VIO bot 产品说明

发布版本：V001D001

日期：2022-11-17

一、产品简介

该产品内置 VIO 定位算法，是一个即插即用的视觉惯性里程计设备，能提供 200hz 的三维位姿输出，可用于机器人、无人机等领域。

开放系统的用户权限，硬件接口支持 RJ45 网口、HDMI、CAN、UART、USB2.0、USB3.0、I2C，用户可根据紫川提供的 SDK 在设备内灵活修改数据输出的硬件接口，默认由网口输出。

相机、IMU 具有硬件同步触发，也可将设备当成纯传感器设备使用。

二、硬件信息

- 相机：99x85°广角黑白全局相机
- 接口：
  - 千兆网口
  - USB 2.0 Type-A
  - USB 2.0 Type-C
  - USB 3.0 Type-A
  - CAN
  - UART
  - I2C
- 尺寸：76mm x 50mm x 46mm
- 重量：180g
- 安装：1/4 英寸螺丝孔

三、vio 系统配置说明

可以根据实际使用场景修改对应阈值。

配置文件位置：/home/PRR/install/share/config/own.yaml

3.1 系统相关

1. acc_threshold

   ZUPT 最大加速度阈值，默认为 100，如果系统工作正常请勿修改，建议值:30，表示系统在遇到>30 米/秒加速度时进行纯 VO 处理

2. acc_*zero*_velocity

   ZUPT 静止状态最小加速度阈值，默认为 0 即未生效，如果系统工作正常请勿修改，建议值:0.2，表示系统在速度<0.2 米/秒加速度时进行 ZUPT 零速更新

3. ang_*zero*_velocity

   ZUPT 静止状态最小角速度阈值，默认为 0 即未生效，如果系统工作正常请勿修改，建议值:0.1，表示系统在角速度<0.1 米/秒加速度时进行 ZUPT 零速更新

4. acc_*high*_velocity

   ZUPT 静止状态最大加速度阈值，默认为 100 即未生效，如果系统工作正常请勿修改，建议值:10，表示系统在速度>10 米/秒加速度时启动 ZUPT

5. ang_high_velocity

   ZUPT 静止状态最大角速度阈值，默认为 100 即未生效，如果系统工作正常请勿修改，建议值:10，表示系统在速度>10 米/秒加速度时启动 ZUPT

6. big_translation

   ZUPT 初始化期间帧间最大位移，默认 0.5 米，如漂移则重初始化，不建议修改。

7. init*finish*wait_time

   初始化等待时间，默认 7 秒，不建议修改，用于在初始化完成后的 init*finish*wait_time 秒内检测初始化质量，初始化质量差则重新初始化。可根据初始化成功率修改。

8. max*translation*threshold

运行期间帧间最大位移，默认 0，即未生效，生效配置建议 3(米)，这个功能**还有些问题，暂时保持不生效状态**。目的是系统发生大的 drift 后原地初始化并更新坐标。

9. all_frame_cnt

设备开机后长期未初始化的规避项，**默认 168**，不建议修改

3.2 前端相关

1. Fast_threshold

特征点的阈值，可根据使用场景复杂度修改。（**不建议修改**，若修改建议在 15-30 之间）

方法：若画面中的特征点较少，则降低该阈值。

2. Harris_threshold

特征点的质量，可根据特征点跟踪质量修改。（**不建议修改**，若修改建议在 0-0.0001 之间）

方法：若画面中的特征点较多，但特征点的跟踪质量一般，则提高该阈值。

3. min_dist

特征点最小间距，可根据特征点密集程度和相机 fov 修改。（**不建议修改**，若修改建议在 10-30 之间）

4. max_cnt

运行期间特征点数量。（**默认 120**，不建议修改，若修改建议在 80-120 之间）

5. freq

运行期间特征点发布频率。（**默认 8**，**不建议修改**，若修改建议在 7-10 之间）

3.3 后端相关

1. slide*window*size

运行期间滑窗数量。（**默认 7**，不建议修改，若修改建议在 7-10 之间）

2. keyframe_parallax

关键帧判断阈值，可根据特征点密集程度和相机 fov 修改。（不建议修改）

3.4 回环相关

/home/PRR/install/share/pr_loop/launch/PR_LOOP.launch

skip_cnt=1 默认跳 1 个关键帧进行回环，不建议修改，测试过程中可以用 0

reserve_raw_image=300 默认等待 300 帧，即 10 秒，不建议修改。

四、SDK 使用说明

具体代码及描述在 vio_sdk.h 中, 后续会同步修改及更新

```
//#pragma once
#ifndef _vio_sdk_h_
#define _vio_sdk_h_

#if (defined(_WIN32)) //windows
#ifdef VIOSDK_EXPORTS
#define NET_VIO_API  extern "C" __declspec(dllexport)
#else
#define NET_VIO_API  extern "C" __declspec(dllimport)
#endif
#elif
#define NET_ROBO_API extern "C"
#endif

#ifndef _WINDOWS_
#if (defined(_WIN32) || defined(_WIN64))
#include <winsock2.h>
#include <windows.h>
#endif
#endif
```

```
#ifdef _linux_
typedef int BOOL;
typedef void* HANDLE;
#define __stdcall
#endif

#define vio_call __stdcall

/**
 * @brief 连接回调
 * @param [out] state：连接状态,1-已连接，0-已断开，-1-错误
 * @param [out] userData：用户自定义数据
 * @return 无
 */
typedef void(vio_call* vio_connect_callback)(int state, void* userData);

/**
 * @brief 事件回调
 * @param [out] data：数据内容
 * @param [out] length：数据长度
 * @param [out] userData：用户自定义数据
 * @return 无
 */
typedef void(vio_call* vio_event_callback)(const char* data, int length, void*
userData);

//登录信息
typedef struct
```

```
{
	char	ipaddr[32];
	int		port;
	char	username[32];
	char	password[32];
	void* userData;
	vio_connect_callback connect_cb;
	vio_event_callback event_cb;
}vio_login_info_s;

//网络参数
typedef struct
{
	char ipaddr[32];
	char submask[32];
	char gateway[32];
	char macaddr[32];
	int commandPort;
	int heartbeatPort;
}vio_network_cfg_s;

//镜头参数
typedef struct
{
	float focal_length_x;
	float focal_length_y;
	float optical_center_point_x;
	float optical_center_point_y;
```

```
        float radia_distortion_coef_k1;
        float radia_distortion_coef_k2;
        float tangential_distortion_p1;
        float tangential_distortion_p2;
}vio_lens_cfg_s;

//智能参数
typedef struct
{
        int algorithm_enable;//算法启用：1/0
}vio_smart_cfg_s;

//日期时间 SOD
typedef struct
{
        int display;
        int x, y;
}vio_osd_datetime_s;

//OSD 参数
typedef struct
{
        vio_osd_datetime_s osd_datetime;
        int dispaly_feature_pots;//叠加特征点
}vio_osd_cfg_s;

//imu 内参
typedef struct
```

```
{
	float acc_n, acc_w, gyr_n, gyr_w;
}vio_imu_inter_cfg_s;

//4x3 矩阵
typedef struct
{
	float value[4][3];
}vio_mat4x3_s;

//帧类型
typedef enum
{
	vio_frame_imu = 1,
	vio_frame_image_gray,
	vio_frame_tof_cloud,
	vio_frame_image_amp,
	vio_frame_algo_pose,
	vio_frame_loop_pose,
	vio_frame_map_cloud
}vio_frame_type_e;

//帧信息
typedef struct
{
	unsigned int   type;
	unsigned int   timestamp;
	unsigned int   seq;
```

```
        unsigned int   width;
        unsigned int   height;
        unsigned int   length;
}vio_frame_info_s;

/**
 * @brief 获取 SDK 版本信息
 * @param 无
 * @return SDK 版本信息，2 个高字节表示主版本，2 个低字节表示次版本。如 0x00030000：表示版本为 3.0
 */
NET_VIO_API LONG vio_call net_vio_sdk_version();

/**
 * @brief SDK 初始化，程序开始后调用一次即可
 * @param 无
 * @return 成功返回 1，失败返回错误码
 */
NET_VIO_API BOOL vio_call net_vio_sdk_init();

/**
 * @brief SDK 退出，程序结束前调用一次即可
 * @param 无
 * @return 成功返回 1，失败返回错误码
 */
NET_VIO_API BOOL vio_call net_vio_sdk_exit();

/**
```

```
* @brief 设备登录
* @param [in] login_info：登录信息
* @return 登录句柄-成功，NULL-失败
*/
NET_VIO_API HANDLE vio_call net_vio_login(vio_login_info_s loginInfo);

/**
* @brief 设备登出
* @param [in] loginHandle：登录句柄
* @return 成功返回 1，失败返回错误码
*/
NET_VIO_API BOOL vio_call net_vio_logout(HANDLE loginHandle);

/**
* @brief 流数据回调
* @param [out] channel：通道号
* @param [out] frameInfo：帧信息
* @param [out] frame：帧数据
* @param [out] userData：用户自定义数据
* @return 无
*/
typedef void(vio_call *vio_stream_callback)(int channel, const vio_frame_info_s*
frameInfo, const char* frameData, void* userData);

/**
* @brief 流数据连接
* @param [in] loginHandle：登录句柄
```

* @param [in] channel：通道号，1-imu；2-可见光灰度图；3-tof 点云；4-tof 幅度图；5-私有数据

* @param [in] cb：流数据回调

* @return 流句柄-成功，NULL-失败

*/

NET_VIO_API HANDLE vio_call net_vio_stream_connect(HANDLE loginHandle, int channel, vio_stream_callback cb);

/**

* @brief 流数据断开

* @param [in] streamHandle：流句柄

* @return 成功返回 1，失败返回错误码

*/

NET_VIO_API BOOL vio_call net_vio_stream_disconnect(HANDLE streamHandle);

/**

* @brief 获取网络参数

* @param [in] loginHandle：登录句柄

* @param [out] cfg：网络参数指针

* @return 成功返回 1，失败返回错误码

*/

NET_VIO_API BOOL vio_call net_vio_get_cfg_network(HANDLE loginHandle, vio_network_cfg_s* cfg);

/**

* @brief 设置网络参数

* @param [in] loginHandle：登录句柄

* @param [in] cfg：网络参数指针

* @return 成功返回 1，失败返回错误码

*/

NET_VIO_API BOOL vio_call net_vio_set_cfg_network(HANDLE loginHandle, vio_network_cfg_s* cfg);

/**

* @brief 获取镜头参数

* @param [in] loginHandle：登录句柄

* @param [out] cfg：镜头参数指针

* @return 成功返回 1，失败返回错误码

*/

NET_VIO_API BOOL vio_call net_vio_get_cfg_lens(HANDLE loginHandle, vio_lens_cfg_s* cfg);

/**

* @brief 获取镜头参数

* @param [in] loginHandle：登录句柄

* @param [out] cfg：镜头参数指针

* @return 成功返回 1，失败返回错误码

*/

NET_VIO_API BOOL vio_call net_vio_get_cfg_imu_inter(HANDLE loginHandle, vio_imu_inter_cfg_s* cfg);

/**

* @brief 获取 Smart 参数

* @param [in] loginHandle：登录句柄

* @param [out] cfg：Smart 参数

* @return 成功返回 1，失败返回错误码

```
*/
NET_VIO_API BOOL vio_call net_vio_get_cfg_smart(HANDLE loginHandle,
vio_smart_cfg_s* cfg);

/**
* @brief 设置 Smart 参数
* @param [in] loginHandle：登录句柄
* @param [in] cfg：Smart 参数
* @return 成功返回 1，失败返回错误码
*/
NET_VIO_API BOOL vio_call net_vio_set_cfg_smart(HANDLE loginHandle,
vio_smart_cfg_s* cfg);

/**
* @brief 算法重启
* @param [in] loginHandle：登录句柄
* @return 成功返回 1，失败返回错误码
*/
NET_VIO_API BOOL vio_call net_vio_algorithm_reboot(HANDLE loginHandle);

/**
* @brief 重定位
* @param [in] loginHandle：登录句柄
* @return 成功返回 1，失败返回错误码
*/
NET_VIO_API BOOL vio_call net_vio_location_again(HANDLE loginHandle);

/**
```

```
* @brief 获取外参 cam to imu

* @param [in] loginHandle：登录句柄

* @param [out] mat：矩阵

* @return 成功返回 1，失败返回错误码

*/

NET_VIO_API BOOL vio_call net_vio_get_cfg_cam2imu(HANDLE loginHandle,
vio_mat4x3_s* mat);


/**

* @brief 获取外参 tof to imu

* @param [in] loginHandle：登录句柄

* @param [out] mat：矩阵

* @return 成功返回 1，失败返回错误码

*/

NET_VIO_API BOOL vio_call net_vio_get_cfg_tof2imu(HANDLE loginHandle,
vio_mat4x3_s* mat);


/**

* @brief 获取 OSD 参数

* @param [in] loginHandle：登录句柄

* @param [out] cfg：osd 参数

* @return 成功返回 1，失败返回错误码

*/

NET_VIO_API BOOL vio_call net_vio_get_cfg_osd(HANDLE loginHandle,
vio_osd_cfg_s* cfg);


/**

* @brief 设置 OSD 参数
```

```
 * @param [in] loginHandle：登录句柄

 * @param [in] cfg：osd 参数

 * @return 成功返回 1，失败返回错误码

 */

NET_VIO_API BOOL vio_call net_vio_set_cfg_osd(HANDLE loginHandle,
vio_osd_cfg_s* cfg);


/**

 * @brief 系统重启

 * @param [in] loginHandle：登录句柄

 * @return 成功返回 1，失败返回错误码

 */

NET_VIO_API BOOL vio_call net_vio_system_reboot(HANDLE loginHandle);


#endif
```