先临三维消费扫描 SDK 用户手册

Version 2.5.0.7

先临三维科技股份有限公司

版本历史记录		
版本号	时间	修改记录
V2.5.0.7	2017-09-28	1 手持 SN3D_INIT_DATA 增加 plus 设备类型支持 2 手持标定的 HD 和白平衡标定距离个数减少到 7 个 3 手持扫描增加预扫

目　录

数据结构与宏定义

数据结构

初始化数据

SN3D_INIT_DATA

```
typedef struct tag SN3D_INIT_DATA
{
    char*           device_type;
    wchar_t         config_path[SN3D_MAX_PATH];
    unsigned char   reserved[256];
} SN3D_INIT_DATA, *LPSN3D_INIT_DATA;
```

Members

device_type

设备类型，控制设备的类型，取值 EinScan-Pro 或者 EinScan-Plus。

config_path

配置文件夹，保留参数，暂未使用。

reserved

保留字段。

Remarks

扫描参数

SN3D_SCAN_PARAM

```
typedef struct tag SCAN_PARAM
{
    double      resolution;
    int         flag_texture;
    int         align_mode;
} SN3D_SCAN_PARAM, *LPSN3D_CAN_PARAM;
```

Members

resolution

扫描空间分辨率：快速扫描取值设备为 Pro 时为 0.5-3.0,设备为 Plus 时为 0.7-3.0 ;

HD 扫描 0.3-3.0。

flag_texture

是否为纹理扫描：0 无纹理；1 有纹理。

align_mode

拼接模式，参考拼接模式宏定义

Remarks

图片数据

SN3D_IMAGE_DATA

```
typedef struct tag SN3D_IMAGE_DATA
{
    int             width;
    int             height ;
    int             channel ;
    int             length ;
    unsigned char*  data;
} SN3D_IMAGE_DATA, *LPSN3D_IMAGE_DATA;
```

Members

width

图像宽度；

height

图像高度；

channel

图像通道数；

length

图像数据长度=width*height；

data

图像数据。

Remarks

视频帧数据

SN3D_VIDEO_FRAME

```
typedef struct tag SN3D_VIDEO_FRAME
{
    int                     id ;
    int                     fps ;
    unsigned long long    stamp;
    IMAGE_DATA          video_data;
} SN3D_VIDEO_FRAME, *LPSN3D_VIDEO_FRAME;
```

Members

id

相机索引：0 左相机；1 右相机；2 纹理相机。

fps

帧率，保留参数。

stamp

时间戳，保留参数。

video_data

图像数据。

Remarks

标定状态数据

SN3D_STATE_CALIBRATE

```
typedef struct tagSN3D_STATE_CALIBRATE
{
    int         current_calibrate;
    int         distance_indicate;
    int         current_group ;
    int         snap_state;
    int         compute_state;
} SN3D_STATE_CALIBRATE, *LPSN3D_TATE_CALIBRATE;
```

Members

current_calibrate

取值参考标定类型宏定义。

0： 相机标定，以下为标定过程：

A、平放标定板

B、设备“十字”对准标定板的白色矩形框内

C、上下移动设备，采集五个不同距离图像记为一组（距离见 distance_indicate）

D、移动标定板摆放角度，重复步骤 B-C 四次

E、采集完成，进入相机标定标定计算

1： HD 标定，以下为标定过程

上下移动设备，直到 7 个不同的距离都采集到图像。

2： 纹理标定，又叫白平衡标定，以下为标定过程

上下移动设备，直到采集到距离指示值为 2-4 的图像表示标定结束。

3： 标定结束，程序退出标定。

distance_indicate

标定过程中设备距离标定版的距离指示，值越大距离越远。

不同的标定类型取值范围不一样。

相机双目标定取值： 0-4 每组必须每个值出现一次，表示这一组标定完，总共 5 组；

HD 标定取值： 0-6 必须每个值出现一次；

纹理标定取值： 0-6 当值为 2-4 的时候表示标定成功；

-1 无效数据忽略。

current_group

-1 无效数据忽略；1-5 表示当前双目标定采集的第几组。

snap_state

取值参考采集状态宏定义

-1 无效数据忽略；0 图像采集进行；1 图像采集结束。

compute_state

取值参考计算状态宏定义

-1 无效数据忽略；0 计算中；1 计算成功；2 计算失败。

Remarks

点数据

SN3D_POINT_DATA

```
typedef struct tag SN3D_POINT_DATA
{
    int                 id;
    float               x;
    float               y;
    float               z;
} SN3D_POINT_DATA, *LPSN3D_POINT_DATA;
```

Members

id

数据对象 id，保留参数。

x

x 坐标。

y

y 坐标。

z

z 坐标。

Remarks

点云数据

SN3D_CLOUD_POINT

```
typedef struct tag SN3D_CLOUD_POINT
{
    int                 id;
    int                 vertex_count ;
    POINT_DATA          *vertex_data;
    int                 norma_count ;
    POINT_DATA          *norma_data;
    int                 vertex_color_count ;
    POINT_DATA          * vertex_color _data;
} SN3D_CLOUD_POINT, *LPSN3D_CLOUD_POINT;
```

Members

id

数据对象 id，保留参数。

vertex_count

顶点数据个数。

vertex_data

顶点数据。

norma_count

顶点法向数据个数。

norma_data

顶点法向数据。

vertex_ color _count

顶点颜色数据个数。

vertex_color_data

顶点颜色数据。

Remarks

扫描变换矩阵 RT

SN3D_SCANNER_RT

```
typedef struct tag SN3D_SCANNER_RT
{
    float       rotate[9];
    float       trans[3];
} SN3D_SCANNER_RT, *LPSN3D_SCANNER_RT;
```

Members

rotate

旋转矩阵。

trans

平移矩阵。

Remarks

纹理索引坐标

SN3D_UVCOORD

```
typedef struct tag SN3D_UVCOORD
{
    int                 uu;
    int                 vv;
} SN3D_UVCOORD, *SN3D_UVCOORD;
```

Members

uu

对应纹理图行坐标

vv

对应纹理图列坐标

Remarks

三角面片信息

SN3D_TRI_FACE

```
typedef struct tag SN3D_TRI_FACE
{
    int                 vid[3];
} SN3D_TRI_FACE, *LSN3D_TRI_FACE;
```

Members

vid[3]

三角面片三个顶点索引。

Remarks

网格数据

SN3D_TRI_MESH

```
typedef struct tag SN3D_TRI_MESH
{
    int                 id;
    int                 vertex_count ;
    POINT_DATA          *vertex_data;
    POINT_DATA          *norma_data;
    POINT_DATA          *vertex_color _data;
    int                 face_count ;
    SN3D_TRI_FACE       *face_ids;
    SN3D_UVCOORD        tex_uuvv;
    IMAGE_DATA          tex_data;
}SN3D_TRI_MESH, *LSN3D_TRI_MESH;
```

Members

id

数据对象 id，保留参数。

vertex_count

顶点数据个数。

vertex_data

顶点数据。

norma_data

顶点法向数据，保留参数。

vertex_color_data

顶点颜色数据，保留参数。

vertex_count

三角面片数据个数，保留参数。

face_ids

三角面片三个顶点索引。

tex_uuvv

纹理坐标索引，保留参数。

tex_data

纹理图像数据，保留参数。

网格化处理参数

SN3D_MESH_RPOCESS_PARAM

```
typedef struct tag SN3D_MESH_PROCESS_PARAM
{
    int         mesh_type;
    int         mesh_resolution;
    double      simplification_ratio;
    int         smooth_flag;
    int         sharpen_flag;
    int         mark_point_fill_hole_flag;
} SN3D_MESH_PROCESS_PARAM, *LPSN3D_MESH_PROCESS_PARAM;
```

Members

mesh_type

网格化类型，参考网格化类型定义。

mesh_resolution

网格化精度。

simplification_ratio

简化比例

smooth_flag

平滑标志，0 禁用，1 启用

sharpen_flag

锐化标志，0 禁用，1 启用

mark_point_fill_hole_flag

标志点补洞标志，0 不启动，1 启用

plaint_fill_hole_flag

普通标志标志，0 不启动，1 启用

plaint_fill_hole_perimeter

普通补洞的半径，单位 mm,建议值 10-100

Remarks

补洞相关的参数只有在网格化类型为非封闭类型的时候才有意义，封闭类型的网格化时补洞参数会被忽略

宏定义

初始化类型定义

初始化类型	宏定义值	含义
SN3D_INIT_CALIBRATE	0x00000000	标定初始化
SN3D_INIT_RAPIDSCAN	0x00000001	快速扫描初始化
SN3D_INIT_HD_SCAN	0x00000002	HD 扫描初始化
SN3D_INIT_FIX_SCAN	0x00000003	固定扫描

标定通知事件

宏定义	宏定义值	含义
SN3D_VIDEO_IMAGE_DATA_READY	6	图像数据通知
SN3D_CALIBRATION_STATE_DATA	13	标定状态通知

标定类型常量

宏定义	宏定义值	含义
SN3D_CALIBRATE_STATE_CAMERA	0	相机双目标定
SN3D_CALIBRATE_STATE_HD	1	HD 标定
SN3D_CALIBRATE_STATE_WB	2	纹理相机白平衡标定
SN3D_CALIBRATE_STATE_EXIT	3	标定结束退出

标定采集状态常量

宏定义	宏定义值	含义
SN3D_CALIBRATE_INVALID	-1	无效数据，忽略该数据
SN3D_CALIBRATE_SNAP_STATE_ON	0	图像采集开
SN3D_CALIBRATE_SNAP_STATE_OFF	1	图像采集关

标定计算状态常量

宏定义	宏定义值	含义
SN3D_CALIBRATE_INVALID	-1	无效数据，忽略该数据
SN3D_CALIBRATE_COMPUTING	0	计算处理中
SN3D_CALIBRATE_COMPUTE_SUCCESS	1	标定计算成功
SN3D_CALIBRATE_COMPUTE_FAILED	2	标定计算失败

扫描通知事件宏定义

宏定义	宏定义值	含义
SN3D_VIDEO_IMAGE_DATA_READY	6	图像数据通知

SN3D_DISTANCE_INDECATE	7	扫描仪距离通知
SN3D_SCANNER_RT_READY	8	扫描变换矩阵数据通知
SN3D_CURRENT_MARKPOINT_DATA_READY	9	当前标志点数据通知
SN3D_CURRENT_SCAN_POINT_CLOUD_READY	10	当前扫描点云数据通知
SN3D_WHOLE_SCAN_POINT_CLOUD_READY	11	整个扫描点云数据通知
SN3D_WHOLE_MARKPOINT_DATA_READY	12	全部标志点数据
SN3D_SCANNER_DOUBLECLICK	13	设备双击通知
SN3D_SCANNER_CLICK	14	设备单击通知
SN3D_SCANNER_PLUS	15	+通知
SN3D_SCANNER_SUB	16	-通知
SN3D_START_REQUESTED	20	开始扫描请求
SN3D_START_COMPLETED	21	开始扫描完成
SN3D_PAUSE_REQUESTED	22	暂停扫描请求
SN3D_PAUSE_COMPLETED	23	暂停扫描完成
SN3D_STOP_REQUESTED	24	停止扫描请求
SN3D_STOP_COMPLETED	25	停止扫描完成
SN3D_RESET_REQUESTED	26	重置扫描请求
SN3D_RESET_COMPLETED	27	重置扫描完成
SN3D_MESH_DATA _READY	28	扫描后网格化完成的数据通知
SN3D_PREVIEW_SCAN_POINT_CLOUD_READY	29	预扫当前扫描点云数据通知
SN3D_PREVIEW_SCAN_MARKPOINT_DATA_READY	30	预扫当前标志点数据通知

拼接模式

宏定义	宏定义值	含义
SN3D_ALIGN_MARK_FEATURE	2	特征拼接
SN3D_ALIGN_MODE_MARK_POINT	1	标志点拼接

网格化类型

宏定义	宏定义值	含义
SN3D_MESH_WATERTIGHT	1	封闭模式
SN3D_MESH_UNWATERTIGHT	2	非封闭模式

网格化精度类型

宏定义	宏定义值	含义
SN3D_MESH_HIGHT	1	高精度
SN3D_MESH_MIDDLE	2	中等精度

SN3D_MESH_LOW	3	低精度

返回值定义

宏定义	宏定义值	含义
SN3D_RET_NOERROR	0	没有错误
SN3D_RET_PARAM_ERROR	-1	参数错误
SN3D_RET_ORDER_ERROR	-2	调用顺序错误
SN3D_RET_TIME_OUT_ERROR	-3	调用超时错误
SN3D_RET_NOT_SUPPORT_ERROR	-4	不支持
SN3D_RET_NO_DEVICE_ERROR	-6	没有连接设备
SN3D_RET_DEVICE_LICENSE_ERROR	-7	设备 license 错误
SN3D_RET_GPU_ERROR	-8	显卡不匹配
SN3D_RET_INNER_ERROR	-9	SDK 内部错误
SN3D_RET_NOT_CALIBRATE_ERROR	-10	没有标定
SN3D_RET_LOST_CONFIG_FILE_ERROR	-11	配置文件丢失
SN3D_RET_NO_DATA_ERROR	-12	没有点云数据
SN3D_RET_LOST_CALIBRATE_FILE_ERROR	-13	标定文件丢失

手持扫描接口定义

打开设备句柄

sn3d_initialize

初始化设备句柄

```
void* sn3d_Initialize(
    int          type,
    LPSN3D_INIT_DATA    init_data,
    void* & handle
);
```

Parameters

type

[in]　初始化类型，参考宏定义里面的初始化定义

init_data

[in]　　设备初始化参数 参考 SN3D_INIT_DATA 的定义

handle
[out]　返回设备句柄

Return Values
参考返回值定义宏

Remarks
1 函数的功能为初始化设备句柄。
2 一次只能使用一种类型的设备句柄。
3 必须调用 sn3d_close 释放了一种类型的设备句柄才可以再次调用 sn3d_initialize 初始化另外一种类型的设备句柄。

关闭设备句柄

sn3d_close
关闭设备句柄

```
int sn3d_close(
    void* handle
);
```

Parameters
handle
[in]　设备句柄　sn3d_initialize 返回的设备句柄

Return Values
参考返回值定义宏

Remarks
1 函数的功能为释放设备句柄资源。
2 一次只能使用一种类型的设备句柄。

3 必须调用该函数释放了一种类型的设备句柄才可以再次调用 sn3d_initialize 初始化另外一种类型的设备句柄。
4 标定、扫描处于计算中的时候不能调用该函数关闭句柄

参数配置

获取相机亮度取值范围

sn3d_get_brightness_range

```
int sn3d_get_brightness_range (
void* handle,
int&    min，
int&    max
)
```

Parameters
handle
[in]　设备句柄　sn3d_initialize 的返回值
min
[out]　　相机亮度最小等级，，默认为 0 保留参数，暂未使用
max
[out]　　相机亮度最大等级

Return Values

参考错误码定义

Remarks

设置相机亮度

sn3d_set_ brightness

```
int sn3d_set_brightness (
void* handle ,
int    brightness
)
```

Parameters

handle

[in]　设备句柄　sn3d_initialize 的返回值

brightness

[in]　　相机亮度等级，取值范围由 **sn3d_get_brightness_range** 返回的值决定，值越大越亮。

Return Values

参考错误码定义

Remarks

获取相机当前亮度等级

sn3d_get_ brightness

```
int sn3d_get_ brightness (
        void* handle,
        int& brightness
);
```

Parameters

handle

[in]　　设备句柄　sn3d_initialize 的返回值

brightness

[out]　　相机当前亮度等级，值越大越亮。

Return Values

参考错误码定义

Remarks

设置扫描参数

sn3d_set_scan_param

```
int sn3d_set_scan_param (
        void* handle,
        LPSN3D_SCAN_PARAM param,
);
```

Parameters

handle

[in] 设备句柄 sn3d_initialize 返回的设备句柄

param

[in] 扫描配置参数 **SN3D_SCAN_PARAM** 定义

Return Values

参考错误码定义

Remarks

1 仅在调用 sn3d_initialize 获取设备句柄时候传递的类型为：SN3D_INIT_RAPIDSCAN， SN3D_INIT_HD_SCAN 有效。

2 必须在调用 sn3d_start_scan 前或者 sn3d_abandon_scan 之后调用。

3 调用该函数设置扫描参数会导致先前扫描的数据被清空。

获取扫描参数

sn3d_get_scan_param

```
int sn3d_get_ param (
        void* handle,
        SN3D_SCAN_PARAM& param,
);
```

Parameters

handle

[in] 设备操作句柄 sn3d_initialize 返回的设备句柄

param

[out] 扫描配置参数 **SN3D_SCAN_PARAM** 定义

Return Values

参考错误码定义

Remarks

仅在调用 sn3d_initialize 获取设备句柄时候传递的类型为：

SN3D_INIT_RAPIDSCAN，SN3D_INIT_HD_SCAN 有效。

导入全局框架点

sn3d_import_global_mark_point

```
int sn3d_import_ global_mark_point (
        void* handle,
        SN3D_CLOUD_POINT& mark_point,
);
```

Parameters

handle

[in] 设备操作句柄 sn3d_initialize 返回的设备句柄

mark_point

[in] 标志点，参考 SN3D_CLOUD_POINT 定义

A 仅需传递顶点数据

B 至少 4 个顶点才有效

C 当顶点个数为 0 时表示清空全局框架点

Return Values

参考错误码定义

Remarks

1 仅在 SN3D_INIT_RAPIDSCAN,SN3D_INIT_HD_SCAN 时有效。

2 必须在调用 sn3d_set_scan_param 前或者 sn3d_abandon_scan 之后调用。

3 设置全局框架点后，会一直使用全局框架点数据，直到再次调用该函数将全局框架点个数清零。

PRO 标定

注册标定事件通知的回调函数

sn3d_regist_callback

```
int sn3d_regist_callback(
        void*    handle,
        sn3d_callback    call_back,
        void* user_data
) ;
```

Parameters

handle

[in]设备句柄 sn3d_initialize 的设备句柄

call_back

[in]系统通过此回调函数通知相机采集的状态事件的消息变化，如标定类型、标定状态、标定是否成功。

```
void _sn3d_callback(int,int,void*,int,void*)
```

user_data

[in] call_back 函数的参数，当系统通知用户状态发生变化时将此参数返回给用户

Return Values

参考错误码定义

Remarks

注册此回调函数后，系统将正确的事件从此回调通知注册者，此回调不能与扫描回调同时注册。

标定事件回调响应函数

sn3d_callback

```
void   sn3d_callback (
       void*      handle,
       int        event_type,
       int        event_sub_type,
       void*      data
       int        data_len
       void*      user_data
       );
```

Parameters

handle

[out]设备句柄 与 sn3d_initialize 返回的设备句柄一致。

event_type

[out]事件类型，参考标定事件类型宏定义。

event_sub_type

[out] 事件子类型，暂未使用。

data

[out] 与事件对应的数据结构体指针，见下表

event_type	data	含义
SN3D_VIDEO_IMAGE_DATA_READY	**SN3D_VIDEO_FRAME**	图像数据通知
SN3D_CALIBRATION_STETE__READY	SN3D_STATE_CALIBRATE	标定状态

data_len

[out] data 的长度

user_data

[out] 此返回值为 **sn3d_regist_callback** 设置时传入的回调参数

Return Values

Remarks

1 此函数只有在注册之后才能被调用，并且函数实现代码需要 SDK 调用者自己维护；

2 回调函数在 SDK 内部的工作线程执行，因此外部需要注意多线程处理；

3 回调函数里面不能调用其它 API 函数。

开始采集标定图像

sn3d_start_calibrate

该函数通知 SDK 内部开始当前标定的图像采集及计算

```
int sn3d_start_calibrate (
    void* handle,
);
```

Parameters

handle

[in] 设备操作句柄 sn3d_initialize 返回的设备句柄

Return Values

参考错误码定义

Remarks

1. SDK 内部实现了三种标定：相机标定、HD 标定、纹理相机白平衡标定。
   参考标定类型宏定义
2. 相机标定：

   A、标定初始化后 SDK 内部进入该标定状态，等待执行该函数后开始采集图像；

   B、标定过程需要在标定板的五个不同方位采集数据，每个方位需要采集 5 张不同距离的有效图片；

   C、采集过程中通过回调的方式通知采集相关数据（参考标定状态数据结构定义），一组采集完成 SDK 内部会停止图像采集并通过回调函数通知采集状态的变化，然后等待外部调整好标定板方位，再次调用该函数进入下一组的采集；

   D、五组都采集完后，SDK 内部会自动进入计算标定数据的状态，外部在接收到计算结果通知前不可以再调用该函数和退出标定，需要等待计算完成。

   E、标定计算成功，SDK 内部自动进入 HD 标定状态。

   F、标定计算失败，SDK 内部仍然保留在相机标定状态，等待外部再次调用该函数重新开始采集。

3. HD 标定(plus 设备没有这一步)：

A、相机标定成功或者调用 sn3d_skip_calibrate 跳过标定进入 HD 标定状态，然后调用该函数开始 HD 标定的图像采集。

B、标定过程需要在不同距离采集 7 幅图，采集过程中通过回调的方式通知采集的状态。

C、采集完后进入计算并通过回调函数通知标定计算的结果。

D、标定计算成功，如果有纹理相机 SDK 内部自动进入白平衡标定状态；否则退出标定状态。

E、标定计算失败，SDK 内部仍然保留在 HD 标定状态，等待外部再次调用该函数重新开始采集。

4. 白平衡标定：

A、SDK 内部检测到设备有纹理相机，才会在 HD 标定成功后或者调用 sn3d_skip_calibrate 跳过 HD 标定后进入白平衡标定状态。

B、标定过程中移动设备，直到接收到回调函数状态数据里面的距离值为 2-3 表示标定采集结束。

C、标定计算成功，SDK 内部自动退出标定状态。

D、标定计算失败，SDK 内部仍然保留在白平衡标定状态，等待外部再次调用该函数重新开始采集。

5. 标定退出状态

A、标定结束，所有标定类型都标定成功。

B、该状态下只能调用 sn3d_close 清理设备句柄。

跳过当前标定

sn3d_skip_calibrate

```
int sn3d_skip_calibrate (
    void* handle,
);
```

Parameters

handle

[in] 设备操作句柄 sn3d_initialize 返回的设备句柄

Return Values

参考错误码定义

Remarks

1．只能在相机标定、HD 标定、白平衡标定之间跳转。不能在相机标定的五个组内跳转。

2．比如当前标定为相机标定，调用该函数会进入 HD 标定，如果当前标定为 HD 标定，调用该函数进入白平衡标定，如果当前为白平衡标定，调用后进入相机标定。

手持快速扫描

注册扫描事件通知的回调函数

sn3d_regist_callback

```
int sn3d_regist_callback(
        void*    handle,
        sn3d_callback    call_back,
        void* user_data
) ;
```

Parameters

handle

[in]设备句柄 sn3d_initialize 返回的设备句柄

call_back

[in]系统通过此回调函数通知事件的消息变化

```
void _sn3d_callback(int,int,void*,int,void*)
```

user_data

[in] call_back 函数的参数，当系统通知用户状态发生变化时将此参数返回给用户

Return Values

参考错误码定义

Remarks

1 注册此回调函数后，系统将标定的事件从此回调通知注册者。

2 在调用 start_scan 前必须注册该回调函数，否则得不到时间通知。

扫描事件回调响应函数

sn3d_callback

```
void   sn3d_callback (
void*      handle,
int        event_type,
int        event_sub_type,
void*      data
int         data_len
void*      user_data
);
```

Parameters

handle

[out] 设备句柄 与 sn3d_initialize 返回的设备句柄一致

event_type

[out]事件类型 参考扫描事件类型宏定义

event_sub_type

[out] 事件类型 参考扫描事件的子类型定义，暂未使用

data

[out] 与事件对应的数据结构体指针,具体参考下表

event_type	data	含义
SN3D_VIDEO_IMAGE_DATA_READY	SN3D_VIDEO_FRAME	图像数据通知
SN3D_DISTANCE_INDECATE	int	扫描距离通知： 0 ：太近 1-10：正常，由近及远 11：太远 12：跟踪丢失
SN3D_SCANNER_RT_READY	SN3D_SCANNER_PARAM	扫描变换矩阵数据通知，该 RT 为相对于第一帧的的 RT
SN3D_PREVIEW_SCAN_MARKPOINT_DATA_READY	0	预扫时标志点数据通知： 必须在接收到通知时，在回调函数里面调用 sn3d_get_current_mark_point 获取对应的点云数据
SN3D_PREVIEW_SCAN_POINT_CLOUD_READY	0	预扫时当前点云数据通知： 必须在接收到通知时，在回调函数里面调用

		sn3d_get_current_scan_point_cloud 获取对应的点云数据
SN3D_CURRENT_MARKPOINT_DATA_READY	0	标志点数据通知： 必须在接收到通知时，在回调函数里面调用 sn3d_get_current_mark_point 获取对应的点云数据
SN3D_CURRENT_SCAN_POINT_CLOUD_READY	0	当前点云数据通知： 必须在接收到通知时，在回调函数里面调用 sn3d_get_current_scan_point_cloud 获取对应的点云数据
SN3D_WHOLE_SCAN_POINT_CLOUD_READY	0	整个点云数据通知： 必须在接收到通知时，在回调函数里面调用 sn3d_get_current_scan_whole_point_cloud 获取对应的点云数据
SN3D_WHOLE_MARKPOINT_DATA_READY	0	全部点数据通知： 必须在接收到通知时，在回调函数里面调用 sn3d_get_whole_mark_point 获取对应的点云数据
SN3D_SCANNER_DOUBLECLICK	int	设备双击事件： 0 暂未使用
SN3D_SCANNER_CLICK	int	设备单击事件： 0 当前没有启动扫描，上层接收到通知后可执行 sn3d_start_scan 开启扫描。 1 当前处于扫描中，上层接收到通知后可执行 sn3d_pause_scan 暂停扫描。
SN3D_SCANNER _PLUS	int	设备按键+事件通知： 固定值 1
SN3D_SCANNER _SUB	int	设备按键-事件通知： 固定值-1
SN3D_START_REQUESTED	0	开始扫描请求
SN3D_START_COMPLETED	0	开始扫描完成
SN3D_PAUSE_REQUESTED	0	暂停扫描请求
SN3D_PAUSE_COMPLETED	0	暂停扫描完成
SN3D_STOP_REQUESTED	0	停止扫描请求
SN3D_STOP_COMPLETED	0	停止扫描完成
SN3D_RESET_REQUESTED	0	重置扫描请求
SN3D_RESET_COMPLETED	0	重置扫描完成

data_len

[out] data 的长度

user_data

[out]此返回值为 **sn3d_regist_callback** 设置时传入的回调参数

Return Values

参考错误码定义

Remarks

1．此函数只有在注册之后才能被调用，并且函数实现代码需要 SDK 调用者自己维护；

2．回调函数在 SDK 内部的工作线程执行，因此外部需要注意多线程处理；

3．回调函数里面除了获取数据的特定函数 **sn3d_get_current_scan_point_cloud**，**sn3d_get_current_scan_whole_point_cloud**,**sn3d_get_current_mark_point**,**sn3d_get_whole_mark_point**，不能调用其它 API 函数

开始扫描

sn3d_start_scan

```
int sn3d_start_scan(
    void* handle
);
```

Parameters

handle

[in]设备句柄 sn3d_initialize 返回的设备句柄。

Return Values

错误代码　参考返回值宏定义

Remarks

1 此函数只有在注册回调函数之后才能被调用

2 每次调用 sn3d_set_scan_param 后的第一次调用 sn3d_start_scan 开始预扫，预扫描时候只有单片数据和当前片的标志点数据。

第二次调用 sn3d_start_scan 才开始扫描数据。

3 扫描过程中会通过回调的方式向用户传递扫描信息：包括扫描远近提示、扫描数据等信息，sn3d_get_current_scan_whole_point_cloud 和 sn3d_get_current_scan_point_cloud 获取的数据（中间过程数据）的仅供显示使用，通过 sn3d_get_whole_mark_point 和 sn3d_get_current_mark_point 获取标志点信息。详见 sn3d_callback

4 可以调用 sn3d_finish_scan 停止扫描

5 可以调用 sn3d_pause_scan 暂停扫描

6 可以调用 sn3d_abandon_scan 放弃扫描数据

7 禁止调用 sn3d_set_scan_param

停止当前扫描

sn3d_finish_scan

```
int sn3d_finish_scan(
    void*    handle
);
```

Parameters

handle

[in]设备句柄 sn3d_initialize 的返回值。

Return Values

错误代码 参考返回值的宏定义

Remarks

1 可以通过 sn3d_get_current_scan_whole_point_cloud，获取融合后精度较高的最终数据

2 可以调用 sn3d_start_scan 在已有数据的基础上继续扫描

3 可以调用 sn3d_abandon_scan 放弃扫描数据

4 禁止调用 sn3d_set_scan_param

暂停当前扫描

sn3d_pause_scan

```
int sn3d_pause_scan(
    void*    handle
);
```

Parameters

handle

[in]设备句柄　sn3d_initialize 的返回值。

Return Values

错误代码 参考返回值的宏定义

Remarks

1 可以调用 sn3d_start_scan 继续扫描

2 可以调用 sn3d_finish_scan 停止扫描

3 可以调用 sn3d_abandon_scan 放弃当前扫描

4 禁止调用 sn3d_set_scan_param

重置当前扫描数据

sn3d_abandon_scan

```
int sn3d_abandon_scan(
    void*    handle
);
```

Parameters

handle

[in]设备句柄　sn3d_initialize 的返回值。

Return Values

错误代码 参考返回值的宏定义。

Remarks

1 当前扫描的数据会被全部清空，谨慎使用

2 可以调用 sn3d_set_scan_param 可重新设置扫描参数

3 可以调用 sn3d_start_scan 继续扫描

获取当前扫描点云

sn3d_get_current_scan_point_cloud

```
int sn3d_get_current_scan_point_cloud (
    void* handle,
    SN3D_CLOUD_POINT& point_cloud
);
```

Parameters

handle

[in] 设备句柄 sn3d_initialize 返回的设备句柄。

point_cloud

[in,out] 点云数据。

Return Values

错误代码 参考返回值宏定义

Remarks

1 该函数只有在回调函数里面接收到通知 SN3D_CURRENT_SCAN_POINT_CLOUD_READY 时候，在回调函数返回前立刻调用有效。

2 该函数需要调用两次，第一次调用时候 point_cloud 指针域保持空，函数返回点数，外部根据点数分配内存，指针域不为空再调用一次，函数返回点云数据

3 该函数所获取的数据仅中间过程数据

获取当前扫描融合后的整个点云

sn3d_get_current_scan_whole_point_cloud

```
int sn3d_get_current_scan_whole_point_cloud(
    void* handle,
```

```
    SN3D_CLOUD_POINT& point_cloud
);
```

Parameters

handle

[in] 设备句柄 sn3d_initialize 返回的设备句柄。

point_cloud

[in,out] 点云数据。

Return Values

错误代码 参考返回值宏定义

Remarks

1 该函数只有在回调函数里面接收到通知 SN3D_WHOLE_SCAN_POINT_CLOUD_READY 时候，在回调函数返回前立刻调用有效。

2 该函数需要调用两次，第一次调用时候 point_cloud 指针域保持空，函数返回点数，外部根据点数分配内存，指针域不为空再调用一次，函数返回点云数据

3 sn3d_finish_scan 被调用后，该函数所获取的数据为最终扫描的数据，如果 sn3d_finish_scan 未被调用则为中间过程数据

获取当前扫描标志点

sn3d_get_current_mark_point

```
int sn3d_get_current_mark_point(
    void* handle,
    SN3D_CLOUD_POINT& point_cloud
);
```

Parameters

handle

[in] 设备句柄 sn3d_initialize 返回的设备句柄。

point_cloud

[in,out] 点云数据。

Return Values

错误代码　参考返回值宏定义

Remarks

1．该函数只有在回调函数里面接收到通知 SN3D_CURRENT_MARKPOINT_DATA_READY 时候，在回调函数返回前立刻调用有效。

2．该函数需要调用两次，第一次调用时候 point_cloud 指针域保持空，函数返回点数，外部根据点数分配内存，指针域不为空再调用一次，函数返回点云数据

获取全部标志点

sn3d_get_whole_mark_point

```
int sn3d_get_whole_mark_point(
    void* handle,
    SN3D_CLOUD_POINT& point_cloud
);
```

Parameters

handle

[in] 设备句柄 sn3d_initialize 返回的设备句柄。

point_cloud

[in,out]　点云数据。

Return Values

错误代码　参考返回值宏定义

Remarks

1 该函数只有在回调函数里面接收到通知 SN3D_WHOLE_MARKPOINT_DATA_READY 时候，在回调函数返回前立刻调用有效。

2 该函数需要调用两次，第一次调用时候 point_cloud 指针域保持空，函数返回点数，外部根据点数分配内存，指针域不为空再调用一次，函数返回点云数据

删除部分扫描数据

sn3d_update_cloud_point

```
int sn3d_update_cloud_point(
        void*      handle
        SN3D_CLOUD_POINT& point_cloud
);
```

Parameters

handle

[in]设备句柄　sn3d_initialize 的返回值。

point_cloud

[in] 外部编辑当前帧后，剩下的点云数据。

Return Values

错误代码 参考返回值的宏定义

Remarks

1 通过该函数删除的点数据是不可以撤销的，如果要实现点云的数据撤销重做功能需要外部自己实现，将最终编辑后保留下来的点云数据传入给 SDK。

2 该函数只有在扫描处过程处于**暂停**状态（sn3d_pause_scan）时候可以使用。

3．不支持删除所有点，删除所有点须调用 sn3d_abandon_scan。

手持 HD 扫描

同手持快速扫描

网格数据处理接口定义

生成网格数据

sn3d_mesh_process

```
int sn3d_mesh_process (
void* handle,
SN3D_MESH_PROCESS_PARAM& param
)
```

Parameters

handle

[in] 设备句柄 sn3d_initialize 的返回值

param

[in] 数据处理参数 参考 **SN3D_MESH_PROCESS_PARAM 定义**

Return Values

参考错误码定义

Remarks

1 必须在调用 **sn3d_finish_scan** 停止扫描结束后调用该函数才有效

2 该函数为异步调用

3 数据处理后的结果通过回调接口通知，通知类型为 SN3D_MESH_DATA _READY

4 外部在回调接口里面调用函数 sn3d_get_current_mesh_data 获取数据

获取当前网格数据

sn3d_get_current_mesh_data

```
int sn3d_get_current_mesh_data(
    void* handle,
    SN3D_CLOUD_POINT& point_cloud
);
```

Parameters

handle

[in] 设备句柄 sn3d_initialize 返回的设备句柄。

point_cloud

[in,out] 点云数据。

Return Values

错误代码　参考返回值宏定义

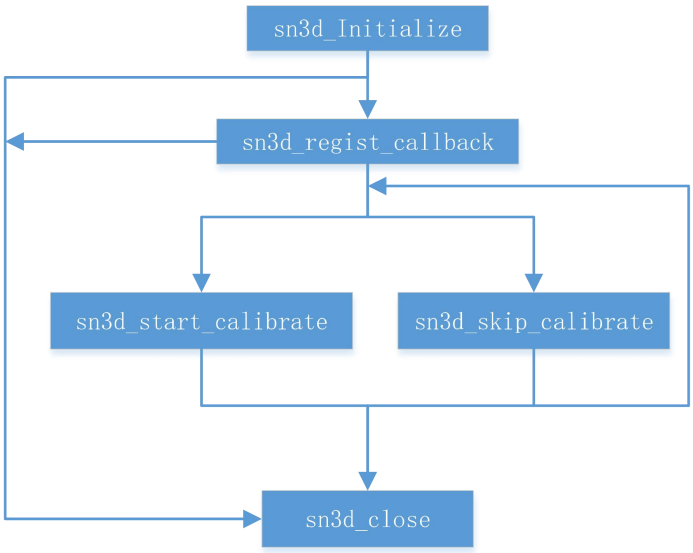
Remarks

1 该函数只有在回调函数里面接收到通知 SN3D_MESH_DATA _READY 时候，在回调函数返回前立刻调用有效。

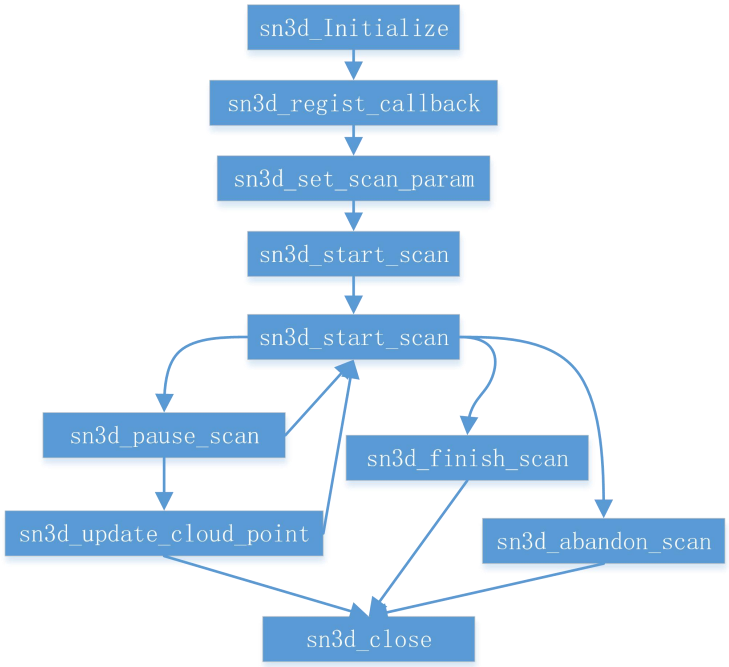
2 该函数需要调用两次，第一次调用时候 point_cloud 指针域保持空，函数返回点数，外部根据点数分配内存，指针域不为空再调用一次，函数返回点云数据

使用流程

标定流程



HD 扫描和快速扫描流程



固定扫描接口定义

打开设备句柄

sn3d_Initialize_Scan_Session

初始化设备句柄

```
int    sn3d_Initialize_Scan_Session(
       SCANTYPE scantype,
       const TCHAR* szIniPath,
       void*&hScan,
       int unitSize = 1);
```

Parameters

scantype

[in]　初始化类型：　scantype = BINOCULARS

szIniPath

[in]　初始化设备路径（res 文件夹所在的路径）

hScan

[out]　　返回设备句柄

unitSize

[in]　　预留参数

Return Values

0 初始化成功，其他值失败

Remarks

1 函数的功能为初始化设备句柄。

2 一次只能使用一种类型的设备句柄。

3. 如果已经获取了设备句柄信息，可以调用 **sn3d_Destroy_Scan_Session** 来释放已经获取的句柄，然后继续再调用 sn3d_Initialize_Scan_Session 获取句柄。

关闭设备句柄

sn3d_Destroy_Scan_Session

关闭设备句柄.

```
int sn3d_Destroy_Scan_Session(
    void* hScan
);
```

Parameters

hScan

[in] 设备句柄，sn3d_Initialize_Scan_Session 返回的设备句柄。

Return Values

0 成功，其他值失败

Remarks

1 函数的功能为释放设备句柄资源。

2 一次只能使用一种类型的设备句柄。

3 必须调用该函数释放一种类型的设备句柄才可以再次调用 sn3d_Initialize_Scan_Session.初始化另外一种类型的设备句柄。

注册相机视频显示回调函数

Sn3d_Regist_Fix_Scan_Camera_Callback

```
int sn3d_Set_Fix_Scan_Camera_Callback(
HSCAN hScan,
CAMERA_DISPLAY_CALLBACK_FUNCTION fun_,
void* pOwner)
```

Parameters

hScan

[in] 设备句柄，sn3d_Initialize_Scan_Session 返回的设备句柄。

call_back

[in] 系统通过此回调函数通知视频图像变化

pOwner

[in] call_back 函数的参数，当系统通知用户状态发生变化时将此参数返回给用户

Return Values

0 成功，其他值失败

Remarks

1 注册此回调函数后，系统将视频图像从此回调通知注册者。

视频图像回调相应函数

CAMERA_DISPLAY_CALLBACK_FUNCTION

```
void(*CAMERA_DISPLAY_CALLBACK_FUNCTION)(
const void* const& pOwner,
const int& cameraID,
const unsigned char* const& pImage,
const int& width,
const int& height,
const int& channel
);
```

Parameters

pOwner

[out] 此返回值为 Sn3d_Regist_Fix_Scan_Camera_Callback 设置时传入的回调参数

cameraID

[out] 相机 id. 相机 id 参见 : SN3D_VIDEO_FRAME

pImage

[out] 图像数据

width

[out] 图像宽度

height

[out] 图像高度

channel

[out] 图像通道数

Return Values

Remarks

1．此函数只有在注册之后才能被调用，并且函数实现代码需要 SDK 调用者自己维护；

2．回调函数在 SDK 内部的工作线程执行，因此外部需要注意多线程处理；

3. 回调函数里面不能调用其它 API 函数。

获取设备亮度最大等级

sn3d_get_brightness_max_range

```
int sn3d_get_brightness_max_range (
void* hScan,
int&    max
)
```

Parameters

hScan

[in]　设备句柄，sn3d_Initialize_Scan_Session 返回的设备句柄。

max

[out]　设备亮度等级的最大值.

Return Values

0 成功，其他值失败

.

Remarks

设置设备亮度值

sn3d_set_ brightness

```
int sn3d_set_brightness (
void* handle ,
int    brightness
)
```

Parameters

hScan

[in]　设备句柄，sn3d_Initialize_Scan_Session 返回的设备句柄。

brightness

[in]　设置设备等级值，范围：0-12.

Return Values

0 成功，其他值失败

Remarks

设置纹理扫描参数

sn3d_enable_textrue

```
int sn3d_enable_textrue (
  HSCAN hScan,
  int bEnable
);
```

Parameters

hScan

[in]　设备句柄，sn3d_Initialize_Scan_Session 返回的设备句柄。

bEnable

[in]　1 纹理扫描　0　非纹理扫描

Return Values

0 成功，其他值失败

Remarks

1 仅仅在 sn3d_Initialize_Scan_Session 初始化完成调用

导入全局框架点

sn3d_import_global_mark_point

```
int sn3d_import_ global_mark_point (
      HSCAN hScan,
      sn3DTargetAlign::RefPoints,
);
```

Parameters

hScan

[in]　设备句柄，sn3d_Initialize_Scan_Session 返回的设备句柄。

RefPoints

[in]　标志点，至少 4 个点以上

Return Values

0 成功,其他值失败

Remarks

1. 必须在转台编码点或者自由扫描前调用
2 一旦导入全局框架点，扫描将一直与导入的标志点进行拼接
3 全局框架点的使用参见 **sdk_demo_fix**

转台编码点扫描

sn3d_turntable_code_Scan

```
sn3d_turntable_code_Scan(
    HSCAN hScan,
    sn3DCore::sn3DRangeData*& data,
    bool isfirst_model
    );
```

Parameters

hScan

[in]　设备句柄，sn3d_Initialize_Scan_Session 返回的设备句柄。

data

[out]　扫描数据

isfirst_model

[in]　转台扫描一组的起始位置

Return Values

0 成功,其他值失败

Remarks

1 sn3DCore::sn3DRangeData 的使用参见:**sdk_demo_fix**

2 Example:

```
int turntable_times_ = 8;
int turntable_angle_ = 360 /8 + 0.5;
//group 1
for (int turn_i_ = 0; turn_i_ < turntable_times_; ++turn_i_)
{
  if(0= =turn_i_ )
      sn3d_turntable_code_Scan(HSCAN, sn3DCore::sn3DRangeData*& ,true);
else
    sn3d_turntable_code_Scan(HSCAN, sn3DCore::sn3DRangeData*& ,false);
    if( sn3DTurnTable::GetTurnTableState())
       sn3DTurnTable::Run2Coordinate(X_ROTATE, turntable_angle_, RELATIVE_ANGLE);
}

//group 2
for (int turn_i_ = 0; turn_i_ < turntable_times_; ++turn_i_)
{
  if(0= =turn_i_ )
      sn3d_turntable_code_Scan(HSCAN, sn3DCore::sn3DRangeData*& ,true);
else
```

```
    sn3d_turntable_code_Scan(HSCAN, sn3DCore::sn3DRangeData*& ,false);
    if( sn3DTurnTable::GetTurnTableState())
        sn3DTurnTable::Run2Coordinate(X_ROTATE, turntable_angle_, RELATIVE_ANGLE);
}
```

转台标志点扫描

sn3d_turntable_mark_Scan

```
sn3d_turntable_mark_Scan(
HSCAN hScan,
 sn3DCore::sn3DRangeData*& data,
std::vector<sn3DCore::sn3DRangeData*> datas,
bool isfirst_model);
```

Parameters

hScan

[in]　设备句柄，sn3d_Initialize_Scan_Session 返回的设备句柄。

data

[out]　扫描数据

Datas

[in]　从 (true ==**isfirst_model)** 开始到扫描结束的数据.

isfirst_model

[in]　转台扫描一组的起始位置

Return Values

0 成功,其他值失败

Remarks

1 sn3DCore::sn3DRangeData. Reference:**sdk_demo_fix**

2 Example:multiple group scan

```
std::vector<sn3DCore::sn3DRangeData*> all_datas
std::vector<sn3DCore::sn3DRangeData*> group_datas

int turntable_times_ = 8;
int turntable_angle_ = 360 /8 + 0.5;
//group 1
for (int turn_i_ = 0; turn_i_ < turntable_times_; ++turn_i_)
{
sn3DCore::sn3DRangeData* data;
   if(0= =turn_i_ )
       sn3d_turntable_mark_Scan(HSCAN, sn3DCore::sn3DRangeData*& ,true);
```

```
else
    sn3d_turntable_code_Scan(HSCAN, data ,group_datas, false);
    if( sn3DTurnTable::GetTurnTableState())
        sn3DTurnTable::Run2Coordinate(X_ROTATE, turntable_angle_, RELATIVE_ANGLE);

all_datas.push_back(data);
group_datas.push_back(data);
}

//group 2
group_datas.clear();
for (int turn_i_ = 0; turn_i_ < turntable_times_; ++turn_i_)
{
sn3DCore::sn3DRangeData* data;
  if(0= =turn_i_ )
      sn3d_turntable_mark_Scan(HSCAN, sn3DCore::sn3DRangeData*& ,true);
else
    sn3d_turntable_code_Scan(HSCAN, data ,group_datas, false);
    if( sn3DTurnTable::GetTurnTableState())
        sn3DTurnTable::Run2Coordinate(X_ROTATE, turntable_angle_, RELATIVE_ANGLE);

all_datas.push_back(data);
group_datas.push_back(data);
}
```

自由扫描

sn3d_free_Scan

```
sn3d_free_Scan(
HSCAN hScan,
std::vector<sn3DCore::sn3DRangeData*> datas,
sn3DCore::sn3DRangeData*& data);
```

Parameters

hScan

[in]　设备句柄，sn3d_Initialize_Scan_Session 返回的设备句柄。

data

[out]　扫描数据

Datas

[in]　所有扫描模型

Return Values

0 成功，其他值失败

Remarks

1 sn3DCore::sn3DRangeData 的使用参见:**sdk_demo_fix**

打开转台设备

SetType

打开转台第一步：设置类型.

```
int    SetType(
       char *idType);
```

Parameters

[in]　转台类型，idType = "COM9-115200-1100"

Return Values

0 成功，其他值失败

Remarks

1.打开转台分为 3 步 : **SetType, OpenTurntable 和 initTurntable.**

2.**Example:** 打开转台

```
int set_ = -1;
char *motortype = "COM9-115200-1100";
set_ = sn3DTurnTable::SetType(motortype);
if(0==set)
{
        if(0==sn3DTurnTable::OpenTurntable())
                set_ = sn3DTurnTable::initTurntable();
 }
 return set;
```

OpenTurntable

打开转台设备第二步：打开转台.

```
int OpenTurntable();
```

Return Values

0 成功，其他值失败

Remarks

1.打开转台分为 3 步 : **SetType, OpenTurntable 和 initTurntable.**

initTurntable

打开转台第三步：初始化转台

```
int initTurntable();
```

Return Values

0 成功，其他值失败

Remarks

1.打开转台分为 3 步 : **SetType, OpenTurntable 和 initTurntable。**

关闭转台设备

CloseTurntable

关闭转台设备

```
int CloseTurntable (
);
```

Return Values

0 成功，其他值失败

Remarks

转动指定角度

Run2Coordinate

```
int Run2Coordinate(
    UCHAR degreefreedomID,
```

```
    float desangle,
    bool state=RELATIVE_ANGLE
);
```

Parameters

degreefreedomID

[in]　自由度，目前仅仅支持 degreefreedomID = **X_ROTATE**

desangle

[in]　转动的角度

state

[in]　　转动方式目前仅仅支持　state = RELATIVE_ANGLE

Return Values

0 成功，其他值失败

Remark

1.Example: 转动 45 度

```
int turntable_angle_ = 45;
sn3DTurnTable::Run2Coordinate(X_ROTATE, turntable_angle_, RELATIVE_ANGLE);
```

查询转台的状态

GetTurnTableState

查询当前转台的状态.

```
int GetTurnTableState();
```

Parameters

Return Values

1：可以继续进行扫描.

0: 不能继续扫描，等待转台转动结束

Remarks

1.Example: 检查转台是否可以转动，并且转动 45 度

```
int turntable_angle_ = 45;
if( sn3DTurnTable::GetTurnTableState())
    sn3DTurnTable::Run2Coordinate(X_ROTATE, turntable_angle_, RELATIVE_ANGLE);
```

使用流程