Pavo SDK 编程指南(2.0.2)

上海星秒光电科技有限公司
(版权所有,翻版必究)

目 录

1 概述.....5
2 系统要求.....6
3 Pavo体系结构.....7
4 接口说明.....8
4.1 数据结构.....8
4.2 接口说明.....8
4.2.1 pavo_driver.....9
4.2.2 get_scanned_data.....9
4.2.3 is_lidar_connected.....11
4.2.4 get_device_type.....11
4.2.5 get_device_sn.....11
4.2.6 get_device_pn.....12
4.2.7 enable_data.....12
4.2.8 get_dest_ip.....12
4.2.9 set_dest_ip.....12
4.2.10 get_dest_port.....13
4.2.11 set_dest_port.....13
4.2.12 get_lidar_ip.....13
4.2.13 set_lidar_ip.....13
4.2.14 get_lidar_port.....14
4.2.15 set_lidar_port.....14
4.2.16 apply_net_config.....14
4.2.17 get_motor_speed.....14
4.2.18 set_motor_speed.....15
4.2.19 get_merge_coef.....15
4.2.20 set_merge_coef.....15
4.2.21 get_degree_shift.....15
4.2.22 set_degree_shift.....16
4.2.23 get_degree_scope.....16
4.2.24 set_degree_scope.....16
4.2.25 enable_tail_filter.....16
4.2.26 enable_motor.....17
5 快速使用指南.....18
5.1 ROS系统.....18
5.2 Windows系统--PavoView.....18
6 SDK开发流程.....20
6.1 雷达被动上传数据模式流程.....20
6.2 雷达主动上传数据模式流程.....21
7 参考代码.....23
7.1 Scan数据.....23
7.2 PointCloud数据.....23

修订历史

日期	内容	版本	修订人
2018 年 6 月 20 日	1.文档创建	0.1	郭强
2018 年 8 月 15 日	1.增加修改 Lidar IP 地址和端口号的接口 2.增加数据合并接口。	1.0	郭强
2018 年 9 月 27 日	1.增加设置电机转速，角度偏移和有效角度范围的接口。	1.1	郭强
2018 年 10 月 24 日	1.增加获取数据的重载接口，提高灵活性。	1.1.1	郭强
2018 年 12 月 21 日	1.添加了去除点云数据脱尾的功能 2.解决不同编译器不兼容，存在编译不通过的问题	1.1.2	郭强 宋朝帅
2019 年 01 月 05 日	1.增加自定义 Exception 2.删除默认 IP 地址 3.寄存器读写方式修改：设置超时，并回读确认 4.修改 Reset 函数,修复其可能引起挂死的 bug. 5.脱尾处理处理新的版本 0.0.2，相比较上一个版本，这个版本不受距离和点云数据量的限制，适用情况更加广泛，缺点有些点云数据处理不干净	1.3.0	郭强 宋朝帅
2019 年 01 月 31 日	1.修正 Driver 析构时调用 Boost 引发的 BUG	1.4.0	郭强
2019 年 03 月 15 日	1.修改判断点云数据一圈的判定方法，原来是一组数据为单位进行判定，改成以一个单元数据为判定标准	1.5.0	宋朝帅
2019 年 04 月 9 日	1.设置输出的点云数据只含有 45 度-315 度的有效范围内的数据 2.发布脱尾处理 0.0.3 版本，本次版本使用了比较相邻两点距离大于一定值的过滤方法	1.6.0	宋朝帅
2019 年 04 月 18 日	1.由于雷达稳定性提升删除聚类功能，精简代码 2.将点云数据中距离不为 0，强度为 0 的点，距离设置为 0，去除噪点现象	1.7.0	宋朝帅
2019 年 05 月 25 日	1.增加电机使能开关 2.增加获取设备 SN/PN 号功能	1.8.0	郭强
2019 年 06 月 26 日	1.增加去除点云拖尾功能	1.9.0	周俊

2019 年 07 月 26 日	1.在获取的 PCD 数据中增加过滤算法 2.支持雷达被动上传数据模式	2.0.0	宋朝帅 周俊
2019 年 08 月 27 日	1. 修改获取设备 SN 和 PN 号失败的 BUG 2.修改打开设备失败仍返回成功的 BUG 3.修改连接雷达失败仍返回成功的 BUG 4.减小因频繁打开和关闭雷达导致的内存泄露	2.0.1	宋朝帅 周俊
2019 年 11 月 8 日	1. 去除去拖尾算法中和强度相关部分，只使用和角度相关部分 2. 修改资源二次释放带来的 BUG 3. 修改 set_degree_scope 和 get_degree_scope 使用 pavo_driver 命名空间重复的问题	2.0.2	周俊 宋朝帅

1 概述

本文档描述了 SIMINICS Pavo SDK 的功能与使用方法，与 Pavo Lidar 同时发布。

本 SDK 不能用于对其他设备或系统的控制，请勿移作他用。

2 系统要求

本 SDK 为源码发布，请用户自行集成到目标系统。

SDK 使用了 boost 库的 thread 和 asio 模块，在编译前，请准备好 boost 库。

3 Pavo 体系结构

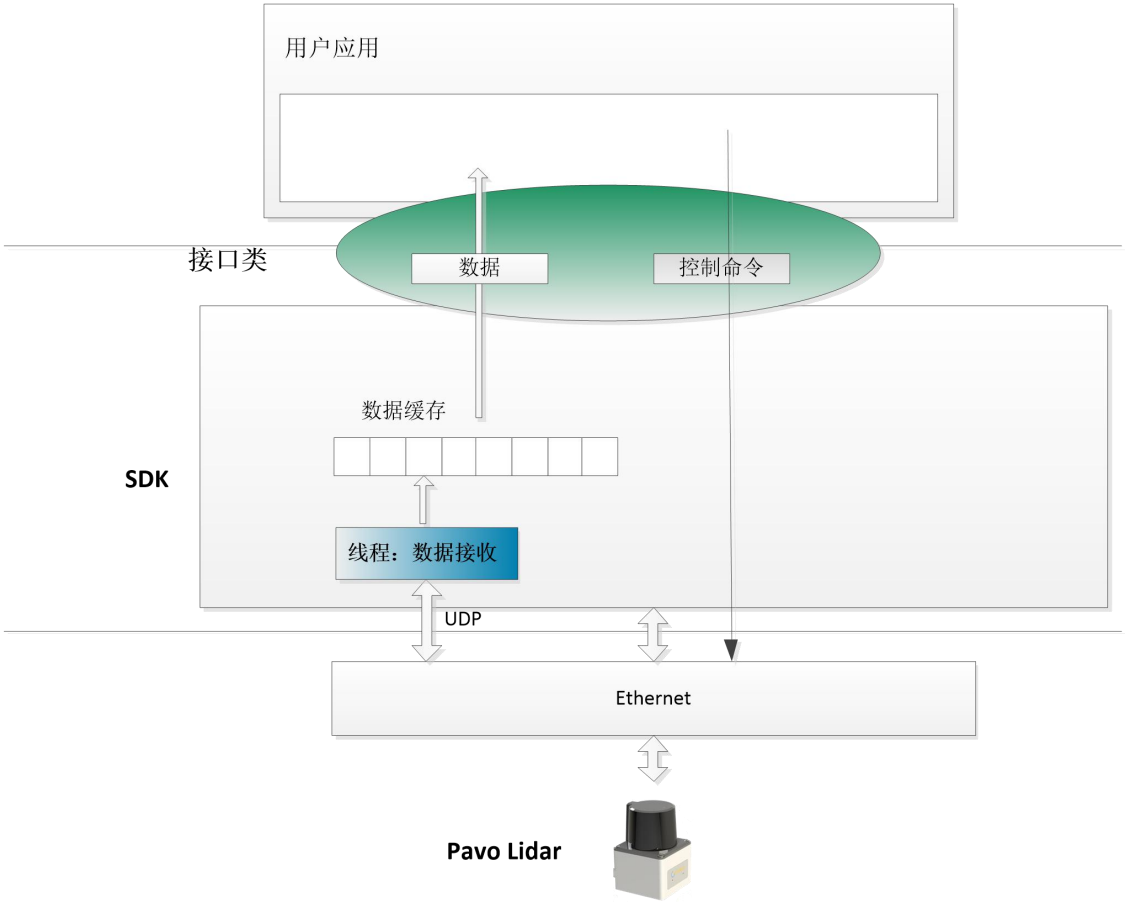


图 1：Pavo 系统基本结构

Pavo 数据采集系统从软件角度包括以下几个部分：

1. **Pavo Lidar**：采集环境数据。
2. **Ethernet**： Pavo Lidar 通过 Ethernet 与上位机连接。Pavo Lidar 采集到的数据经由 UDP 数据包发送到上位机。
3. **SDK**：上位机程序可以通过 SDK 提供的接口接收 Pavo Lidar 采集到的数据或对 Pavo Lidar 进行控制：
   1) 获取设备数据线程：读取设备数据，保证数据传输速度。
   获取雷达数据有两种方式：一种是雷达主动上传数据模式，一种是雷达被动上传数据模式。采用哪种模式获取设备数据，通过生成设备对象的方法进行区分，具体参加接口说明。
   2) 数据缓存：接收到的数据先缓存在内存中。
   用户通过 SDK 接口类获得的数据都是最新一帧的数据，该帧以前的数据如未及时读取，会被丢弃。

4 接口说明

4.1 数据结构

```
typedef struct pavo_response_scan
{
        uint16_t angle;
        uint16_t distance;
        uint8_t  intensity;
} __DATA_ALIGN__ pavo_response_scan_t;

typedef struct pavo_response_pcd
{
        double x;
        double y;
        double z;  //fixed as 0 for single-laser device
        uint8_t  intensity;
} __DATA_ALIGN__ pavo_response_pcd_t;
```

pavo_response_scan_t：角度和距离坐标。

angle：单位为 0.01 度。

distance：单位为 0.002 米。

intensity：值在 0~255 之间，为一相对值。

pavo_response_pcd_t： 笛卡尔坐标。

x,y,z：单位为 0.002 米。

intensity： 值在 0~255 之间，为一相对值。

这两个数据结构定义在 pavo_types.h 头文件中。

4.2 接口说明

用户通过类“class pavo_driver”对 Pavo Lidar 进行访问。该类定义在 pavo_driver.h 头文件中，SDK 的版本号通过 SDK_VER 宏定义进行获取，主要函数有：

```
class pavo_driver
{
public:
        pavo_driver() throw(pavo_exception);
        pavo_driver(std::string dest_ip, uint16_t dest_port) throw(pavo_exception);

        pavo_driver(std::string device_ip, uint16_t device_port, std::string dest_ip, uint16_t dest_port) throw(pavo_exception);

        ~pavo_driver();

        bool pavo_open(std::string device_ip, uint16_t device_port) throw(pavo_exception);
        void pavo_close();

        bool get_scanned_data(pavo_response_scan_t* data_buffer, int& count, int timeout=0);
        bool get_scanned_data(std::vector<pavo_response_scan_t>& vec_buff, int timeout = 0);

        bool get_scanned_data(pavo_response_pcd_t* data_buffer, int& count, int timeout=0);
        bool get_scanned_data(std::vector<pavo_response_pcd_t>& vec_buff, int timeout = 0);

        bool is_lidar_connected();

        int get_device_type();

        bool get_device_sn(uint32_t &sn);
```

```
        bool get_device_pn(uint32_t &pn);

        void enable_data(bool en);

        bool get_dest_ip(std::string& dest_ip);
        bool set_dest_ip(const std::string& dest_ip);

        bool get_dest_port(uint16_t& dest_port);
        bool set_dest_port(uint16_t dest_port);

        bool get_lidar_ip(std::string& lidar_ip);
        bool set_lidar_ip(const std::string& lidar_ip);

        bool get_lidar_port(uint16_t& port);
        bool set_lidar_port(uint16_t port);

        bool apply_net_config();

        bool get_motor_speed(int& motor_spped);
        bool set_motor_speed(int motor_speed);

        bool get_merge_coef(int& merge_coef);
        bool set_merge_coef(int merge_coef);

        bool get_degree_shift(int &degree_shift);
        bool set_degree_shift(int degree_shift);

        void get_degree_scope(int& min, int& max);
        bool set_degree_scope(int min, int max);

        void enable_motor(bool en);

        bool reset();
        bool reset(std::string device_ip, uint16_t device_port, std::string dest_ip, uint16_t dest_port);

        void enable_tail_filter(int method);
        //其后省略.....
    }
```

4.2.1 pavo_driver

```
pavo_driver() throw(pavo_exception);
pavo_driver(std::string dest_ip, uint16_t dest_port) throw(pavo_exception);
```

SDK 核心类，在创建时，会启动一个 UDP 通信节点，用于和 Pavo 通信。

pavo_driver()默认采用雷达被动上传数据模式

pavo_driver(std::string dest_ip, uint16_t dest_port)默认采用主动上传数据模式。如果指定的 IP 地址与运行主机上的 IP 地址不一致，则会抛出 std::runtime_error 异常，用户应捕获该异常，并检查配置。

参数：

dest_ip：输入参数，上位机的 IP 地址。

dest_port：输入参数，上位机接受数据的端口。

4.2.2 get_scanned_data

该接口用于获取 Pavo Lidar 扫描到的数据，根据数据类型的不同，函数重载四次。

```
bool get_scanned_data(pavo_response_scan_t* data_buffer, int& count, int timeout=0);//扫描数据
```

参数：

data_buffer：输出参数，用户提供的数据返回数组。

count： 输入/输出参数，返回的数据个数。

输入：data_buffer 大小；

输出：返回数据个数，如返回数据大于 data_buffer 大小，则表示缓冲大小不足。

timeout：输入参数，超时时间，单位 ms。如果该值为 0，则在获得有效数据之前，该函数会一直阻塞。如果该值大于零，则如果超过该时间还没有获得有效数据，则函数仍然返回，返回值为 false;

返回值：

是否获得有效数据：

true： 获得有效数据。

false：获取数据失败。

```
bool get_scanned_data(std::vector<pavo_response_scan_t>& vec_buff, int timeout=0);//扫描数据
```

参数：

vec_buff：输出参数，返回数据。

timeout：输入参数，超时时间，单位 ms。如果该值为 0，则在获得有效数据之前，该函数会一直阻塞。如果该值大于零，则如果超过该时间还没有获得有效数据，则函数仍然返回，返回值为 false;

返回值：

是否获得有效数据：

true： 获得有效数据。

false：获取数据失败。

```
bool get_scanned_data(pavo_response_pcd_t* data_buffer, int& count, int timeout=0);//点云数据
```

参数：

data_buffer：输出参数，用户提供的数据返回数组。

count：输入/输出参数，返回的数据个数。

输入：data_buffer 大小；

输出：返回数据个数，如返回数据大于 data_buffer 大小，则表示缓冲大小不足。

timeout：输入参数，超时时间，单位 ms。如果该值为 0，则在获得有效数据之前，该函数会一直阻塞。如果该值大于零，则如果超过该时间还没有获得有效数据，则函数仍然返回，返回值为 false;

返回值：

是否获得有效数据：

true： 获得有效数据。

false：获取数据失败。

```
bool get_scanned_data(std::vector<pavo_response_pcd_t>& vec_buff, int timeout=0);//点云数据
```

参数：

Vec_buff：输出参数，返回数据。

timeout：输入参数，超时时间，单位 ms。如果该值为 0，则在获得有效数据之前，该函数会一直阻塞。如果该值大于零，则如果超过该时间还没有获得有效数据，则函数仍然返回，返回值为 false;

返回值：

是否获得有效数据：

true： 获得有效数据。

false：获取数据失败。

4.2.3 is_lidar_connected

判断是否与 Pavo Lidar 成功连接

```
bool is_lidar_connected();
```

参数： 无

返回值：

连接是否成功：

true： 连接成功。

false：连接失败。

4.2.4 get_device_type

获取设备型号

```
int get_device_type();
```

参数： 无

返回值：

Pavo Lidar 型号

4.2.5 get_device_sn

获取设备 SN 序列号

```
bool get_device_sn(uint32_t &sn);
```

参数：

sn：获取到的设备 SN 序列号，以十六进制显示

返回值：

是否成功获取设备 SN 序列号

true： 获取 SN 序列号成功

false:：获取 SN 序列号失败

4.2.6 get_device_pn

获取设备 PN 序列号

```
bool get_device_pn(uint32_t &pn);
```

参数：

pn：获取到的设备 PN 序列号,以十六进制显示

返回值：

是否成功获取设备 PN 序列号

true： 获取 PN 序列号成功

false：获取 PN 序列号失败

4.2.7 enable_data

开启/停止 Pavo Lidar 数据传输

```
void enable_data(bool en);
```

参数：

en：输入参数。true：开启数据传输；false：停止数据传输。

返回值：无。

4.2.8 get_dest_ip

获取 Pavo Lidar 数据的目的地址

```
bool get_dest_ip(std::string& dest_ip);
```

参数：

dest_ip：输出参数。目的 IP 地址。

返回值：

true： 调用成功。

false：调用失败。

4.2.9 set_dest_ip

配置 Pavo Lidar 数据的目的地址

```
bool set_dest_ip(const std::string& dest_ip);
```

参数：

dest_ip：输入参数。目的 IP 地址。

返回值：

true： 调用成功。

false：调用失败。

4.2.10 get_dest_port

获取 Pavo Lidar 数据的目的端口

```
bool get_dest_port(uint16_t& dest_port);
```

参数：

dest_port：输出参数。目的端口。

返回值：

true： 调用成功。

false：调用失败。

4.2.11 set_dest_port

配置 Pavo Lidar 数据的目的端口

```
bool set_dest_port(uint16_t dest_port);
```

参数：

dest_port：输入参数。目的端口。

返回值：

true： 调用成功。

false：调用失败。

4.2.12 get_lidar_ip

获取 Pavo Lidar 数据的源 IP 地址

```
bool get_lidar_ip(std::string& lidar_ip);
```

参数：

lidar_ip：输出参数。Pavo Lidar 地址。

返回值：

true： 调用成功。

false：调用失败。

4.2.13 set_lidar_ip

配置 Pavo Lidar 数据的源 IP 地址

```
bool set_lidar_ip(const std::string& lidar_ip);
```

参数：

lidar_ip：输入参数。Pavo Lidar IP 地址。

返回值：

true： 调用成功。

false：调用失败。

4.2.14 get_lidar_port

获取 Pavo Lidar 数据的源端口

```
bool get_lidar_port(uint16_t& lidar_port);
```

参数：

lidar_port：输出参数。源端口。

返回值：

true： 调用成功。

false：调用失败。

4.2.15 set_lidar_port

配置 Pavo Lidar 数据的源端口

```
bool set_lidar_port(uint16_t lidar_port);
```

参数：

lidar_port：输入参数。源端口。

返回值：

true： 调用成功。

false：调用失败。

4.2.16 apply_net_config

使网络配置生效。在调用 set_dest_ip, set_dest_port, set_lidar_ip, set_lidar_port 之后，须调用 apply_net_config 才能使对 Pavo Lidar 的网络配置生效。

```
bool apply_net_config();
```

参数：

无。

返回值：

true： 调用成功。

false：调用失败。

4.2.17 get_motor_speed

获取电机转速。

```
bool get_motor_speed(int& motor_spped)
```

参数：

motor_speed： 电机转速，单位 Hz。

返回值：

true： 调用成功。

false：调用失败。

4.2.18 set_motor_speed

设置电机转速。

```
bool set_motor_speed(int motor_speed);
```

参数：

motor_speed：电机转速，单位 Hz。

返回值：

true：调用成功。

false：调用失败。

4.2.19 get_merge_coef

获取点云数据合并参数。

```
bool get_merge_coef(int& merge_coef);
```

参数：

merge_coef：点云数据合并参数。

返回值：

true： 调用成功。

false：调用失败。

4.2.20 set_merge_coef

设置点云数据合并参数。

```
bool set_merge_coef();
```

参数：

merge_coef： 点云数据合并参数，可取值 1，2，4，8。

返回值：

true： 调用成功。

false：调用失败。

4.2.21 get_degree_shift

获取点云数据零点偏移大小。

```
bool get_degree_shift(int &degree_shift);
```

参数：

degree_shift：零点偏移大小。

返回值：

true：调用成功。

false：调用失败。

4.2.22 set_degree_shift

配置点云数据零点偏移大小。

```
bool set_degree_shift(int degree_shift);
```

参数：

degree_shift：零点偏移大小， 取值范围[0,35999]，单位：0.01 度

返回值：

true： 调用成功。

false：调用失败。

4.2.23 get_degree_scope

获取点云数据的有效角度范围。

有效角度范围：只有在该范围内的数据才返回给用户。

```
void get_degree_scope(int& min, int& max);
```

参数：

min： 有效角度范围的最小值。

max： 有效角度范围的最大值。

返回值：

无

4.2.24 set_degree_scope

配置点云数据的有效角度范围。

```
bool set_degree_scope(int min, int max);
```

参数：

min： 有效角度范围的最小值。取值范围[0,35999]，单位：0.01 度，且 min<max

max：有效角度范围的最大值。取值范围[0,35999]，单位：0.01 度，且 min<max

返回值：

true： 调用成功。

false：调用失败。

4.2.25 enable_tail_filter

开启点云数据拖尾处理模式

```
void enable_tail_filter(int method);
```

参数：

method：设置使用的去除拖尾的算法。取值范围[0,1,2,3]。0 表示不启动去除拖尾算法，1 开启的去除拖尾算法效果最弱，2 中等，3 最强。

返回值：

无

4.2.26 enable_motor

开启或者禁止雷达电机转动

```
void enable_motor(bool en);
```

参数：

en：设置开启还是关闭雷达电机转动。取值范围[true,false]。默认开启。

返回值：

无

5 快速使用指南

5.1 ROS 系统

1. 编译 Pavo ROS

   catkin_make

2. 启动 LaserScan Demo

   roslaunch pavo_ros pavo_scan_view.launch

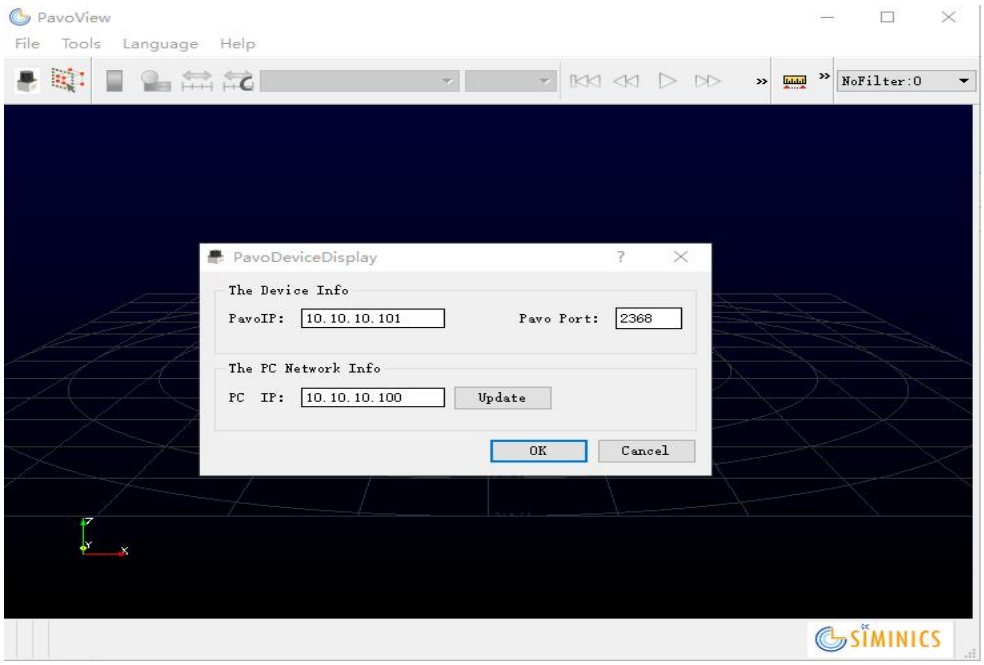
3. 启动 PointCloud Demo

   roslaunch pavo_ros pavo_pcd_view.launch

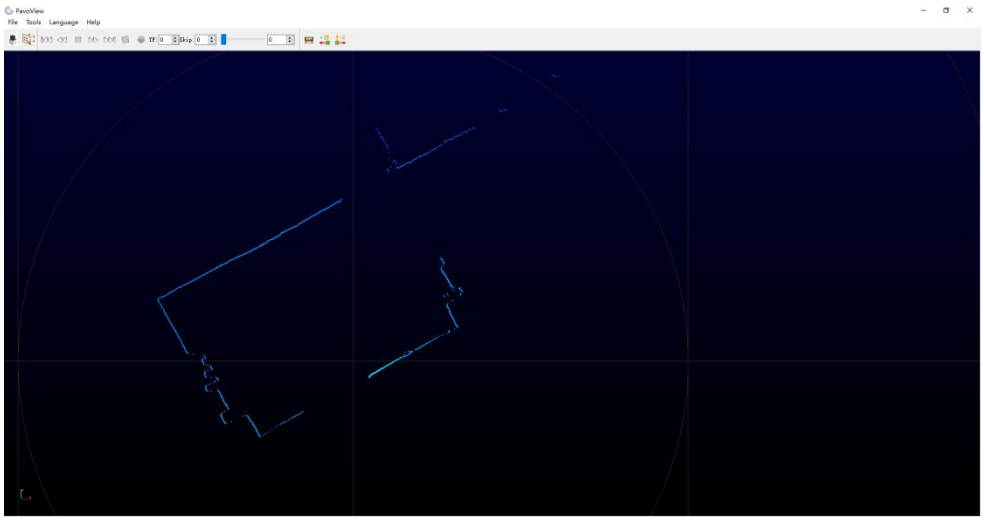
5.2 Windows 系统--PavoView

1. 安装 PavoView

2. 启动 PavoView



3. 效果



6 SDK 开发流程

上位机获取 Pavo 雷达的点云数据有两种模式，一种是雷达被动上传数据模式，一种是雷达主动上传数据模式。

雷达被动上传数据模式，指的是在上位机和雷达能进行网络正常通信的情况下，由上位机根据雷达 IP 打开雷达，直接发送数据请求，从而获取雷达数据。
要点：

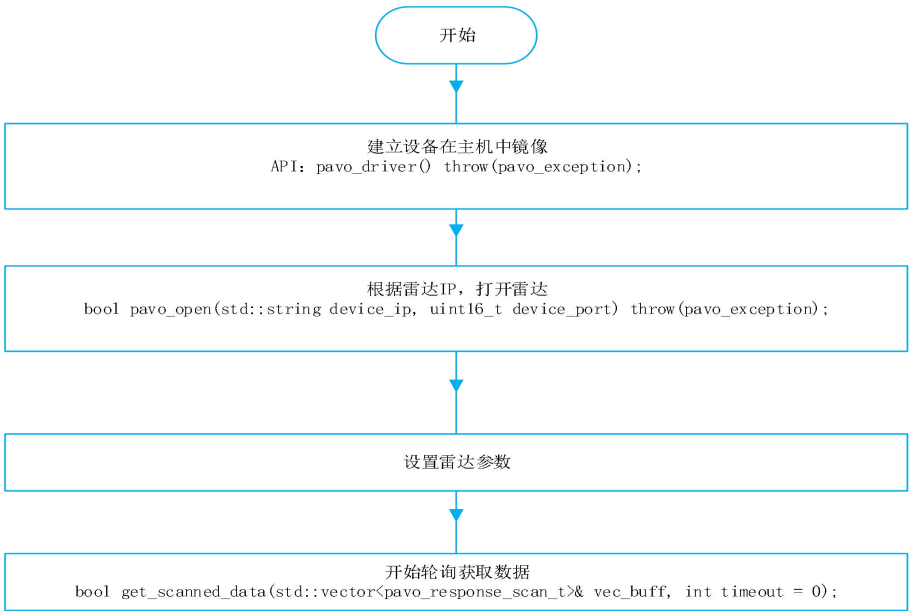
1.知晓雷达配置界面中的 PavoIP、PavoPort 信息
2.保证雷达与上位机能进行正常网络通信

雷达主动上传数据模式，指的是根据雷达中关于 DestIP 和 DestPort 的配置，设置与雷达相连的网口 IP，由上位机根据配置界面中 DestIP 和 DestPort,打开接受数据的端口，根据雷达 IP 打开雷达，从而获取雷达主动上传数据。
要点：

1. 知晓雷达配置界面中的 PavoIP、PavoPort、DestIP、DestPort 信息
2. 设置与雷达相连的网卡 IP 为配置界面中的 DestIP
3. 根据雷达配置界面中的 DestIP 和 DestPort，设置接受数据的端口

雷达上电初始，默认的数据传输方式是主动上传数据模式，此时上位机可以采用雷达主动上传数据模式方案获取数据，也可以采用雷达被动上传数据模式方案获取数据(此时雷达上传数据模式变为被动上传数据模式）。当雷达的数据传输方式是雷达被动上传数据模式时，雷达将只支持此种模式上传数据，不支持雷达主动上传数据模式，除非硬重启雷达。

6.1 雷达被动上传数据模式流程



使用流程：

1. 建立设备在上位机中的映射镜像用以操作对应设备
2. 根据雷达 IP，打开雷达，并请求数据传输
3. 设置雷达参数
4. 轮询获取数据

备注：

当上位机同时集连多个雷达时，多个雷达需要配置不同的 IP，在上位机上建立多个雷达的镜像，用来获取多个雷达数据

6.2 雷达主动上传数据模式流程

在雷达主动上传数据模式流程下，可以通过 Windows 下的 Pavoview 演示软件获取雷达中的 PavoIP、PavoPort、DestIP、DestPort 配置信息,如图 1 所示

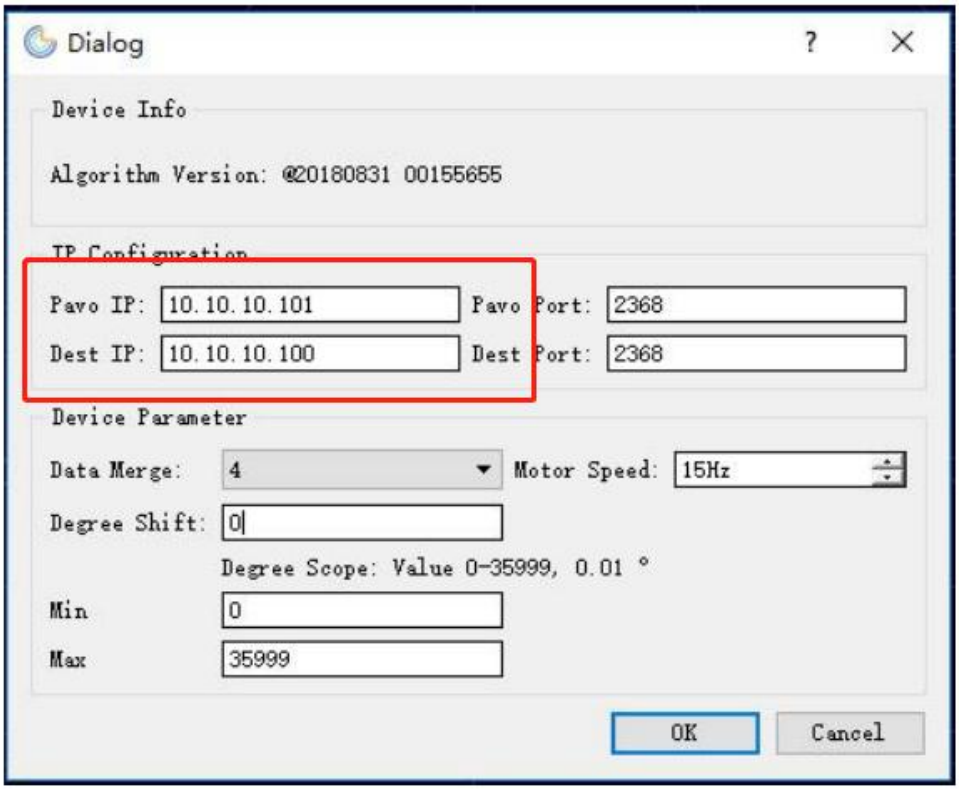
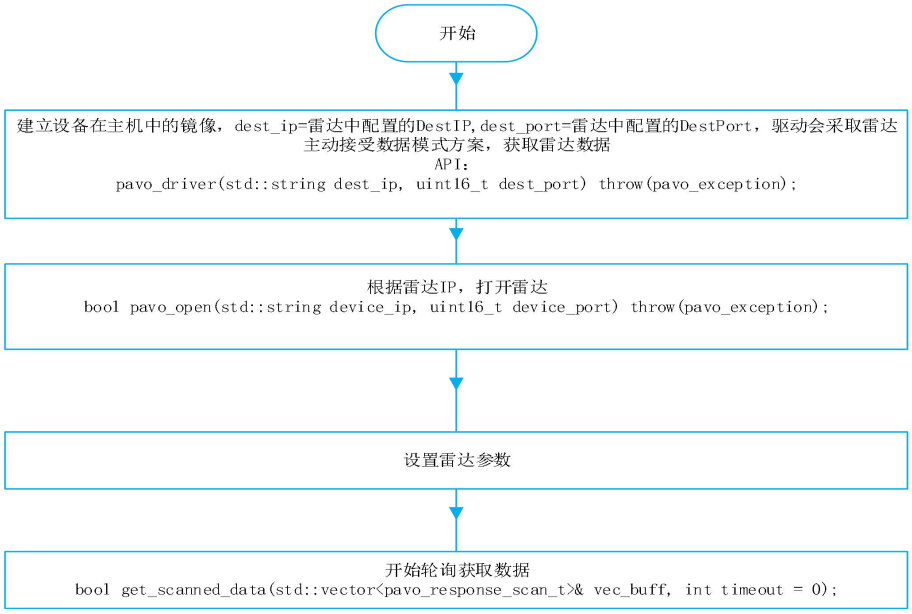


图 1

当需要修改雷达配置信息时，打开配置界面如图 1 所示，修改雷达网络配置，要求 PavoIP 和 DestIP 要在同一网段。

初始雷达默认设置如下：

	IP	Port
Pavo	10.10.10.101	2368
Dest	10.10.10.100	2368



使用流程：

1. 建立设备对象，dest_ip=雷达中配置的 DestIP,dest_port=雷达中配置的 DestPort，由于 passive_mode 默认为 false，驱动采取雷达主动上传数据数据方案，获取雷达数据
2. 根据雷达 IP，打开雷达，并请求数据传输
3. 设置雷达参数
4. 轮询获取数据

7 参考代码

7.1 Scan 数据

请参考 pavo_ros/src/pavo_scan_node.cpp

7.2 PointCloud 数据

请参考 pavo_ros/src/pavo_pcd_node.cpp