AIeveR
Robotics Limited
雲芯機器人有限公司

AirInsight - A 系列 工业级面阵 3D 相机通信库

参考手册

V1.0.0

修订日期：2025.05

AIeveR Robotics Limited

软件使用许可协议

使用本软件前，客户需同意下述软件使用许可协议 （以下简称“本协议”）的内容。 客户使用或复制 AirInsight - A 系列 工业级面阵 3D 相机通信库（以下简称“本软件”）的部分或全部功能时，将视为客户已同意本协议中规定的所有内容，且本协议成立。

第 1 条 （使用权的授予）

在客户遵守本协议内容的前提下，云芯未来机器人（深圳）有限公司（以下简称“本公司”）将对客户授予本软件的非独占性使用权。

第 2 条 （禁止事项）

客户不可对本软件实施以下操作或改动。

a. 更改本软件现有功能或向本软件添加新功能。

b. 对本软件实施的所有逆向工程行为，如：反编译、反汇编等。

c. 向第三方二次销售、转让、二次分配、授予、租赁本软件及本公司提供的本软件授权密钥等。但允许客户将本软件与使用本软件编写的应用程序共同进行二次分配。

第 3 条 （著作权等）

与本软件及本软件手册相关的所有知识产权 （如著作权），归本公司所有。

第 4 条 （免责）

客户或第三方因使用本软件而遭受的所有损害，本公司概不负责！

第 5 条 （支持）

本公司将基于本协议，根据客户针对本软件提出的问题，提供技术支持。但并不保证本公司提供的技术支持服务可使客户达成期望目的。

第 6 条 （协议终止）

1. 当客户进行废弃本软件及其复制品等以停止使用本软件时，本协议自动终止。

2. 当客户违反本协议中规定的任一条款时，本公司可单方面解除本协议。同时，客户应立即废弃本软件及其复制品，或将之返还至本公司。

3. 因客户违反本协议，而使本公司蒙受损失时，客户应向本公司赔偿相关损失。

第 7 条 （准据法）

本协议遵从中华人民共和国法律。

前言

使用前请务必阅读本用户手册。

阅读后，请妥善保管，以便日后查阅。

AirInsight -A 系列工业级面阵 3D 相机通信库，为客户提供通过应用程序控制 AirInsight - A 系列工业级面阵 3D 相机的通信接口。具体使用方法，请向本公司索取示例程序。

本手册的内容是本着准确无误的目标进行编制的。但是如果发现有不清楚、错误或含糊的内容，请联系本公司销售部们。如有缺页或装订错误，本公司予以更换。

联系我们

云芯未来机器人（深圳）有限公司

电子邮件：linhong@aiever-robotics.com
地 址：深圳市福田区 ISPSZHK 深港国际科技园 E 栋 6 楼 608 室
邮 编：518000

感谢选用云芯机器人的机器视觉相关产品

为回报客户，我们将以一流的机器视觉产品、完善的售后服务、高效的技术支持，帮助您建立自己的机器视觉系统。

云芯机器人的更多信息

在我们的网页上可以获得更多关于公司 和产品的信息，包括：公司简介、产品介绍、技术支持、产品最新发布等。您也可以通过邮件（linhong@aiever-robotics.com）咨询关于公司和产品的更多信息。

技术支持和售后服务

您可以通过以下途径获得我们的技术支持和售后服务：

电子邮件：linhong@aiever-robotics.com

地 址：深圳市福田区 ISPSZHK 深港国际科技园 E 栋 6 楼 608 室

邮 编：518000

用户手册的用途

用户通过阅读本手册，能够熟悉机器视觉传感产品 A 系列工业级面阵 3D 相机通信库的基本使用，能够在实际工程项目中进行开发。

用户手册的使用对象

本用户手册适用于，对软件开发调试操作有一定了解的工程人员。

用户手册的主要内容

本手册由 6 节内容组成。详细介绍了 A 系列工业级面阵 3D 相机通信库的运行环境、文件构成、使用示例、数据结构定义、类详细说明、开发例程等。

相关文件

关于 A 系列工业级面阵 3D 相机的硬件安装，请参阅随本产品配套的《AirInsight - A 系列工业级面阵 3D 相机硬件用户手册》。关于 A 系列工业级面阵 3D 相机的软件调试，请参阅随本产品配套的《AirInsight - A 系列工业级面阵 3D 相机软件使用手册》

文档修订版本

编号	版本号	修订日期
	1.0.0	2025.06.09

目录

1. 运行环境

操作系统	Windows 10 64bit 系统 Windows 11 64bit 系统
CPU	Intel I5 及以上
GPU	带有独显的电脑最佳
内存容量	16GB 以上
硬盘空间	16GB 以上
通信接口	Ethernet 1000BASE

1.1 防火墙设置

使用 SDK 生成的工程，在运行可执行文件时需要勾选“专用网络”和“公用网络”选项。

1.2 接口测试电脑配置及环境

测试电脑 1

电脑型号	Lenovo 小新 90X0005RCP
操作系统	Windows 11 64bit 系统
CPU	Intel I5-14400F 2.50GHz
内存容量	32GB
硬盘空间	1TB
通信接口	Ethernet 1000BASE

测试电脑 2

电脑型号	Dell MS-7A93
操作系统	Windows 10 Pro 64bit 系统
CPU	Intel(R) Core(TM) I9-9920X 3.50GHz

内存容量	64GB
硬盘空间	2TB
通信接口	Ethernet 1000BASE

2. 文件构成

Release SDK：

AIRScanner.dll	相机底层 sdk 动态链接库
AIRScanner.lib	相机底层 sdk 动态链接库的链接文件
AIRScanner.h	定义通信接口的头文件
AIRTypes.h	定义数据结构的头文件

Debug SDK：

AIRScannerd.dll	相机底层 sdk 动态链接库
AIRScannerd.lib	相机底层 sdk 动态链接库的链接文件
AIRScanner.h	定义通信接口的头文件
AIRTypes.h	定义数据结构的头文件

3. SDK 使用示例

- 在工程中使用 SDK

SDK 包含 bin/include/lib 文件夹直接放入工程目录下即可。 注意区分 bin 文件与 lib 文件下的 debug 版本库与 release 版本库。

- 包含文件

请务必将下列头文件放置在 include 文件夹中：

AIRScanner.h、AIRTypes.h

- 链接

明链接与暗链接均可使用。进行链接时，Release 模式下请链接“AIRScanner.lib”；Debug 模式下请链接“AIRScannerd.lib”。

- CMake 链接

下面为 CMakeLists.txt 链接语句示例（使用 Release 版本）。

```
cmake_minimum_required(VERSION 3.10)
project(AIRDemoCode_C)

# 查找AIRScanner包
find_package(AIRScanner REQUIRED
  PATHS "${CMAKE_SOURCE_DIR}/sdk/lib/cmake/AIRScanner"
  NO_DEFAULT_PATH)

# 添加可执行文件
add_executable(AIRDemoCode_C DemoCode.cpp)

# 包含头文件目录
target_include_directories(AIRDemoCode_C
    PRIVATE
        ${CMAKE_CURRENT_SOURCE_DIR}
)

# 链接AIRScanner库
target_link_libraries(AIRDemoCode_C
    PRIVATE
        AIRScanner::AIRScanner
        ${OpenCV_LIBS}
)

# 复制动态链接库文件
set(3rd_DLLS
    $<TARGET_FILE:AIRScanner::AIRScanner>
)

foreach(dll ${3rd_DLLS})
    add_custom_command(TARGET ${PROJECT_NAME} POST_BUILD
        COMMAND ${CMAKE_COMMAND} -E copy_if_different
        ${dll}
        $<TARGET_FILE_DIR:${PROJECT_NAME}>)
endforeach()
```

4. 数据结构定义

4.1 数据结构一览

文件名	AIRTypes.h
数据结构成员	1) AIRstatus; 2) AIRLogLevel（air_log_level）； 3) AIRExceptionType（air_exception_type）； 4) air_error; 5) AIRPatternMode（air_pattern_mode）； 6) AIRReconstructionMode（air_reconstruction_mode）； 7) AIRScannerInfo（air_scanner_info）； 8) AIRProjectorIdensity（air_projector_idensity）； 9) AIROutlierFilterLevel（air_outlier_filter_level）； 10) AIRSmoothingFilterLevel（air_smoothing_filter_level）； 11) AIRCameraIntrinsics（air_camera_intrinsics）； 12) AIRDistortionCoefficients（air_distortion_coefficients）； 13) AIRScannerParams（air_scanner_params）； 14) AIRPoint3f（air_point_3f）； 15) AIRColorPoint（air_color_point）； 16) AIRPointCloud（air_point_cloud_t）； 17) AIRROI（air_roi）；

4.2 数据结构参考

名称	AIRstatus
所属文件	AIRTypes.h

定义	typedef enum {
	kSuccess = 0,
	kFail = 1,
	kUnknown = 2,
	kBusy = 3,
	kNotConnected = 4,
	kNotSupported = 5
	};

描述	**枚举说明：**用于表示操作的状态。 **枚举常量说明：** 1) kSuccess：操作成功; 2) kFail：操作失败; 3) kUnknown：设备未识别; 4) kBusy：设备忙; 5) kNotConnected：设备未连接; 6) kNotSupported：设备未支持;

表 1 AIRstatus

名称	AIRLogLevel（air_log_level）
所属文件	AIRTypes.h
定义	typedef enum { AIR_LOG_LEVEL_DEBUG = 0, AIR_LOG_LEVEL_INFO = 1, AIR_LOG_LEVEL_WARN = 2, AIR_LOG_LEVEL_ERROR = 3, AIR_LOG_LEVEL_FATAL = 4, AIR_LOG_LEVEL_OFF = 5, } AIRLogLevel, air_log_level;
描述	**枚举说明：**用于表示日志的严重等级。 **枚举常量说明：** 1) AIR_LOG_LEVEL_DEBUG：接口调试日志; 2) AIR_LOG_LEVEL_INFO：接口信息日志; 3) AIR_LOG_LEVEL_WARN：接口警告日志; 4) AIR_LOG_LEVEL_ERROR：接口错误日志; 5) AIR_LOG_LEVEL_FATAL：接口严重错误日志; 6) AIR_LOG_LEVEL_OFF：接口结束日志;

表 2 AIRLogLevel（air_log_level）

名称	AIRExceptionType（air_exception_type）
所属文件	AIRTypes.h
定义	typedef enum { AIR_EXCEPTION_TYPE_UNKNOWN = 0, AIR_EXCEPTION_TYPE_INVALID_ARGUMENT = 1, AIR_EXCEPTION_TYPE_OUT_OF_RANGE = 2, AIR_EXCEPTION_TYPE_NOT_IMPLEMENTED = 3, AIR_EXCEPTION_TYPE_NOT_SUPPORTED = 4, AIR_EXCEPTION_TYPE_RUNTIME = 5, AIR_EXCEPTION_TYPE_LOGIC = 6, AIR_EXCEPTION_TYPE_IO = 7, AIR_EXCEPTION_TYPE_FILE_NOT_FOUND = 8, AIR_EXCEPTION_TYPE_MEMORY = 9, } AIRExceptionType, air_exception_type;
描述	**枚举说明：**用于表示不同的异常类型。 **枚举常量说明：** 1) AIR_EXCEPTION_TYPE_UNKNOWN：未知异常类型； 2) AIR_EXCEPTION_TYPE_INVALID_ARGUMENT：非法参数值； 3) AIR_EXCEPTION_TYPE_OUT_OF_RANGE：数值超过范围； 4) AIR_EXCEPTION_TYPE_NOT_IMPLEMENTED：接口未实现； 5) AIR_EXCEPTION_TYPE_NOT_SUPPORTED：参数不支持； 6) AIR_EXCEPTION_TYPE_RUNTIME：接口内存访问异常； 7) AIR_EXCEPTION_TYPE_LOGIC：接口逻辑异常； 8) AIR_EXCEPTION_TYPE_IO：IO 异常； 9) AIR_EXCEPTION_TYPE_FILE_NOT_FOUND：文件不存在； 10) AIR_EXCEPTION_TYPE_MEMORY：内存访问异常；

表 3 AIRExceptionType（air_exception_type）

名称	air_error
所属文件	AIRTypes.h
定义	typedef struct air_error { AIRStatus status; AIRExceptionType type; const char* message; const char* file; const char* function; int line; } air_error;
描述	**结构体说明：**用于表示错误的详细信息。 **成员变量说明：** 1) AIRStatus status：接口状态码；

	2) AIRExceptionType type：异常捕获类型； 3) const char* messag：详细错误信息； 4) const char* file：错误信息文件； 5) const char* function：错误信息函数； 6) int line：错误信息行号；

表 4 air_error

名称	AIRPatternMode（air_pattern_mode）
所属文件	AIRTypes.h
定义	typedef enum { AIR_PATTERN_MODE_NORMAL = 0, AIR_PATTERN_MODE_ANTI_GHOSTING = 1, AIR_PATTERN_MODE_ANTI_TRANSPARENCY = 2, } AIRPatternMode, air_pattern_mode;
描述	**枚举说明：**用于表示不同的重建模式。 **枚举常量说明：** 1) AIR_PATTERN_MODE_NORMAL：传统模式； 2) AIR_PATTERN_MODE_ANTI_GHOSTING：抗反光模式； 3) AIR_PATTERN_MODE_ANTI_TRANSPARENCY：抗透明模式；

表 5 AIRPatternMode（air_pattern_mode）

名称	AIRReconstructionMode（air_reconstruction_mode）
所属文件	AIRTypes.h
定义	typedef enum { AIR_RECONSTRUCTION_MODE_STEREO = 0, AIR_RECONSTRUCTION_MODE_SINGLE_LEFT = 1, AIR_RECONSTRUCTION_MODE_SINGLE_RIGHT = 2, }AIRReconstructionMode, air_reconstruction_mode;
描述	**枚举说明：**用于表示不同的相机模式。 **枚举常量说明：** 1) AIR_RECONSTRUCTION_MODE_STEREO：双目模式； 2) AIR_RECONSTRUCTION_MODE_SINGLE_LEFT：左相机模式； 3) AIR_RECONSTRUCTION_MODE_SINGLE_RIGHT：右相机模式；

表 6 AIRReconstructionMode（air_reconstruction_mode）

名称	AIRScannerInfo（air_scanner_info）
所属文件	AIRTypes.h
定义	typedef struct { const char* ip; uint16_t port;

	const char* serial_number; const char* name; const char* firmware_version; const char* firmware_git_tag; } AIRScannerInfo, air_scanner_info;
描述	**结构体说明：**用于表示相机信息。 **成员变量说明：** 1) const char* ip：IP 地址； 2) uint16_t port：端口； 3) const char* serial_number：序列号； 4) const char* name：相机型号； 5) const char* firmware_version：硬件版本； 6) const char* firmware_git_tag：硬件 git 标签；

表 7 AIRScannerInfo（air_scanner_info）

名称	AIRProjectorIdensity（air_projector_idensity）
所属文件	AIRTypes.h
定义	typedef enum { AIR_PROJECTOR_IDENSITY_LOW = 0, AIR_PROJECTOR_IDENSITY_MEDIUM, AIR_PROJECTOR_IDENSITY_HIGH } AIRProjectorIdensity, air_projector_idensity;
描述	**枚举说明：**用于设置投影仪亮度等级。 **枚举常量说明：** 1) AIR_PROJECTOR_IDENSITY_LOW：低亮度模式； 2) AIR_PROJECTOR_IDENSITY_MEDIUM：中等亮度模式； 3) AIR_PROJECTOR_IDENSITY_HIGH：高亮度模式；

表 8 AIRProjectorIdensity（air_projector_idensity）

名称	AIROutlierFilterLevel（air_outlier_filter_level）
所属文件	AIRTypes.h
定义	typedef enum { AIR_OUTLIER_FILTER_LEVEL_WEAK = 0, AIR_OUTLIER_FILTER_LEVEL_NORMAL = 1, AIR_OUTLIER_FILTER_LEVEL_STRONG = 2, } AIROutlierFilterLevel, air_outlier_filter_level;
描述	**枚举说明：**用于表示离群点过滤的强度等级。 **枚举常量说明：** 1) AIR_OUTLIER_FILTER_LEVEL_WEAK：弱过滤等级； 2) AIR_OUTLIER_FILTER_LEVEL_NORMAL：正常过滤等级； 3) AIR_OUTLIER_FILTER_LEVEL_STRONG：强过滤等级；

表 9 AIROutlierFilterLevel（air_outlier_filter_level）

名称	AIRSmoothingFilterLevel（air_smoothing_filter_level）
所属文件	AIRTypes.h
定义	typedef enum { AIR_SMOOTHING_FILTER_LEVEL_WEAK = 0, AIR_SMOOTHING_FILTER_LEVEL_NORMAL = 1, AIR_SMOOTHING_FILTER_LEVEL_STRONG = 2, } AIRSmoothingFilterLevel, air_smoothing_filter_level;
描述	**枚举说明：**用于表示数据平滑处理的强度等级。 **枚举常量说明：** 1) AIR_SMOOTHING_FILTER_LEVEL_WEAK：弱平滑等级； 2) AIR_SMOOTHING_FILTER_LEVEL_NORMAL：正常平滑等级； 3) AIR_SMOOTHING_FILTER_LEVEL_STRONG：强平滑等级；

表 10 AIRSmoothingFilterLevel（air_smoothing_filter_level）

名称	AIRCameraIntrinsics（air_camera_intrinsics）
所属文件	AIRTypes.h
定义	typedef struct { float fx; float fy; float cx; float cy; int16_t width; int16_t height; } AIRCameraIntrinsics, air_camera_intrinsics;
描述	**结构体说明：**用于表示相机的内参。 **成员变量说明：** 1) float fx：x 轴焦距； 2) float fy：y 轴焦距； 3) float cx：x 轴主点坐标； 4) float cy：y 轴主点坐标； 5) int16_t width：图像宽度； 6) int16_t height：图像高度；

表 11 AIRCameraIntrinsics（air_camera_intrinsics）

名称	AIRDistortionCoefficients（air_distortion_coefficients）
所属文件	AIRTypes.h
定义	typedef struct {

	float k1; float k2; float k3; float k4; float k5; float k6; float p1; float p2; } AIRDistortionCoefficients, air_distortion_coefficients;
描述	**结构体说明：**用于表示相机的畸变系数。 **成员变量说明：** 1) k1/k2/k3/k4/k5/k6：切向畸变系数； 2) p1/p2：径向畸变系数；

表 12 AIRDistortionCoefficients（air_distortion_coefficients）

名称	AIRScannerParams（air_scanner_params）
所属文件	AIRTypes.h
定义	typedef struct { AIRCameraIntrinsics intrinsics; AIRDistortionCoefficients distortion; AIRTransform transform; }AIRScannerParams, air_scanner_params;
描述	**结构体说明：**用于表示相机的参数集合。 **成员变量说明：** 1) AIRCameraIntrinsics intrinsics：相机内参； 2) AIRDistortionCoefficients distortion：畸变系数； 3) AIRTransform transform：变换矩阵；

表 13 AIRScannerParams（air_scanner_params）

名称	AIRPoint3f（air_point_3f）
所属文件	AIRTypes.h
定义	typedef struct{ float x; float y; float z; } AIRPoint3f, air_point_3f;
描述	**结构体说明：**用于表示点云中的一个点。 **成员变量说明：** 1) float x：x 轴坐标； 2) float y：y 轴坐标； 3) float z：z 轴坐标；

表 14 AIRPoint3f（air_point_3f）

名称	AIRColorPoint（air_color_point）
所属文件	AIRTypes.h
定义	typedef struct{ float x; float y; float z; uint8_t r; uint8_t g; uint8_t b; } AIRColorPoint, air_color_point;
描述	**结构体说明：**用于表示带颜色属性的点云。 **成员变量说明：** 1) float x：x 轴坐标； 2) float y：y 轴坐标； 3) float z：z 轴坐标； 4) uint8_t r：Red 通道值； 5) uint8_t g：green 通道值； 6) uint8_t b：blue 通道值；

表 15 AIRColorPoint（air_color_point）

名称	AIRPointCloud（air_point_cloud_t）
所属文件	AIRTypes.h

定义	typedef struct {
	air_color_point* points = nullptr;
	int points_count = 0;
	} AIRPointCloud, air_point_cloud_t;

描述	**结构体说明：**用于表示三维点云数据。 **成员变量说明：** 1) air_color_point* points：指向三维点数组的指针； 2) int points_count：点的数量；

表 16 AIRPointCloud（air_point_cloud_t）

名称	AIRROI（air_roi）
所属文件	AIRTypes.h
定义	typedef struct { int pt_lt_x; int pt_lt_y; int width;

	int height; }AIRROI, air_roi;
描述	**结构体说明：**用于设置投影仪感兴趣区域范围。 **成员变量说明：** 1) int pt_lt_x：矩形区域左上角顶点的 X 坐标（像素单位）； 2) int pt_lt_y：矩形区域左上角顶点的 Y 坐标（像素单位）； 3) int width：矩形区域的宽度（像素单位）； 4) int height：矩形区域的高度（像素单位）；

表 17 AIRROI（air_roi）

5. AIRLink 类

6.1 AIRLink 类一览

6.1.1 通信建立与相机控制

文件名	AIRScanner
所属成员	AIRScanner.h
概要	搜索设备、连接/断开设备、相机基础参数设置、获取基础参数
类成员函数	1) AIR_CreateScanner; 2) AIR_DestroyScanner; 3) AIR_GetVersion; 4) AIR_Connect; 5) AIR_Disconnect; 6) AIR_SetExposureTime; 7) AIR_GetExposureTime; 8) AIR_SetProjectorIntensity; 9) AIR_SetPatternMode; 10) AIR_SetReconstructionMode; 11) AIR_SetEnableTexture; 12) AIR_SetEnableTextureSupplementary; 13) AIR_SetTextureExposureTime; 14) AIR_SetEnableOutlierFilter; 15) AIR_SetOutlierFilterLevel; 16) AIR_SetEnableDepthSmoothingFilter; 17) AIR_SetDepthSmoothingFilterLevel; 18) AIR_SetDepthRange; 19) AIR_GetDepthRange; 20) AIR_SetROI; 21) AIR_Capture; 22) AIR_ResolvePointCloud;

6.1.2 数据获取

文件名	AIRScanner
所属成员	AIRScanner.h
概要	主要包含数据获取
类成员函数	1)

6.2 AIRLink 类参考

6.2.1 通信建立与相机控制

函数名	AIR_CreateScanner
所属文件	AIRScanner.h
定义	AIR_C_API AIRScannerHandle AIR_CreateScanner();
描述	**说明：**创建投影仪实例。 **入参：** 无入参。 **返回值：** 1) AIRScannerHandle 创建的投影仪实例的句柄。

表 1 AIR_CreateScanner

函数名	AIR_DestroyScanner
所属文件	AIRScanner.h
定义	AIR_C_API void AIR_DestroyScanner(AIRScannerHandle handle);
描述	**说明：**销毁投影仪实例。 **入参：** 1) 需要销毁的投影仪句柄。 **返回值：** 无返回值

表 2 AIR_DestroyScanner

函数名	AIR_GetVersion
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_GetVersion(AIRScannerHandle handle, char* out_version, int max_len);
描述	**说明：**获取 SDK 版本。 **入参：** 1) handle：投影仪句柄。 2) out_version：输出缓冲区，用于存储版本字符串。 3) max_len：输出缓冲区的最大长度。

	返回值： 1) 0 表示成功，非 0 表示失败。

表 3 AIR_GetVersion

函数名	AIR_Connect
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_Connect(AIRScannerHandle handle, AIRScannerInfo* info);
描述	**说明：**连接到投影仪设备。 **入参：** 1) handle：投影仪句柄。 2) info：投影仪的 IP 地址。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 4 AIR_Connect

函数名	AIR_Disconnect
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_Disconnect(AIRScannerHandle handle);
描述	**说明：**断开与投影仪的连接。 **入参：** 1) handle：投影仪句柄。 **返回值：** 1) 0 表示成功，非 0 表示失败。 **注意：在程序结束之前必须调用 AIR_Disconnect 接口，否则占用的资源无法释放，可 能导致重启程序之后出现无法连接相机等问题。**

表 5 AIR_Disconnect

函数名	AIR_SetExposureTime
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_SetExposureTime(AIRScannerHandle handle, int* exposure_time_us, int count);
描述	**说明：**设置投影仪的曝光时间。 **入参：** 1) handle：投影仪句柄。 2) exposure_time_us：曝光时间数组（支持多组配置）。 3) count：输入时为数组长度，输出时为实际返回的配置数量。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 6 AIR_SetExposureTime

函数名	AIR_GetExposureTime
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_GetExposureTime(AIRScannerHandle handle, int* exposure_time_us, int* count);
描述	**说明：**获取投影仪的曝光时间。 **入参：** 1) handle：投影仪句柄。 2) exposure_time_us：曝光时间数组（支持多组配置）。 3) count：输入时为数组长度，输出时为实际返回的配置数量。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 7 AIR_GetExposureTime

函数名	AIR_SetProjectorIntensity
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_SetProjectorIntensity(AIRScannerHandle handle, int Intensity);
描述	**说明：**设置投影仪的投影光亮度。 **入参：** 1) handle：投影仪句柄。 2) Intensity：投影光亮度（范围 [1, 100]）。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 8 AIR_SetProjectorIntensity

函数名	AIR_SetPatternMode
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_SetPatternMode(AIRScannerHandle handle, AIRPatternMode mode);
描述	**说明：**设置投影仪的重建模式。 **入参：** 1) handle：投影仪句柄。 2) mode：重建模式，类型为 AIRPatternMode。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 9 AIR_SetPatternMode

函数名	AIR_SetReconstructionMode
所属文件	AIRScanner.h

定义	AIR_C_API int AIR_SetReconstructionMode(AIRScannerHandle handle, AIRReconstructionMode mode);
描述	**说明：**设置投影仪的相机模式。 **入参：** 1) handle：投影仪句柄。 2) mode：相机模式枚举值。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 10 AIR_SetReconstructionMode

函数名	AIR_SetEnableTexture
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_SetEnableTexture(AIRScannerHandle handle, bool enable);
描述	**说明：**纹理映射开关。 **入参：** 1) handle：投影仪句柄。 2) enable：是否开启纹理映射（0 表示禁用，1 表示启用）。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 11 AIR_SetEnableTexture

函数名	AIR_SetEnableTextureSupplementary
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_SetEnableTextureSupplementary(AIRScannerHandle handle, bool enable);
描述	**说明：**辅助纹理映射开关。 **入参：** 1) handle：投影仪句柄。 2) enable：是否开启辅助纹理映射（0 表示禁用，1 表示启用）。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 12 AIR_SetEnableTextureSupplementary

函数名	AIR_SetTextureExposureTime
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_SetTextureExposureTime(AIRScannerHandle handle, int exposure_time_us);
描述	**说明：**设置纹理映射的曝光时间。 **入参：** 1) handle：投影仪句柄。 2) exposure_time_us：要设置的曝光时间。

	返回值： 1) 0 表示成功，非 0 表示失败。

表 13 AIR_SetTextureExposureTime

函数名	AIR_SetEnableOutlierFilter
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_SetEnableOutlierFilter(AIRScannerHandle handle, bool enable);
描述	**说明：**离群点过滤开关。 **入参：** 1) handle：投影仪句柄。 2) enable：是否开启离群点过滤（0 表示禁用，1 表示启用）。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 14 AIR_SetEnableOutlierFilter

函数名	AIR_SetOutlierFilterLevel
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_SetOutlierFilterLevel(AIRScannerHandle handle, AIROutlierFilterLevel level);
描述	**说明：**设置离群点过滤强度。 **入参：** 1) handle：投影仪句柄。 2) level：离群点过滤强度，类型为 AIROutlierFilterLevel。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 15 AIR_SetOutlierFilterLevel

函数名	AIR_SetEnableDepthSmoothingFilter
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_SetEnableDepthSmoothingFilter(AIRScannerHandle handle, bool enable);
描述	**说明：**表面平滑开关。 **入参：** 1) handle：投影仪句柄。 2) enable：是否开启表面平滑（0 表示禁用，1 表示启用）。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 16 AIR_SetEnableDepthSmoothingFilter

函数名	AIR_SetDepthSmoothingFilterLevel

所属文件	AIRScanner.h
定义	AIR_C_API int AIR_SetDepthSmoothingFilterLevel(AIRScannerHandle handle, AIRSmoothingFilterLevel level);
描述	**说明：**设置表面平滑强度。 **入参：** 1) handle：投影仪句柄。 2) level：表面平滑强度，类型为 AIRSmoothingFilterLevel。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 17 AIR_SetDepthSmoothingFilterLevel

函数名	AIR_SetDepthRange
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_SetDepthRange(AIRScannerHandle handle, int min_depth, int max_depth);
描述	**说明：**设置深度范围。 **入参：** 1) handle：投影仪句柄。 2) min_depth：深度最小值（mm）。 3) max_depth：深度最大值（mm）。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 18 AIR_SetDepthRange

函数名	AIR_GetDepthRange
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_GetDepthRange(AIRScannerHandle handle, int* min_depth, int* max_depth);
描述	**说明：**获取深度范围。 **入参：** 1) handle：投影仪句柄。 2) min_depth：深度最小值（mm）。 3) max_depth：深度最大值（mm）。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 19 AIR_GetDepthRange

函数名	AIR_SetROI
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_SetROI(AIRScannerHandle handle, AIRROI roi);
描述	**说明：**设置感兴趣区域。 **入参：** 1) handle：投影仪句柄。 2) roi：感兴趣区域参数，类型为 AIRROI。

	返回值： 1) 0 表示成功，非 0 表示失败。

表 20 AIR_SetROI

6.2.2 数据获取

函数名	AIR_Capture
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_Capture(AIRScannerHandle handle, AIRFrameData* out_frame);
描述	**说明：**执行单次扫描，此接口以同步的方式执行，会阻塞当前线程，直到扫描完成。 **入参：** 1) handle：投影仪句柄。 2) out_frame：输出帧数据，类型为 AIRFrameData。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 1 AIR_Capture

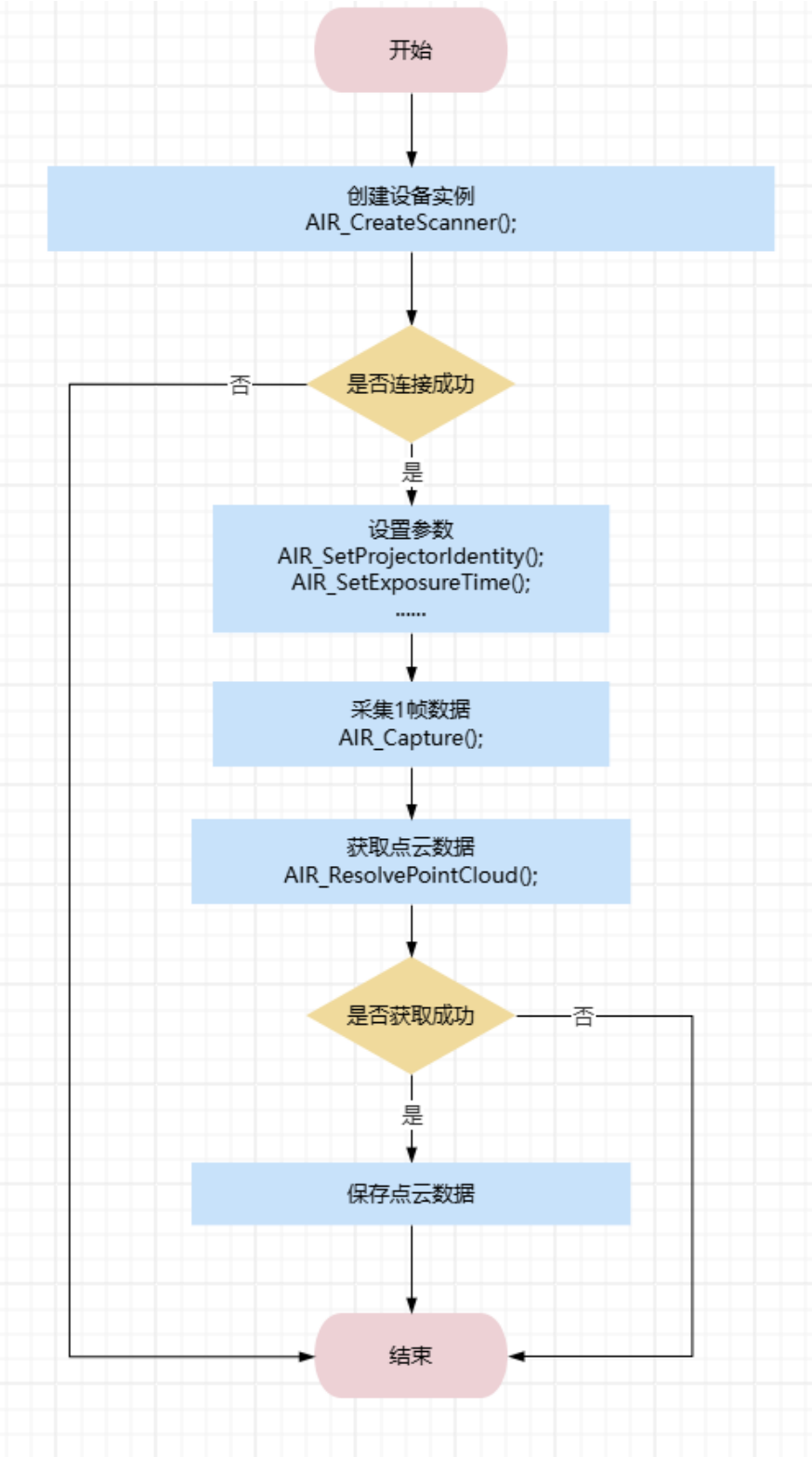
函数名	AIR_ResolvePointCloud
所属文件	AIRScanner.h
定义	AIR_C_API int AIR_ResolvePointCloud(AIRScannerHandle handle, AIRPointCloud* out_point_cloud);
描述	**说明：**保存为点云数据。 **入参：** 1) handle：投影仪句柄。 2) out_point_cloud：输出点云数据，类型为 AIRPointCloud。 **返回值：** 1) 0 表示成功，非 0 表示失败。

表 2 AIR_ResolvePointCloud

6. 接口函数调用示例

6.1 采集 1 次数据并保存点云

基本流程图：



示例代码：

```
#include <AIRScanner.h>
#include <filesystem>
#include <string>
#include <vector>
#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;
using std::vector;
using std::string;
namespace fs = std::filesystem;
vector<AIRScannerInfo> scannerList;
std::vector<int> ExposureValues;
std::vector<int> PatternValues;
int exposureTimes[] = { 5000 };
int maxDepth = 0;
int minDepth = 0;
AIRPointCloud pointCloud;
AIRFrameData frameData;
AIRScannerInfo scannerInfo;

void Scanner_Control(int argc, char* argv[]) {

    //设备信息
    scannerInfo.ip = "10.30.7.157";
    scannerInfo.port = 50051;

    //创建设备实例
    AIRScannerHandle scanner = AIR_CreateScanner();

    //若连接相机失败
    if (AIR_Connect(scanner, &scannerInfo) != 0) {
        cout << "Camera connection failed" << endl;
    }
    else {
        // 设置投影光亮度
        AIR_SetProjectorIdentity(scanner, 100);

        // 设置相机曝光时间为5000，曝光次数为1
        AIR_SetExposureTime(scanner, exposureTimes, 1);

        // 设置相机重建模式为传统模式
        AIR_SetPatternMode(scanner, AIRPatternMode::AIR_PATTERN_MODE_NORMAL);

        // 设置相机模式为双目模式
        AIR_SetReconstructionMode(scanner, AIRReconstructionMode::AIR_RECONSTRUCTION_MODE_STEREO);

        // 设置相机深度范围为0~2000
        AIR_SetDepthRange(scanner, 0, 2000);

        // 获取深度范围并打印
        if (AIR_GetDepthRange(scanner, &minDepth, &maxDepth) == 0) {
            cout << "Depth Range: " << minDepth << "mm - " << maxDepth << "mm" << endl;
        }

        //捕获1帧数据
        memset(&frameData, 0, sizeof(AIRFrameData));
        if (AIR_Capture(scanner, &frameData) == 0) {
            cout << "Frame captured. Depth map size: " << frameData.depth_map_size << endl;
        }

        //获取点云数据
        memset(&pointCloud, 0, sizeof(AIRPointCloud));
        if (AIR_ResolvePointCloud(scanner, &pointCloud) == 0) {
            cout << "Point cloud resolved. Point count: " << pointCloud.points_count << endl;

            std::ofstream outFile("point_cloud_xyz.ply", std::ios::binary);
            if (outFile.is_open()) {
                // 写入 PLY 文件头
                outFile << "ply\n";
                outFile << "format binary_little_endian 1.0\n";
                outFile << "comment Depth Vision's A-series camera point clouds.\n";
                outFile << "element vertex " << pointCloud.points_count << "\n";
                outFile << "property float x\n";
                outFile << "property float y\n";
                outFile << "property float z\n";
                outFile << "end_header\n";

                // 写入点云数据
                if (pointCloud.points && pointCloud.points_count > 0) {
                    // 创建一个临时缓冲区
                    std::vector<float> xyzData(pointCloud.points_count * 3);
                    for (int i = 0; i < pointCloud.points_count; ++i) {
                        xyzData[i * 3 + 0] = pointCloud.points[i].x;
                        xyzData[i * 3 + 1] = pointCloud.points[i].y;
                        xyzData[i * 3 + 2] = pointCloud.points[i].z;
                    }

                    // 写入二进制数据
                    outFile.write(reinterpret_cast<const char*>(xyzData.data()),
```

```
                        pointCloud.points_count * sizeof(float) * 3);
                    if (!outFile.good()) {
                        cerr << "Failed to write point cloud data to file." << endl;
                    }
                }

                outFile.close();
                cout << "Point cloud (XYZ only) saved to point_cloud_xyz.ply, "
                    << "Current working directory: " << filesystem::current_path() << endl;
            }
            else {
                cerr << "Failed to open file for writing." << endl;
            }
        }
        else {
            cerr << "Failed to resolve point cloud." << endl;
        }
    }

    // 关闭设备句柄并销毁
    AIR_Disconnect(scanner);
    AIR_DestroyScanner(scanner);
}

int main(int argc, char* argv[]) {
    Scanner_Control(argc, argv);
    return 0;
}
```