlamParameter **_slam**
device info

4.2.8 BoxInfo

struct `indem::BoxInfo`

Public Members

float **score**
box

float **cx**
reliability score

float **cy**
cx

float **width**
cy

float **height**
width

CLASS_NAME **class_name**
height

int **index**
class

4.2.9 InstanceInfo

struct `indem::`**InstanceInfo**

Public Members

int **instance_id**
class

float **location**[3]
instance id

float **location_cam**[3]
location:xyz

float **time**
camera coords

float **scale**
time

cv::Rect **box**
scale

int **_count**
box

float **front_face_points**[3][3]
count

float **rear_face_points**[3][3]
front face points

bool **visible**
rear face points

int **class_count** = 0
visible

bool **valid**
class count

float **cx**
valid

float **cy**
cx

float **depth**
cy

4.2.10 DetectorInfo

struct `indem::DetectorInfo`

Public Members

cv::Mat **img**
time stamp

std::vector<*BoxInfo*> **finalBoxInfo**
mat of image

4.3 Enums

4.3.1 IMG_RESOLUTION

enum `indem::IMG_RESOLUTION`
Values:

enumerator **IMG_640**

enumerator **IMG_1280**

enumerator **IMG_DEFAULT**

4.3.2 MRCapbility

enum indem::**MRCapbility**
Values:

enumerator **GPU_NVidia**

4.3.3 RESOLUTION

enum indem::**RESOLUTION**
Values:

enumerator **RES_640X400**

enumerator **RES_1280X800**

enumerator **RES_DEFAULT**

4.3.4 DepthCalMode

enum indem::**DepthCalMode**
Values:

enumerator **HIGH_SPEED**

enumerator **HIGH_ACCURACY**

4.3.5 CLASS_NAME

enum indem::**CLASS_NAME**
Values:

enumerator **BG**

enumerator **PERSON**

enumerator **PET_CAT**

enumerator **PET_DOG**

enumerator **SOFA**

enumerator **TABLE**

enumerator **BED**

enumerator **EXCREMENT**

enumerator **WIRE**

enumerator **KEY**

CHAPTER 5

技术支持

5.1 联系我们

如果无法解决问题，可以通过客户服务联系我们。客服微信:INDEMIND。

CHAPTER 6

结语

6.1 致谢

SDK 编写过程中，受到了许多用户的帮助与指导，尤其是我们邀请的内测用户，非常感谢你们的支持。同时，我们也收到了来自广大用户的意见，反馈以及批评：正是你们所反馈的不够友好的用户体验及各种槽点，推动我们改进代码与工程结构，甚至是推动我们自我审视，工程人员该如何面向用户搭建一款简洁易用的 SDK。十分感谢你们的反馈，同样也希望你们可以不断的督促我们。以下是部分具有代表性的内测用户及反馈用户：张征、丘文峰、黄雄铮、何佳乐、陈雄杰、刘贵龙、唐剑、王靖淇、焦明星、Summit。

在知乎问题 如何评价 INDEMIND 双目视觉惯性模组 中我们也收到了匿名用户的批评，同样十分感谢这位用户的反馈，我们有不得已的原因，但也修复了这位用户所提到的部分用户会偶现的问题。非常感谢您的督促。

索引

I