

英特尔D435i深度相机和librealsense安装使用



分类: 机器视觉 个人专栏: 视觉感知 发布时间 2022.07.07 阅读数 2425 评论数 0

介绍

这是一款比较好用的相机。
D435i与D435多一个IMU, 官方介绍见[这里](#)

在这张图片的相机中, 从左到右有四个模组, 其中, 第一个和第三个为一组双目红外摄像头, 这款相机通过双目立体视觉来获取深度, 并且都是红外摄像头而非RGBD, 在光线比较暗时也有比较好的效果; 第二个模组是一个红外发射器, 可以理解为一个补光灯; 最右侧为一个RGB相机。
关于相机的主要信息有: 深度有效范围0.3-3m; 双目全局快门, 深度帧率90帧/秒, 深度图像大小可到1280_720; RGB卷帘快门, 帧率30帧/秒图像大小可到1920_1080; 以及视场FOV信息。

特性	使用环境 室内/室外 图像传感器技术: 全局快门	最大范围: 3-3 米
深度	深度技术: 立体 最小深度距离 (Min-Z): ~0.28 米 深度精度 <2% 位于2 米†	深度视场 (FOV): 67° × 58° 深度输出分辨率: 高达 1280 × 720 深度帧率: 高达 90 帧/秒
RGB	RGB帧分辨率: 1920 × 1080 RGB 帧率: 30 帧/秒 RGB 传感器技术: 卷帘快门	RGB 传感器 FOV (H × V): 69° × 42° RGB传感器分辨率: 2 MP

其最大的好处就是, 内参获取容易且内参精度比较高(通过程序接口读取到内参, 数据流之间的对齐轻松(调用程序接口可以直接实现))。

sdk下载安装

参考[sdk2介绍](#)和[说明文档](#)和[Github地址](#)

我的系统是ubuntu18.04

首先安装依赖:

```
1 | sudo apt-get install git libssl-dev libusb-1.0-0-dev pkg-config libgtk-3-dev
2 | sudo apt-get install libglfw3-dev libgl1-mesa-dev libglu1-mesa-dev
```

下载librealsense, 进入其源码目录下

udev规则设置:

```
1 | sudo cp config/99-realsense-libusb.rules /etc/udev/rules.d/
2 | sudo udevadm control --reload-rules
3 | udevadm trigger
```

构建并应用修补的内核模块:

```
1 | ./scripts/patch-realsense-ubuntu-lts.sh
```

编译:

```
1 | mkdir build
2 | cd build
3 | cmake ..
4 | make -j8
5 | sudo make install -j8
```

编译成功后在/build/examples文件夹下可以看到编译好的例程

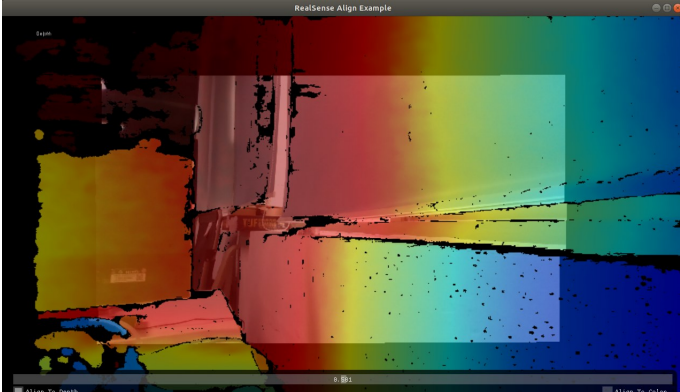
运行前需要检查相机的数据线接到电脑的USB3.0接口

运行前需要检查相机的数据线接到电脑的USB3.0接口

运行前需要检查相机的数据线接到电脑的USB3.0接口

```
1 | ./build/examples/align/rs-align
```

效果如下:



在安装成功后，若要调用librealsense接口，CMakeList.txt内容参考examples/cmake/内的CMakeList.txt文件：

```
1  # License: Apache 2.0. See LICENSE file in root directory.
2  # Copyright(c) 2019 Intel Corporation. All Rights Reserved.
3  cmake_minimum_required(VERSION 3.1.0)
4
5  project(hello_librealsense2)
6
7  # Find librealsense2 installed package
8  find_package(realsense2 REQUIRED)
9
10 # Enable C++11
11 set(CMAKE_CXX_STANDARD 11)
12 set(CMAKE_CXX_STANDARD_REQUIRED TRUE)
13
14 # Add the application sources to the target
15 add_executable(${PROJECT_NAME} hello_librealsense2.cpp)
16
17 # Link librealsense2 to the target
18 target_link_libraries(${PROJECT_NAME} ${realsense2_LIBRARY})
```

使用

这部分内容多参考[示例说明](#)，以及[API文档](#)，示例程序对应librealsense源码目录下的examples/内容。源码目录下的examples/sensor-control/api_how_to.h封装了一些实用的例子：

Code Overview

Please follow the [api_how_to.h](#) for detailed documentation and explanations
This file contains detailed explanations and code on how to do the following:

- [get a realsense device](#)
- [print device information](#)
- [get device name](#)
- [get sensor name](#)
- [get a sensor from a device](#)
- [get sensor option](#)
- [get depth units](#)
- [get field of view](#)
- [get extrinsics](#)
- [change sensor option](#)
- [choose a streaming profile](#)
- [start streaming a profile](#)

借鉴我所看到的一些代码和官方的例子，我自己也做部分使用总结。

获取设备信息

```
1  rs2::device get_a_realsense_device()
2  {
3      // 实例化context
4      rs2::context ctx;
5
6      // context的query_devices方法获取设备列表，返回一个device_list对象
7      rs2::device_list devices = ctx.query_devices();
8
9      //实例化device
10     rs2::device selected_device;
11     if (devices.size() == 0)
12     {
13         std::cerr << "No device connected!!!!" << std::endl;
14         exit(0);
15     }
16     else
17     {
18         //打印设备数量
19         std::cout<<"device num: "<<devices.size()<<std::endl;
20         //将第一个设备传给device实例化得到的selected_device
21         selected_device = devices[0];
22     }
23     //至此，selected_device就代表我们的当前设备，将它返回
24     return selected_device;
25 }
```

读取设备参数

在修改选项之前需要好好看看这个函数打印出来的信息。

```
1  void get_sensor_option(const rs2::sensor& sensor)
2  {
3      std::cout << "Sensor name: " << sensor.get_info(RS2_CAMERA_INFO_NAME) <<
4      std::endl;
5      std::cout << "Sensor supports the following options:\n" << std::endl;
6      // 轮循所有参数
7      for (int i = 0; i < static_cast<int>(RS2_OPTION_COUNT); i++)
8      {
9          // 隐式转换，将数字转为枚举类型
10         rs2_option option_type = static_cast<rs2_option>(i);
11
12         // 首先判断传感器是否支持这个选项设置
13         if (sensor.supports(option_type))
14         {
15             // 获取并打印对当前选项的描述
16             const char* description = sensor.get_option_description(option_type);
17
18             // 获取并打印当前选项的值
19             float current_value = sensor.get_option(option_type);
20
21             // 获取选项的取值范围、默认值、取值步长
22             rs2::option_range range = sensor.get_option_range(option_type);
23             float default_value = range.def;
```

```

24:         float maximum_supported_value = range.max;
25:         float minimum_supported_value = range.min;
26:         float difference_to_next_value = range.step;
27:
28:         // 打印
29:         std::cout << "      " << i << " : " << "<" << option_type << ">" <<
30:         std::endl;
31:         std::cout << "          Description   : " << description << std::endl;
32:         std::cout << "          Min Value     : " << minimum_supported_value <<
33:         std::endl;
34:         std::cout << "          Max Value      : " << maximum_supported_value <<
35:         std::endl;
36:         std::cout << "          Step           : " << difference_to_next_value <<
37:         std::endl;
38:         std::cout << "          Default Value : " << default_value << std::endl;
39:         std::cout << "          Current Value : " << current_value << std::endl;
40:         std::cout << std::endl;
41:     }
42:     else
43:     {
44:         //std::cout << "          is not supported by this sensor" << std::endl;
45:     }
46:     std::cout << std::endl;
47:     std::cout << std::endl;
48: }

```

这里注意，需要将rs2::device实例化得到的selected_device传递到rs2::sensor，但一个设备一般有好几个不同类型的传感器，比如D435i有Stereo Module、RGB Camera、Motion Module三种类型的传感器，所以函数的使用体现为

```
1 std::vector<rs2::sensor> sensors = selected_device.query_sensors();
2 for (rs2::sensor sensor : sensors){
3     get_sensor_option(sensor);
4 }
```

关于详细参数列表参考rs_option.h:

```

// Brief Defines general configuration controls.
// These can generally be mapped to camera UVC controls, and can be set /
// 1
typedef enum rs2_option
{
    RS2_OPTION_BACKLIGHT_COMPENSATION, /* Enable / disable color backli
    RS2_OPTION_BRIGHTNESS, /* Color image brightness*/
    RS2_OPTION_CONTRAST, /* Color image contrast*/
    RS2_OPTION_EXPOSURE, /* Controls exposure time of color camera.
    RS2_OPTION_GAIN, /* Color image gain*/
    RS2_OPTION_GAMMA, /* Color image gamma setting*/
    RS2_OPTION_HUE, /* Color image hue*/
    RS2_OPTION_SATURATION, /* Color image saturation setting*/
    RS2_OPTION_SHARPNESS, /* Color image sharpness setting*/
    RS2_OPTION_WHITE_BALANCE, /* Controls white balance of color ima
    RS2_OPTION_ENABLE_AUTO_EXPOSURE, /* Enable / disable color image
    RS2_OPTION_ENABLE_AUTO_WHITE_BALANCE, /* Enable / disable color
    RS2_OPTION_VISUAL_PRESET, /* Provide access to several recommende
    RS2_OPTION_LASER_POWER, /* Power of the laser emitter, with 0 me
    RS2_OPTION_ACCURACY, /* Set the number of patterns projected per
    RS2_OPTION_MOTION_RANGE, /* Motion vs. Range trade-off, with Low
    RS2_OPTION_FILTER_OPTION, /* Set the filter to apply to each dep
    RS2_OPTION_CONFIDENCE_THRESHOLD, /* The confidence level thresho
    RS2_OPTION_EMITTER_SELECT, /* Emitter select: 0 - disable all emi
    RS2_OPTION_FRAMES_QUEUE_SIZE, /* Number of frames the user is al
    RS2_OPTION_TOTAL_FRAME_DROPS, /* Total number of detected frame
    RS2_OPTION_AUTO_EXPOSURE_MODE, /* Auto-Exposure modes: Static, A
    RS2_OPTION_POWER_LINE_FREQUENCY, /* Power Line Frequency control
    RS2_OPTION_ASIC_TEMPERATURE, /* Current ASIC Temperature */
    RS2_OPTION_ERROR_POLLING_ENABLED, /* disable error handling */
    RS2_OPTION_PROJECTOR_TEMPERATURE, /* Current Projector Temperatu
    RS2_OPTION_OUTPUT_TRIGGER_ENABLED, /* Enable / disable trigger t
    RS2_OPTION_MOTION_MODULE_TEMPERATURE, /* Current Motion-Module T
    RS2_OPTION_DEPTH_UNITS, /* Number of meters represented by a sin
    RS2_OPTION_ENABLE_MOTION_CORRECTION, /* Enable/Disable automatic
    RS2_OPTION_AUTO_EXPOSURE_PRIORITY, /* Allows sensor to dynamical
    RS2_OPTION_COLOR_SCHEME, /* Color scheme for data visualization
    RS2_OPTION_HISTOGRAM_EQUALIZATION_ENABLED, /* Perform histogram
    RS2_OPTION_MIN_DISTANCE, /* Minimal distance to the target */
    RS2_OPTION_MAX_DISTANCE, /* Maximum distance to the target */
    RS2_OPTION_TEXTURE_SOURCE, /* Texture mapping stream unique ID */
    RS2_OPTION_FILTER_MAGNITUDE, /* The 20-filter effect. The specif
    RS2_OPTION_FILTER_SMOOTH_ALPHA, /* 20-filter parameter controls
    RS2_OPTION_FILTER_SMOOTH_DEPTH, /* 20-filter range/validity thro
    RS2_OPTION_DEPTH_SCALE, /* Enhance depth data post-processing wit
    RS2_OPTION_STEREO_BASELINE, /* The distance in mm between the
    RS2_OPTION_VGA_EXPOSURE, /* Controls the exposure of the VGA

```

更改参数

```

1 void change_sensor_option(const rs2::sensor& sensor, rs2_option option_type, float
2 requested_value)
3 {
4     // 首先判断传感器是否支持这个选项设置
5     if (!sensor.supports(option_type))
6     {
7         std::cerr << "option is not supported by sensor" << "<" <<
8         sensor.get_info(RS2_CAMERA_INFO_NAME) << ">" <<std::endl;
9         return;
10    }
11    else
12    {
13        // 使用set_option函数给选项赋新值
14        sensor.set_option(option_type, requested_value);
15        std::cout << "Change " << "<" << option_type << ">" << " to " << ": " <<
16        requested_value << std::endl;
17    }
18 }

```

与读取设备参数的操作一样,这里要注意一下用法,需要轮询前面得到的sensor:sensors,对不同类型的传感器进行各自的设置。

```
1  for (rs2::sensor sensor : sensors){
2      ++index;
3      if (index == 1) { // Stereo Module
4          change_sensor_option(sensor, RS2_OPTION_ENABLE_AUTO_EXPOSURE, 1);
5          change_sensor_option(sensor, RS2_OPTION_AUTO_EXPOSURE_LIMIT, 50000);
6          change_sensor_option(sensor, RS2_OPTION_EMITTER_ENABLED, 1);
7      }
8      if (index == 2) { // RGB Camera
```

```
9     }
10     change_sensor_option(sensor, RS2_OPTION_EXPOSURE, 80.f);
11     if (index == 3){// Motion Module
12         change_sensor_option(sensor, RS2_OPTION_ENABLE_MOTION_CORRECTION, 0);
13     }
14     get_sensor_option(sensor);
15 }
```

获取深度信息的尺度

像素值乘以这个尺度因子就是深度信息。

```
1 float get_depth_units(const rs2::sensor& sensor)
2 {
3     if (rs2::depth_sensor dpt_sensor = sensor.as<rs2::depth_sensor>())
4     {
5         float scale = dpt_sensor.get_depth_scale();
6         std::cout << "Scale factor for depth sensor is: " << scale << std::endl;
7         return scale;
8     }
9     else
10     {
11         std::cout << "Given sensor is not a depth sensor" << scale << std::endl;
12     }
13 }
```

同样需要注意sensor类型, 在D425i上应该归属Stereo Module这个传感器类型, 不过这个scale的值一般都是0.001, 不读也行。

获取图像

获取图像, 包括RGB图、深度图、IR图, 并转为opencv的格式。

```
1 // 首先实例化pipeline和config
2 rs2::pipeline pipe;
3 rs2::config cfg;
4
5 // RGB stream
6 cfg.enable_stream(RS2_STREAM_COLOR, 640, 480, RS2_FORMAT_BGR8, 30);
7
8 // Depth stream
9 cfg.enable_stream(RS2_STREAM_DEPTH, 640, 480, RS2_FORMAT_Z16, 30);
10
11 // IMU stream
12 cfg.enable_stream(RS2_STREAM_ACCEL, RS2_FORMAT_MOTION_XYZ32F, 250);
13 cfg.enable_stream(RS2_STREAM_GYRO, RS2_FORMAT_MOTION_XYZ32F, 400);
14
15 // IR stream
16 cfg.enable_stream(RS2_STREAM_INFRARED, 1, 640, 480, RS2_FORMAT_Y8, 30);
17 cfg.enable_stream(RS2_STREAM_INFRARED, 2, 640, 480, RS2_FORMAT_Y8, 30);
18
19 // 开启
20 rs2::pipeline_profile profile = pipe.start(cfg);
21
22 while(1){
23     // 等待一帧数据流
24     rs2::frameset frames = pipe.wait_for_frames();
25
26     // 取得每一张影像
27     rs2::depth_frame depth_frame = frames.get_depth_frame(); // 获取深度图像数据
28     rs2::video_frame color_frame = frames.get_color_frame(); // 获取彩色图像数据
29
30     rs2::frame irL_frame = frames.get_infrared_frame(1); // 左红外相机图像
31     rs2::frame irR_frame = frames.get_infrared_frame(2); // 右红外相机图像
32
33     // 转Opencv
34     Mat color_image(Size(640, 480), CV_8UC3,
35 (void*)color_frame.get_data(), Mat::AUTO_STEP); // 彩色图像
36     Mat depth_image(Size(640, 480), CV_16U,
37 (void*)depth_frame.get_data(), Mat::AUTO_STEP); // 深度图像
38     Mat irL_image (Size(640, 480), CV_8UC1, (void*)irL_frame.get_data()
39 ,Mat::AUTO_STEP); // 左红外相机图像
40     Mat irR_image (Size(640, 480), CV_8UC1, (void*)irR_frame.get_data()
41 ,Mat::AUTO_STEP); // 右红外相机图像
42     imshow("color", color_image);
43     imshow("depth", depth_image);
44     imshow("irL", irL_image);
45     imshow("irR", irR_image);
46     waitKey(1);
47 }
```

获取imu数据

这里要注意, D435i的IMU只输出三轴加速度与三轴角速度数据, 不会像T265一样输出姿态。

程序内容其实与获取图像一样, 但是SDK没有给封装为完整函数, 在这里会稍微加几句话。

```
1 // Get gyro measures
2 rs2::frame f_gyro = frames.first_or_default(RS2_STREAM_GYRO);
3 rs2_vector gyro_data = f_gyro.as<rs2::motion_frame>().get_motion_data();
4 std::cout << gyro_data << std::endl;
5
6 // Get accelerometer measures
7 rs2::frame f_accel = frames.first_or_default(RS2_STREAM_ACCEL);
8 rs2_vector accel_data = f_accel.as<rs2::motion_frame>().get_motion_data();
9 std::cout << accel_data << std::endl;
```

对比获取图像的函数封装:

```
/**
 * Retrieve the first color frame, if no frame is found, search for the co
 * \return video frame - first found color frame.
 */
video_frame get_color_frame() const
{
    auto f = first_or_default(RS2_STREAM_COLOR);

    if (!f)
    {
        auto ir = first_or_default(RS2_STREAM_INFRARED);
        if (ir && ir.get_profile().format() == RS2_FORMAT_RGB8)
```

```
    } f = ir;
    return f;
}
```

通过回调函数获取图像以及IMU数据

以上是在while(1)内获取图像以及IMU数据, 这会堵塞程序。

使用回调的方式可以让程序更加灵活, 主要就是回调函数的实现, 以及将pipe.start(cfg);改为pipe.start(cfg, stream_callback);

```
1 void stream_callback(const rs2::frame& frame){
2
3     static double frames_fs = 0;
4     static double gyro_fs = 0;
5     static double accel_fs = 0;
6     std::cout << std::endl;
7     std::cout << "frames_fs: " << frames_fs << std::endl;
8     std::cout << "gyro_fs: " << gyro_fs << std::endl;
9     std::cout << "accel_fs: " << accel_fs << std::endl;
10    std::cout << std::endl;
11
12
13    if(rs2::frameset frames = frame.as<rs2::frameset>()){
14
15        //取得每一张图片
16        rs2::depth_frame depth_frame = frames.get_depth_frame(); // 获取深度图像数据
17        rs2::video_frame color_frame = frames.get_color_frame(); // 获取彩色图像数据
18        rs2::frame irL_frame = frames.get_infrared_frame(1); // 左红外相机图像
19        rs2::frame irR_frame = frames.get_infrared_frame(2); // 右红外相机图像
20
21        // 图像转Opencv格式
22        Mat color_image(Size(640, 480), CV_8UC3,
23        (void*)color_frame.get_data(),Mat::AUTO_STEP); // 彩色图像
24        Mat depth_image(Size(640, 480), CV_16U,
25        (void*)depth_frame.get_data(),Mat::AUTO_STEP); // 深度图像
26        Mat irL_image (Size(640, 480), CV_8UC1, (void*)irL_frame.get_data()
27        ,Mat::AUTO_STEP); // 左红外相机图像
28        Mat irR_image (Size(640, 480), CV_8UC1, (void*)irR_frame.get_data()
29        ,Mat::AUTO_STEP); // 右红外相机图像
30        imshow("color", color_image);
31        imshow("depth", depth_image);
32        imshow("irL", irL_image);
33        imshow("irR", irR_image);
34        waitKey(1);
35
36        // 计算、打印帧率
37        static double frames_t_last = 0;
38        // Get the timestamp of the current frame
39        double frames_t_now = frames.get_timestamp();
40        if(frames_t_last > 0){
41            frames_fs = 1000/(frames_t_now - frames_t_last);
42        }
43        frames_t_last = frames_t_now;
44
45    }else if(rs2::motion_frame m_frame = frame.as<rs2::motion_frame>()){
46        if (m_frame.get_profile().stream_name() == "Gyro")
47        {
48            // Get gyro measures
49            rs2_vector gyro_data = m_frame.get_motion_data();
50            // std::cout << gyro_data << std::endl;
51
52            // 计算、打印帧率
53            static double gyro_t_last = 0;
54            // Get the timestamp of the current frame
55            double gyro_t_now = m_frame.get_timestamp();
56            if(gyro_t_last > 0){
57                gyro_fs = 1000/(gyro_t_now - gyro_t_last);
58                std::cout << "gyro_fs: " << gyro_fs << std::endl;
59            }
60            gyro_t_last = gyro_t_now;
61        }
62        else if (m_frame.get_profile().stream_name() == "Accel")
63        {
64            // Get accelerometer measures
65            rs2_vector accel_data = m_frame.as<rs2::motion_frame>
66            ().get_motion_data();
67            // std::cout << accel_data << std::endl;
68
69            // 计算、打印帧率
70            static double accel_t_last = 0;
71            // Get the timestamp of the current frame
72            double accel_t_now = m_frame.get_timestamp();
73            if(accel_t_last > 0){
74                accel_fs = 1000/(accel_t_now - accel_t_last);
75                std::cout << "accel_fs: " << accel_fs << std::endl;
76            }
77            accel_t_last = accel_t_now;
78        }
79    }
80}
81
82int main()
83{
84    // 首先实例化pipeline和config
85    rs2::pipeline pipe;
86    rs2::config cfg;
87
88    // RGB stream
89    cfg.enable_stream(RS2_STREAM_COLOR, 640, 480, RS2_FORMAT_BGR8, 30);
90    // Depth stream
91    cfg.enable_stream(RS2_STREAM_DEPTH, 640, 480, RS2_FORMAT_Z16, 30);
92    // IMU stream
93    cfg.enable_stream(RS2_STREAM_ACCEL, RS2_FORMAT_MOTION_XYZ32F);
94    cfg.enable_stream(RS2_STREAM_GYRO, RS2_FORMAT_MOTION_XYZ32F);
95    // IR stream
96    cfg.enable_stream(RS2_STREAM_INFRARED, 1, 640, 480, RS2_FORMAT_Y8, 30);
97    cfg.enable_stream(RS2_STREAM_INFRARED, 2, 640, 480, RS2_FORMAT_Y8, 30);
```

```
97 | pipe.start(cfg, stream_callback);
98 |
    | while(1){}
```

将其它图像与深度图像对齐

这属于是空间上的对齐而不是时间上的对齐，也就是像素点的一一对应。
由于这个对齐操作耗时久一点，所以不放在回调里，改到while(1)里。

```
1 | double frames_fs = 0;
2 | rs2::frameset fs_to_align;
3 | int frames_point = 0;
4 | void stream_callback(const rs2::frame& frame){
5 |
6 |     // static double frames_fs = 0;
7 |     static double gyro_fs = 0;
8 |     static double accel_fs = 0;
9 |     std::cout << std::endl;
10 |    std::cout << "frames_fs: " << frames_fs << std::endl;
11 |    std::cout << "gyro_fs: " << gyro_fs << std::endl;
12 |    std::cout << "accel_fs: " << accel_fs << std::endl;
13 |    std::cout << std::endl;
14 |
15 |    if(rs2::frameset frames = frame.as<rs2::frameset>()){
16 |
17 |        fs_to_align = frames;
18 |        frames_point = 1;
19 |
20 |    }else if(rs2::motion_frame m_frame = frame.as<rs2::motion_frame>()){
21 |        if (m_frame.get_profile().stream_name() == "Gyro")
22 |        {
23 |            // Get gyro measures
24 |            rs2_vector gyro_data = m_frame.get_motion_data();
25 |            // std::cout << "gyro_data" << gyro_data << std::endl;
26 |
27 |            // 计算、打印帧率
28 |            static double gyro_t_last = 0;
29 |            // Get the timestamp of the current frame
30 |            double gyro_t_now = m_frame.get_timestamp();
31 |            if(gyro_t_last > 0){
32 |                gyro_fs = 1000/(gyro_t_now - gyro_t_last);
33 |            }
34 |            gyro_t_last = gyro_t_now;
35 |        }
36 |        else if (m_frame.get_profile().stream_name() == "Accel")
37 |        {
38 |            // Get accelerometer measures
39 |            rs2_vector accel_data = m_frame.as<rs2::motion_frame>
40 |            ().get_motion_data();
41 |            // std::cout << "accel_data" << accel_data << std::endl;
42 |
43 |            // 计算、打印帧率
44 |            static double accel_t_last = 0;
45 |            // Get the timestamp of the current frame
46 |            double accel_t_now = m_frame.get_timestamp();
47 |            if(accel_t_last > 0){
48 |                accel_fs = 1000/(accel_t_now - accel_t_last);
49 |            }
50 |            accel_t_last = accel_t_now;
51 |        }
52 |    }
53 | }
54 |
55 | int main()
56 | {
57 |     // 首先实例化pipeline和config
58 |     rs2::pipeline pipe;
59 |     rs2::config cfg;
60 |
61 |
62 |     // RGB stream
63 |     cfg.enable_stream(RS2_STREAM_COLOR, 640, 480, RS2_FORMAT_BGR8, 30);
64 |     // Depth stream
65 |     cfg.enable_stream(RS2_STREAM_DEPTH, 640, 480, RS2_FORMAT_Z16, 30);
66 |     // IMU stream
67 |     cfg.enable_stream(RS2_STREAM_ACCEL, RS2_FORMAT_MOTION_XYZ32F);
68 |     cfg.enable_stream(RS2_STREAM_GYRO, RS2_FORMAT_MOTION_XYZ32F);
69 |     // IR stream
70 |     cfg.enable_stream(RS2_STREAM_INFRARED, 1, 640, 480, RS2_FORMAT_Y8, 30);
71 |     cfg.enable_stream(RS2_STREAM_INFRARED, 2, 640, 480, RS2_FORMAT_Y8, 30);
72 |
73 |
74 |     // Create a rs2::align object.
75 |     // rs2::align allows us to perform alignment of depth frames to others frames
76 |     //The "align_to" is the stream type to which we plan to align depth frames.
77 |     rs2::align align(RS2_STREAM_COLOR);
78 |
79 |
80 |     pipe.start(cfg, stream_callback);
81 |
82 |
83 |
84 |     while(1){
85 |
86 |         if(frames_point == 1){
87 |
88 |             // 计算、打印帧率
89 |             static double frames_t_last = 0;
90 |             // Get the timestamp of the current frame
91 |             double frames_t_now = fs_to_align.get_timestamp();
92 |             if(frames_t_last > 0){
93 |                 frames_fs = 1000/(frames_t_now - frames_t_last);
94 |             }
95 |             frames_t_last = frames_t_now;
96 |
97 |
98 |             // Perform alignment here
99 |             rs2::frameset aligned_frameset = align.process(fs_to_align);
100 |
101 |             // 获取对齐后的RGB图像与深度图像
102 |             rs2::video_frame aligned_color_frame =
```

```

102 aligned_frameset.get_color_frame();
103 rs2::depth_frame aligned_depth_frame =
104 aligned_frameset.get_depth_frame();

105 // 取得每一张图片
106 rs2::depth_frame depth_frame = fs_to_align.get_depth_frame(); // 获取深度图像数据
107 rs2::video_frame color_frame = fs_to_align.get_color_frame(); // 获取彩色图像数据
108 rs2::frame irL_frame = fs_to_align.get_infrared_frame(1); // 左红外相机图像
109 rs2::frame irR_frame = fs_to_align.get_infrared_frame(2); // 右红外相机图像
110
111 // // 图像转Opencv格式
112 Mat aligned_color_image(Size(640, 480), CV_8UC3,
113 (void*)aligned_color_frame.get_data(),Mat::AUTO_STEP); // 彩色图像
114 Mat aligned_depth_image(Size(640, 480), CV_16U,
115 (void*)aligned_depth_frame.get_data(),Mat::AUTO_STEP); // 深度图像
116 Mat color_image(Size(640, 480), CV_8UC3,
117 (void*)color_frame.get_data(),Mat::AUTO_STEP); // 彩色图像
118 Mat depth_image(Size(640, 480), CV_16U,
119 (void*)depth_frame.get_data(),Mat::AUTO_STEP); // 深度图像
120 Mat irL_image (Size(640, 480), CV_8UC1, (void*)irL_frame.get_data()
121 ,Mat::AUTO_STEP); // 左红外相机图像
122 Mat irR_image (Size(640, 480), CV_8UC1, (void*)irR_frame.get_data()
123 ,Mat::AUTO_STEP); // 右红外相机图像
124 imshow("aligned_color_image", aligned_color_image);
125 imshow("aligned_depth_image", aligned_depth_image);
126 imshow("color", color_image);
127 imshow("depth", depth_image);
128 imshow("irL", irL_image);
129 imshow("irR", irR_image);
130 waitKey(1);
131 frames_point = 0;
132 }
133 }
134 return 0;
135 }

```

程序的主要内容在于：

```

1 // Create a rs2::align object.
2 // rs2::align allows us to perform alignment of depth frames to others frames
3 //The "align_to" is the stream type to which we plan to align depth frames.
4 rs2::align align(RS2_STREAM_COLOR);
5
6
7 // Perform alignment here
8 rs2::frameset aligned_frameset = align.process(fs_to_align);

```

各传感器的内参获取

内参可以通过程序获取。

RGB、R相机内参内容如下，深度图的内参与R相机相同，包括图像的长宽、内参矩阵、相机模型、畸变系数：

```

/** \brief Video stream intrinsics. */
typedef struct rs2_intrinsics
{
    int width; //< width of the image in pixels */
    int height; //< height of the image in pixels */
    float ppx; //< Horizontal coordinate of the principal point of the image, as a pixel offset from the left edge */
    float ppy; //< Vertical coordinate of the principal point of the image, as a pixel offset from the top edge */
    float fx; //< Focal length of the image plane, as a multiple of pixel width */
    float fy; //< Focal length of the image plane, as a multiple of pixel height */
    rs2_distortion_model; //< Distortion model of the image */
    float coeffs[5]; //< Distortion coefficients. Order for Brown-Conrady: [k1, k2, p1, p2, k3]. Order for F-Theta Fish-eye: [k1, k2, k3, k4, 0].
} rs2_intrinsics;

```

IMU传感器的内参内容如下，包含高斯白噪声、零偏、以及各自的协方差：

```

/** \brief Motion device intrinsics: scale, bias, and variances. */
typedef struct rs2_motion_device_intrinsic
{
    /* \internal
    * Scale X      cross axis   cross axis   Bias X \n
    * cross axis   Scale Y      cross axis   Bias Y \n
    * cross axis   cross axis   Scale Z      Bias Z */
    float data[3][4]; //< Interpret data array values */

    float noise_variances[3]; //< Variance of noise for X, Y, and Z axis */
    float bias_variances[3]; //< Variance of bias for X, Y, and Z axis */
} rs2_motion_device_intrinsic;

```

```

1 void l_get_intrinsics(const rs2::stream_profile& stream)
2 {
3     // A sensor's stream (rs2::stream_profile) is in general a stream of data with
4     // no specific type.
5     // For video streams (streams of images), the sensor that produces the data
6     // has a lens and thus has properties such
7     // as a focal point, distortion, and principal point.
8     // To get these intrinsics parameters, we need to take a stream and first
9     // check if it is a video stream
10    if (rs2::video_stream_profile video_stream =
11        stream.as<rs2::video_stream_profile>())
12    {
13        try
14        {
15            // 使用get_intrinsics方法获取相机内参
16            rs2_intrinsics intrinsics = video_stream.get_intrinsics();
17            // 处理一些结果
18            auto principal_point = std::make_pair(intrinsics.ppx, intrinsics.ppy);
19            auto focal_length = std::make_pair(intrinsics.fx, intrinsics.fy);
20            rs2_distortion_model = intrinsics.model;
21
22            // 打印内参
23            std::cout << "Principal Point      : " << principal_point.first <<
24            ", " << principal_point.second << std::endl;
25            std::cout << "Focal Length          : " << focal_length.first << ",
26            " << focal_length.second << std::endl;
27            std::cout << "Distortion Model      : " << model << std::endl;
28            std::cout << "Distortion Coefficients : [" << intrinsics.coeffs[0] <<
29            ", " << intrinsics.coeffs[1] << ", " <<
30            intrinsics.coeffs[2] << ", " << intrinsics.coeffs[3] << ", " <<
31            intrinsics.coeffs[4] << "]" << std::endl;
32        }
33    }
34 }

```



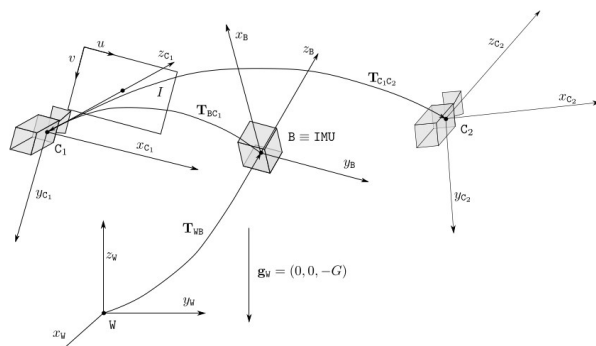
```

26         catch (const std::exception& e)
27         {
28             std::cerr << "Failed to get intrinsics for the given stream. " <<
29             e.what() << std::endl;
30         }
31         else if (rs2::motion_stream_profile motion_stream =
32             stream.as<rs2::motion_stream_profile>())
33         {
34             try
35             {
36                 // 使用get_motion_intrinsics方法获取IMU内参
37                 rs2_motion_device_intrinsic intrinsics =
38                 motion_stream.get_motion_intrinsics();
39
40                 // 打印内参
41                 std::cout << " Scale X      cross axis      cross axis Bias X \n";
42                 std::cout << " cross axis  Scale Y      cross axis Bias Y \n";
43                 std::cout << " cross axis  cross axis  Scale Z      Bias Z \n";
44                 for (int i = 0; i < 3; i++)
45                 {
46                     for (int j = 0; j < 4; j++)
47                     {
48                         std::cout << intrinsics.data[i][j] << " ";
49                     }
50                     std::cout << "\n";
51                 }
52
53                 std::cout << "Variance of noise for X, Y, Z axis \n";
54                 for (int i = 0; i < 3; i++)
55                     std::cout << intrinsics.noise_variances[i] << " ";
56                 std::cout << "\n";
57
58                 std::cout << "Variance of bias for X, Y, Z axis \n";
59                 for (int i = 0; i < 3; i++)
60                     std::cout << intrinsics.bias_variances[i] << " ";
61                 std::cout << "\n";
62             }
63             catch (const std::exception& e)
64             {
65                 std::cerr << "Failed to get intrinsics for the given stream. " <<
66                 e.what() << std::endl;
67             }
68         }
69         else
70         {
71             std::cerr << "Given stream profile has no intrinsics data" << std::endl;
72         }
73     }
74 }
75
76 int main()
77 {
78     // 首先实例化pipeline和config
79     rs2::pipeline pipe;
80     rs2::config cfg;
81
82     // RGB stream
83     cfg.enable_stream(RS2_STREAM_COLOR, 640, 480, RS2_FORMAT_BGR8, 30);
84     // Depth stream
85     cfg.enable_stream(RS2_STREAM_DEPTH, 640, 480, RS2_FORMAT_Z16, 30);
86     // IMU stream
87     cfg.enable_stream(RS2_STREAM_ACCEL, RS2_FORMAT_MOTION_XYZ32F);
88     cfg.enable_stream(RS2_STREAM_GYRO, RS2_FORMAT_MOTION_XYZ32F);
89     // IR stream
90     cfg.enable_stream(RS2_STREAM_INFRARED, 1, 640, 480, RS2_FORMAT_Y8, 30);
91     cfg.enable_stream(RS2_STREAM_INFRARED, 2, 640, 480, RS2_FORMAT_Y8, 30);
92
93     rs2::pipeline_profile pipe_profile = pipe.start(cfg);
94
95     std::cout << std::endl;
96     std::cout << "Get RGB intrinsics :" << std::endl;
97     l_get_intrinsics(pipe_profile.get_stream(RS2_STREAM_COLOR));
98
99     std::cout << std::endl;
100    std::cout << "Get ACCEL intrinsics :" << std::endl;
101    l_get_intrinsics(pipe_profile.get_stream(RS2_STREAM_ACCEL));
102
103    std::cout << std::endl;
104    std::cout << "Get GYRO intrinsics :" << std::endl;
105    l_get_intrinsics(pipe_profile.get_stream(RS2_STREAM_GYRO));
106
107    std::cout << std::endl;
108    std::cout << "Get IR intrinsics :" << std::endl;
109    l_get_intrinsics(pipe_profile.get_stream(RS2_STREAM_INFRARED));
110
111    std::cout << std::endl;
112    std::cout << "Get DEPTH intrinsics :" << std::endl;
113    l_get_intrinsics(pipe_profile.get_stream(RS2_STREAM_DEPTH));
114
115 }

```

各传感器之间的外参获取

默认的坐标系：



如图所示，相机坐标系向前为Z、向左为X、向下为Y；IMU坐标系向前为Z、向左为Y、向上为X；而我们常用的右手坐标系向上为Z、向左为Y、向后为X。

```
1 void l_get_extrinsics(const rs2::stream_profile& from_stream, const
2 rs2::stream_profile& to_stream){
3
4     // If the device/sensor that you are using contains more than a single stream,
5     // and it was calibrated
6     // then the SDK provides a way of getting the transformation between any two
7     // streams (if such exists)
8     try
9     {
10         // Given two streams, use the get_extrinsics_to() function to get the
11         // transformation from the stream to the other stream
12         rs2_extrinsics extrinsics = from_stream.get_extrinsics_to(to_stream);
13         std::cout << "Translation Vector : [" << extrinsics.translation[0] << ", "
14         << extrinsics.translation[1] << ", " << extrinsics.translation[2] << "]\n";
15         std::cout << "Rotation Matrix : [" << extrinsics.rotation[0] << ", " <<
16         extrinsics.rotation[3] << ", " << extrinsics.rotation[6] << "]\n";
17         std::cout << " : [" << extrinsics.rotation[1] << ", " <<
18         extrinsics.rotation[4] << ", " << extrinsics.rotation[7] << "]\n";
19         std::cout << " : [" << extrinsics.rotation[2] << ", " <<
20         extrinsics.rotation[5] << ", " << extrinsics.rotation[8] << "]" << std::endl;
21     }
22     catch (const std::exception& e)
23     {
24         std::cerr << "Failed to get extrinsics for the given streams. " <<
25         e.what() << std::endl;
26     }
27 }
28
29 int main()
30 {
31     // 首先实例化pipeline和config
32     rs2::pipeline pipe;
33     rs2::config cfg;
34
35     // RGB stream
36     cfg.enable_stream(RS2_STREAM_COLOR, 640, 480, RS2_FORMAT_BGR8, 30);
37     // Depth stream
38     cfg.enable_stream(RS2_STREAM_DEPTH, 640, 480, RS2_FORMAT_Z16, 30);
39     // IMU stream
40     cfg.enable_stream(RS2_STREAM_ACCEL, RS2_FORMAT_MOTION_XYZ32F);
41     cfg.enable_stream(RS2_STREAM_GYRO, RS2_FORMAT_MOTION_XYZ32F);
42     // IR stream
43     cfg.enable_stream(RS2_STREAM_INFRARED, 1, 640, 480, RS2_FORMAT_Y8, 30);
44     cfg.enable_stream(RS2_STREAM_INFRARED, 2, 640, 480, RS2_FORMAT_Y8, 30);
45
46     // Create a rs2::align object.
47     // rs2::align allows us to perform alignment of depth frames to others frames
48     //The "align_to" is the stream type to which we plan to align depth frames.
49     rs2::align align(RS2_STREAM_COLOR);
50
51     rs2::pipeline_profile pipe_profile = pipe.start(cfg);
52
53     l_get_extrinsics(pipe_profile.get_stream(RS2_STREAM_COLOR), pipe_profile.get_stream(RS2_STREAM_DEPTH));
54 }
```

OVER

总结的代码我会放到我们码云上, [地址](#)

工欲善其事, 必先利其器~

~

~

结尾玄月~



关于作者



关注者 74

关注

博客

泡泡

积分

勋章

DYNAMIXEL舵机, 单片机keil和
Makefile的C/C++混合编程

阅读数 2306 评论数 2

文档生成神器---doxygen的使用和
代码注释规范

阅读数 7174 评论数 1

关于ROS比赛定位导航movebase
参数相关

阅读数 6053 评论数 1



相关推荐

【机器视觉学习笔记】伽马变换
(C++)

阅读数 348 评论数 0

ViSP学习笔记(七):平面图像投影

阅读数 680 评论数 0

一、从机器人视觉识别领域-三维目
标识别方向讲起

阅读数 4572 评论数 0

数字图像处理(作业二)——离散余
弦变换(DCT)的应用

阅读数 278 评论数 0

picodet 详解——backbone:ESNet



RealSense RGBD

原创文章 作者: 嗣音。如若转载, 请注明出处: 古月居 <http://www.guyuehome.com/38477>



打赏 0



点赞 1



收藏 0



分享

上一篇: 从欧拉角到旋转位移矩阵再到变换矩阵

下一篇: 单目相机内参标定与畸变矫正

为你推荐

事件相机(Event-based camera)模拟器功能介绍

阅读量 448 评论数 0

跟踪数据集汇总

阅读量 116 评论数 0

Tensorflow学习——结合ROS调用模型实现目标识别

阅读量 5431 评论数 0

激光雷达点云操作的工程化分享(三)——RANSAC...

阅读量 771 评论数 0

数字图像处理(一) 绪论

阅读量 3109 评论数 0

神器CLIP: 连接文本和图像, 打造可迁移的视觉模型

阅读量 816 评论数 0

评论(0)

您还未登录, 请[登录](#)后发表或查看评论

阅读量 2451 评论数 0

图像缩放中的插值算法——双线性插值原理及c++实现

阅读量 306 评论数 0

热门泡泡

30秒/分 失眠, 聊聊自己搞ROS的心得体会吧

ros学习路线

30秒/分 TF_REPEATED_DATA ignoring data错误

各位大佬, 有什么ROS定位算法推荐吗

5秒/分 ros中启动gazebo时报错

5秒/分 想买能用ROS2的开发套件。或者开发板

TA的专栏

3篇

3篇

3篇

2篇

4篇

专栏导航 1 / 1



- 齐次坐标、对极约束、基本矩阵与本质矩阵、单应矩阵
- 单目相机内参标定与畸变矫正
- 英特尔D435i深度相机和librealsense安装使用



服务协议

隐私政策

联系我们

友情链接: CSDN-古月居

ROS Wiki

地平线开发者社区



鄂公网安备 42011102004590号 © 2021 古月居 鄂ICP备18024451号-2

