

透视-unity-native-开发文档

本文档主要帮助开发者在使用 xvisio unity plugin sdk 开发 3D 应用的同时，能够在 native 层调用 xvisio c++ sdk，以实现不同应用开发的需求。本文档主要分为两部分：native 层环境搭建以及功能开发；unity 开发环境如何导入 native 层实现的功能。

一、native 层环境搭建以及功能开发

1. 在用 android studio 打开 android-non-root-demo 工程源码；
2. 打开 xv-wrapper.cpp，增加相关功能以及接口，并将需要 unity 调用的接口在 xv-wrapper.h 中声明：

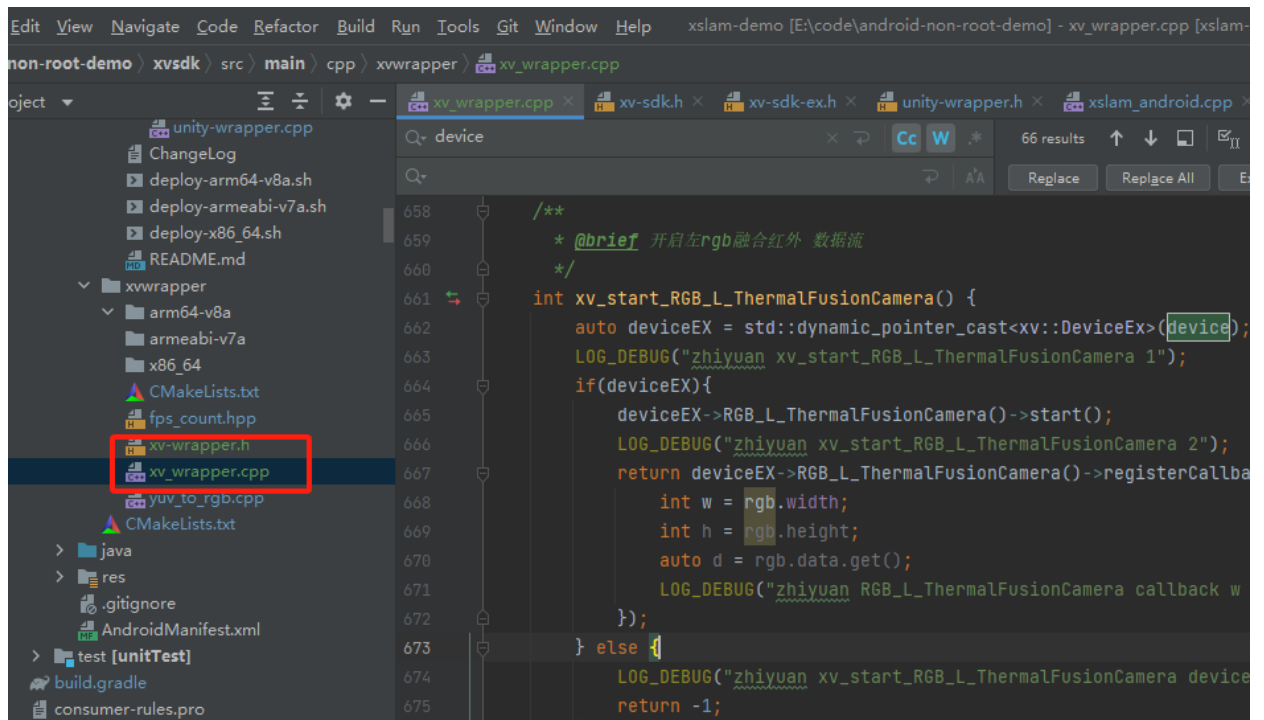


图 1 功能开发

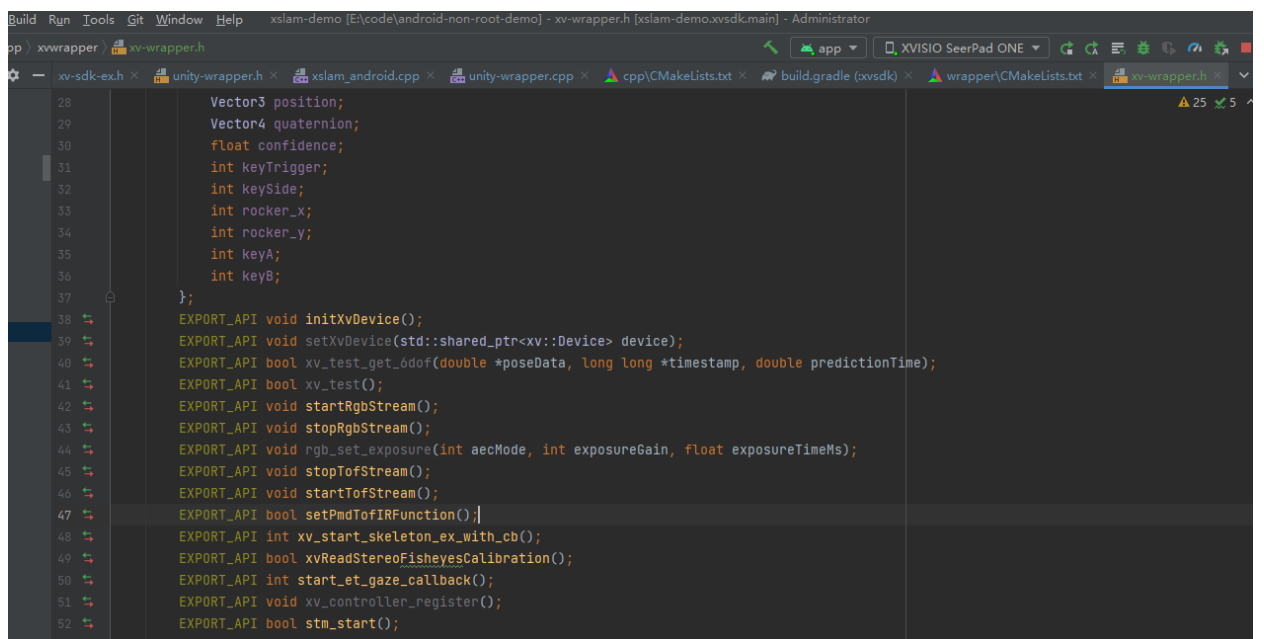


图 2 新增接口头文件

3. 修改完成后点击下图 xvsdk 中的 assemble 编译 wrapper，生成 arm64-v8a 和 armeabi-v7 两个文件夹里面，包含 wrapper 的 so 文件

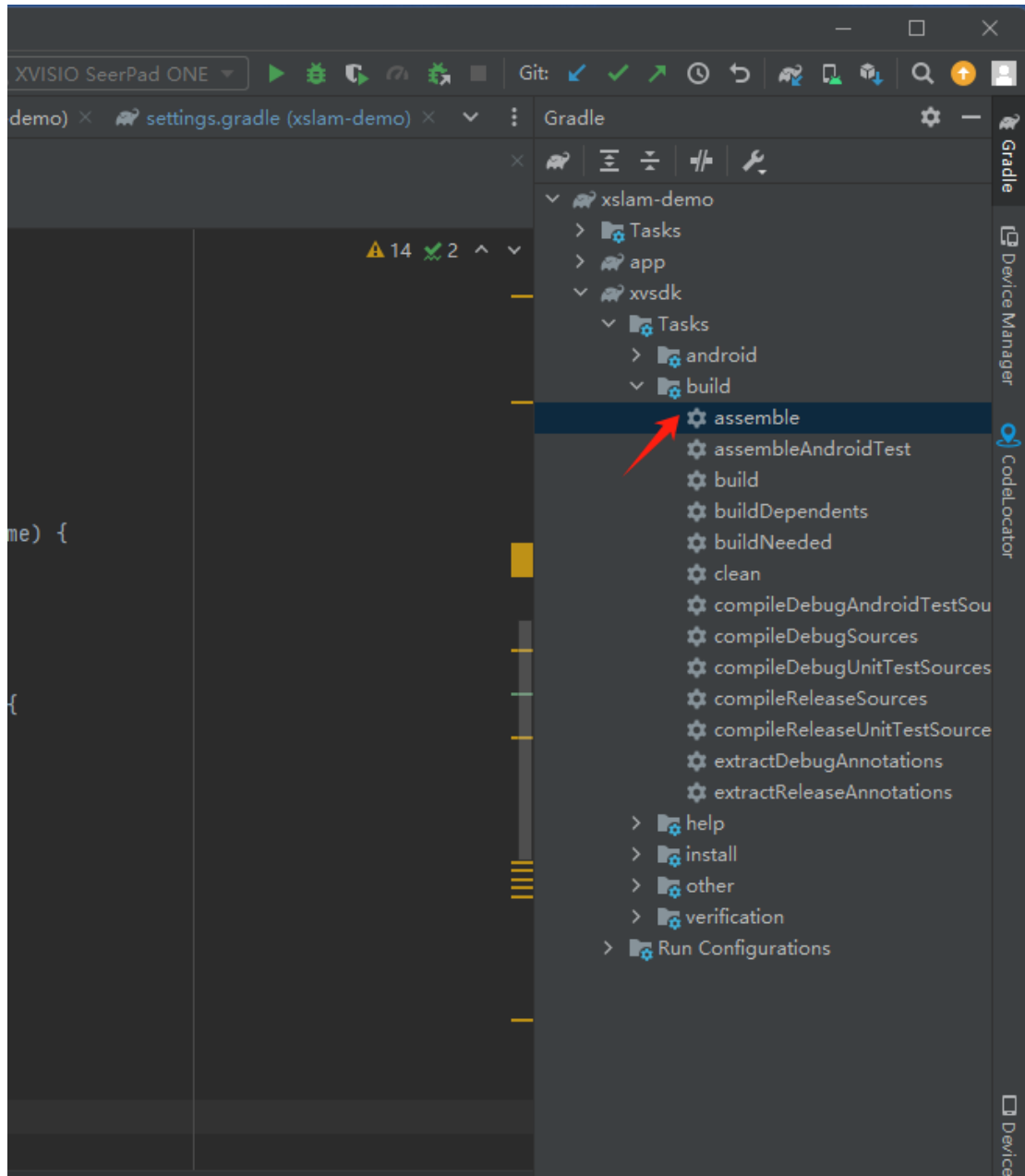


图 3 编译

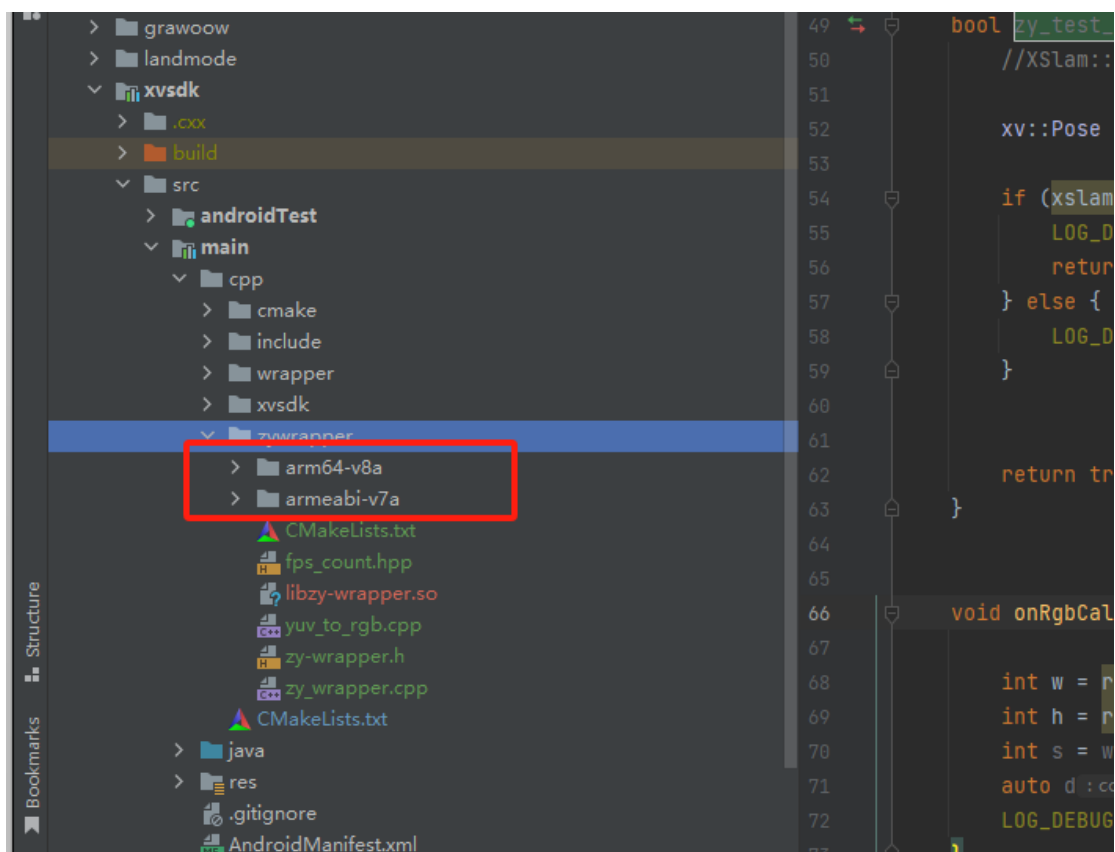


图 4 编译生成的 so

二、unity 开发

1. 将这 2 个文件夹复制到 unity 工程中如下位置：

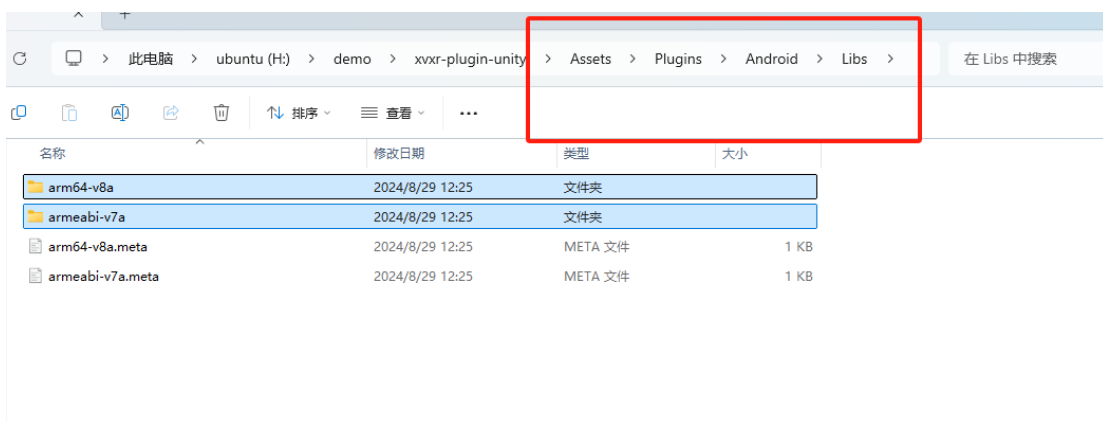


图 5 native so 导入 unity 工程

2. 在 unity 中声明以后可调用 wrapper 封装的接口；

```
public static extern bool xv_stop_infrared_stream(int id);

//识别本地音频
[DllImport("xv-wrapper")]
1 个引用
public static extern void xv_recognize_from_local(string name);

[DllImport("xv-wrapper")]
1 个引用
public static extern void xv_audio_recognize_switch_source(int source);

//音频切换
[DllImport("xv-wrapper")]
```

Unity 工程中的相关测试代码：

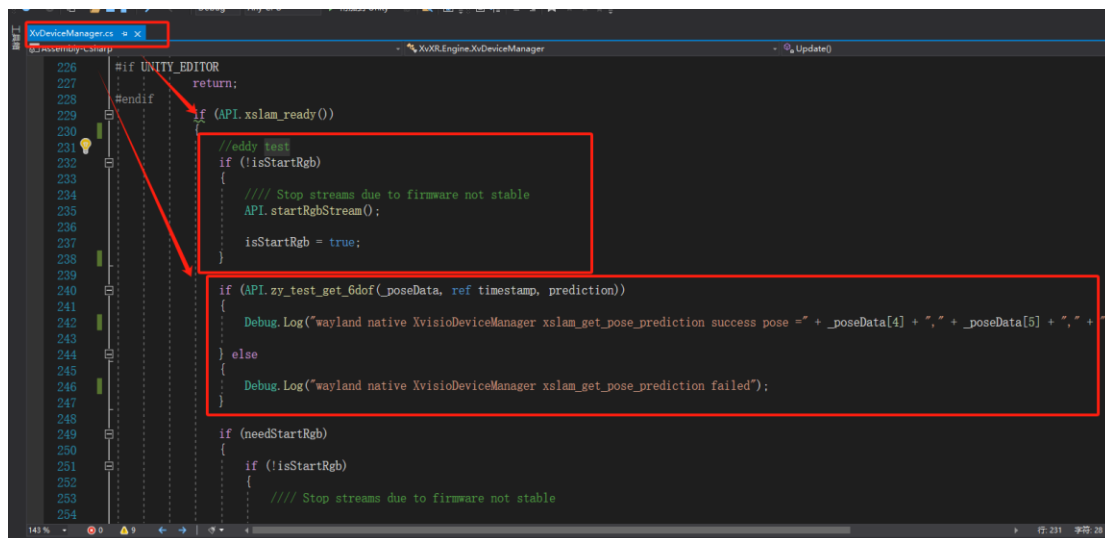


图 6 native 接口导入 unity 工程

3. xvisio native 接口

1. void initXvDevice()

功能描述：

初始化 Xvisio 设备，获取设备指针。

使用说明：

在与 Xvisio 硬件交互之前必须调用该函数进行设备初始化。

2. bool xv_test_get_6dof(double *poseData, long long *timestamp, double predictionTime)

功能描述：

获取 6 自由度（6DoF）的位姿数据，并提供预测。

参数说明：

- poseData：指向存储位姿数据的指针。
- timestamp：指向存储时间戳的指针。
- predictionTime：预测时间，单位为秒。

返回值：

- true：获取位姿成功。
 - false：获取位姿失败。
-

3. void onRgbCallback(const xv::ColorImage &rgb)

功能描述：

处理 RGB 图像数据的回调函数，默认格式为 YUYV。

参数说明：

- rgb：RGB 图像数据。
-

4. void stopRgbStream()

功能描述：

停止 RGB 数据流。

使用说明：

调用此函数可以停止从设备获取 RGB 图像数据流。

5. void startRgbStream()

功能描述：

启动 RGB 数据流。

使用说明：

在调用该函数之前确保设备和相机可用，调用后会注册回调函数 onRgbCallback，并开始接收 RGB 数据。

6. void rgb_set_exposure(int aecMode, int exposureGain, float exposureTimeMs)

功能描述：

设置 RGB 相机的曝光参数。

参数说明：

- aecMode：曝光模式，0 表示自动曝光，1 表示手动曝光。
 - exposureGain：曝光增益，仅在手动曝光模式下有效，范围为 [0, 255]。
 - exposureTimeMs：曝光时间，仅在手动曝光模式下有效，单位为毫秒。
-

7. void onTofCallback(const xv::DepthImage &im)

功能描述：

处理 TOF（飞行时间）深度图像数据的回调函数。

参数说明：

- im：TOF 深度图像数据。

8. void stopTofStream()

功能描述：

停止 TOF 数据流。

使用说明：

调用此函数可以停止从设备获取 TOF 深度数据流。

9. bool setPmdTofIRFunction()

功能描述：

启用 PMD TOF 红外功能。

返回值：

- true：成功启用 IR 功能。
 - false：启用 IR 功能失败。
-

10. void startTofStream()

功能描述：

启动 TOF 数据流。

使用说明：

调用该函数以启动 TOF 相机，并注册回调函数 onTofCallback。

11. int start_skeleton_ex_with_cb()

功能描述：

启动手势识别并注册回调函数获取关键点数据。

返回值：

- 成功时返回回调函数 ID。
 - 失败时返回 -1。
-

12. void gazeCallback(const xv::XV_ET_EYE_DATA_EX &eyedata)

功能描述：

处理眼动数据的回调函数。

参数说明：

- eyedata：眼动数据。
-

13. int start_et_gaze_callback()

功能描述：

启动眼动数据流并注册回调函数 gazeCallback。

返回值：

- 成功时返回回调函数 ID。
 - 失败时返回 -1。
-

14. bool readStereoFisheyesCalibration(stereo_fisheyes *calib, int imu_fisheye_shift_us)

功能描述：

读取双目鱼眼相机的标定参数。

参数说明：

- calib：存储标定参数的结构体。
- imu_fisheye_shift_us：存储 IMU 和鱼眼相机之间的时间偏移，单位为微秒。

返回值：

- true: 成功读取标定参数。
 - false: 读取失败。
-

15. JNIEXPORT void JNICALL Java_com_xv_aitalk_XvInterface_doResult(JNIEnv *env, jclass clazz, jstring result, jint sc, jint id)

功能描述:

处理语音识别结果。

参数说明:

- result: 语音识别返回的结果字符串。
 - sc: 结果置信度
 - id: 定义的语义 id
-

16. void STMDDataCallback(const xv::TerrestrialMagnetismData &stmData)

功能描述:

地磁数据的回调函数，处理并记录接收到的地磁数据。

参数说明:

- stmData: 地磁数据，包括角度、偏移和磁场强度等信息。
-

17. bool stm_start()

功能描述:

启动地磁数据获取，并注册回调函数 STMDDataCallback 用于处理数据。

返回值:

- true: 成功启动地磁数据获取。
-

18. bool stm_stop()

功能描述：

停止地磁数据获取，并取消回调函数的注册。

返回值：

- true：成功停止地磁数据获取。
-

19. void xv_controller_register()

功能描述：

注册手柄数据回调函数，负责接收和处理无线手柄的数据，包括位置信息、姿态信息、按键状态等。

数据处理：

- pose.type：手柄类型。
- pose.position：手柄位置（三维坐标）。
- pose.quaternion：手柄姿态（四元数）。
- pose.confidence：位姿置信度。
- pose.keyTrigger：手柄触发键状态。
- pose.keySide：手柄侧键状态。
- pose.rocker_x、pose.rocker_y：摇杆的 X 和 Y 轴值。
- pose.key：其他按键状态。
- 按键逻辑更新：
 - 当 data.key == 16 时：
 - pose.keyA = 1
 - pose.keyB = 0

- 当 data.key == 32 时:
 - pose.keyA = 0
 - pose.keyB = 1
 - 其他情况下:
 - pose.keyA = 0
 - pose.keyB = 0
-

20. bool xv_start_light_preception()

功能描述:

开启光感检测功能。

返回值:

- 成功启动光感检测返回 true。
-

21. bool xv_stop_light_preception()

功能描述:

关闭光感检测功能。

返回值:

- 成功停止光感检测返回 true。
-

22. int xv_start_beidou_stream(int mode)

功能描述:

启动北斗/GPS 数据流，获取定位信息。

参数说明:

- mode:

- 0: 北斗模式。
- 1: 北斗和 GPS 混合模式。

返回值:

- 成功启动返回回调函数 ID。
- 失败返回 -1。

回调数据结构说明:

- data_ready_flag: 数据有效性标志 (0 无效, 1 有效)。
 - lat_data: 纬度数据。
 - latdir: 纬度方向 (1 南纬, 2 北纬)。
 - lon_data: 经度数据。
 - londir: 经度方向 (1 东经, 2 西经)。
 - satellite_num: 卫星数量。
 - mode: 当前模式。
-

23. void xv_start_imu()

功能描述:

启动 IMU (惯性测量单元) 数据获取。

数据处理:

- accel[]: 加速度数据。
 - gyro[]: 陀螺仪数据。
 - magneto[]: 磁力计数据。
-

24. bool xv_stop_infrared_stream(int id)

功能描述：

停止红外检测，注销回调函数。

参数说明：

- id：注册回调时获得的回调函数 ID。

返回值：

- 成功停止返回 true。
-

25. int xv_start_infrared_stream()

功能描述：

启动红外检测，获取红外图像数据。

返回值：

- 成功启动返回回调函数 ID。
 - 失败返回 -1。
-

26. int xv_start_event_stream()

功能描述：

启动事件流，获取设备的事件信息，如光感数据、佩戴状态、按键事件等(event 详细描述见 XR 应用开发文档)。

返回值：

- 成功启动返回回调函数 ID。
- 失败返回 -1。

事件类型说明：

- 佩戴检测：

- type = 2
 - state = 0: 眼镜摘掉状态。
 - state = 1: 眼镜戴上状态。
- 光感检测:
 - type = 6
 - state: 光感数据值（单位：Lux）。

27. bool xv_switch_rgb()

功能描述：

将显示源切换为微光和红外融合图像模式。

返回值：

- 切换成功返回 true。
-

28. bool xv_switch_display()

功能描述：

将显示源切换为 DP（DisplayPort）源数据。

返回值：

- 切换成功返回 true。
-

29. bool xv_switch_audio(bool status)

功能描述：

切换眼镜的音频开关状态。

参数说明：

- status: false 关闭音频, true 打开音频。

返回值：

- 切换成功返回 true。
-

30. void xv_recognize_from_local(const char *name)

功能描述：

从本地音频文件进行音频识别，目前支持 pcm 和 wav 格式的文件。文件需放置在 sdcard/xv/ 目录下。

参数说明：

- name：音频文件的名称，支持 AudioRecord 录制的
AudioFormat.CHANNEL_IN_MONO 和
AudioFormat.ENCODING_PCM_16BIT 格式的 pcm 或 wav 文件。
-

31. void xv_audio_recognize_switch_source(int source)

功能描述：

切换音频识别的数据来源。

参数说明：

- source：
 - 1：使用麦克风进行实时音频识别。
 - 2：使用本地文件或音频字节数组进行识别。
-

32. void xv_recognize_from_byte(const uint8_t* data, int length)

功能描述：

从音频字节数组进行音频识别。

参数说明：

- data: 音频字节数组。
- length: 音频字节数组的长度。

33. int xv_start_RGB_L_ThermalFusionCamera()

功能描述:

开启左侧 RGB 与红外融合数据流。

返回值:

- 成功返回回调 ID, 否则返回 -1。
-

34. void xv_stop_RGB_L_ThermalFusionCamera(int id)

功能描述:

关闭左侧 RGB 与红外融合数据流。

参数说明:

- id: 回调 ID, 用于取消注册回调。
-

35. int xv_start_RGB_R_ThermalFusionCamera()

功能描述:

开启右侧 RGB 与红外融合数据流。

返回值:

- 成功返回回调 ID, 否则返回 -1。
-

36. void xv_stop_RGB_R_ThermalFusionCamera(int id)

功能描述:

关闭右侧 RGB 与红外融合数据流。

参数说明:

- id: 回调 ID, 用于取消注册回调。
-

37. int xv_start_colorCamera2()

功能描述:

开启 RGB2 数据流。

返回值:

- 成功返回回调 ID。
-

38. void xv_stop_colorCamera2()

功能描述:

关闭 RGB2 数据流。

39. int xv_start_ThermalCamera()

功能描述:

开启红外数据流获取。

返回值:

- 成功返回回调 ID。
-

40. void xv_stop_thermalCamera()

功能描述:

关闭红外数据流

41. void xv_iris_init(JNIEnv *env, jobject context, jstring initLicence)

功能描述:

初始化虹膜识别模块, 获取 JVM 和上下文对象的引用, 并存储初始许可证信息。

参数说明：

- env: JNI 环境指针。
 - context: Java 上下文对象。
 - initLicence: 初始许可证字符串。
-

42. const char* xv_iris_active()

功能描述：

调用设备的激活方法，将虹膜识别模块进行在线激活。

返回值：

- 激活结果字符串。如果失败，返回空字符串。
-

43. void xv_iris_init_licence(const char* licence)

功能描述：

存储提供的离线许可证信息。

参数说明：

- licence: 离线许可证字符串。
-

44. void xv_iris_register(const char* name, fn_iris_callback callback)

功能描述：

注册虹膜特征提取的回调，并启动虹膜识别模块。

参数说明：

- name: 用户名字符串。
 - callback: 回调函数，用于处理提取到的虹膜特征数据。
-

45. void xv_iris_stop()

功能描述：

停止虹膜注册过程，停止虹膜模块并取消注册回调。

46. void xv_iris_start_identity(unsigned char *data, int size, fn_iris_identity_callback callback)

功能描述：

加载虹膜数据并注册一个回调，用于处理身份验证的结果。

参数说明：

- data：虹膜特征数据。
 - size：数据大小。
 - callback：回调函数，用于处理身份验证结果。
-

47. void xv_iris_stop_identity()

功能描述：

停止虹膜身份验证过程，停止虹膜模块并取消身份验证的回调。

48. void xv_iris_setConfigPath(const char* path)

功能描述：

为虹膜模块配置所需的路径信息。

参数说明：

- path：配置文件的路径字符串。

49. void xslam_stereo_set_exposure(float framerate)

功能描述：

设置双目鱼眼摄像头的帧率，用于控制图像采集的速度。

参数说明：

- framerate：帧率值，以帧每秒（FPS）为单位。

注意事项：

- 只有当设备的鱼眼摄像头可用时，函数才会执行操作。
-

50. void xslam_stereo_set_exposure(int aecMode, int exposureGain, float exposureTimeMs)

功能描述：

设置双目鱼眼摄像头的曝光参数，包括自动曝光模式、曝光增益和曝光时间。

参数说明：

- aecMode：自动曝光模式。
- exposureGain：曝光增益，用于调节摄像头的灵敏度。
- exposureTimeMs：曝光时间，以毫秒为单位。

注意事项：

- 只有当设备的鱼眼摄像头可用时，函数才会执行操作。
-

51. void xslam_stereo_set_brightness(int brightness)

功能描述：

设置双目鱼眼摄像头的亮度参数。

参数说明：

- brightness：亮度值，用于调节摄像头的输出亮度，具体值范围由实现决定。

注意事项:

- 只有当设备的鱼眼摄像头可用时，函数才会执行操作。

52. bool xslam_set_rgb_resolution(RgbResolution res)

功能描述:

设置 RGB 彩色相机的分辨率，支持多种预定义分辨率。如果设备包含彩色相机且设置成功，返回 true，否则返回 false。

参数说明:

- res: 目标分辨率，类型为枚举值 RgbResolution，支持以下值：
 - RGB_1920x1080: 分辨率为 1920x1080
 - RGB_1280x720: 分辨率为 1280x720
 - RGB_640x480: 分辨率为 640x480
 - RGB_320x240: 分辨率为 320x240
 - RGB_2560x1920: 分辨率为 2560x1920

返回值:

- 如果分辨率设置成功，返回 true;
- 如果设备没有彩色相机或设置失败，返回 false。

53. xslam_readStereoFisheyesCalibration

功能

读取立体鱼眼相机的标定信息并保存至 m_FECalibration 中。

参数

- **m_FECalibration** (std::vector<xv::CalibrationEx>&): 用于保存读取到的鱼眼相机标定信息。

返回值

- `true`: 如果读取成功并且标定信息有效。
- `false`: 如果读取失败或标定信息无效。

说明

该函数从设备读取立体鱼眼相机的标定数据，存储在 `m_FECalibration` 中。如果标定信息有效，则返回 `true`，否则返回 `false`。

54. readStereoFisheyesCalibration

功能

读取立体鱼眼相机的标定信息，并保存到 `calib` 中。

参数

- **`calib`** (`stereo_fisheyes*`): 用于保存读取到的标定信息。
- **`imu_fisheye_shift_us`** (`int*`): 返回 IMU 和鱼眼相机之间的时间偏移（微秒）。

返回值

- `true`: 标定信息读取成功。
- `false`: 标定信息读取失败。

说明

该函数从设备读取两个鱼眼相机的内外参，并将其存储到 `calib` 中。若读取成功，函数将返回 `true`。

55. readDisplayCalibration

功能

读取显示设备的标定信息，并保存到 `calib` 中。

参数

- **calib** (`pdm_calibration*`): 用于保存读取到的显示设备标定信息。

返回值

- `true`: 显示设备标定读取成功。
- `false`: 显示设备标定读取失败。

说明

该函数从设备的显示模块中读取标定信息，存储到 `calib` 中。返回 `true` 表示读取成功，返回 `false` 表示读取失败。

56. readToFCalibration

功能

读取 ToF (Time of Flight) 相机的标定信息，并保存到 `calib` 中。

参数

- **calib** (`pdm_calibration*`): 用于保存读取到的 ToF 相机标定信息。

返回值

- `true`: ToF 相机标定读取成功。
- `false`: ToF 相机标定读取失败。

说明

该函数从设备的 ToF 相机模块读取标定信息并存储到 `calib` 中。若读取成功，函数返回 `true`，否则返回 `false`。

57. readRGBCalibration

功能

读取 RGB 相机的标定信息，并保存到 calib 中。

参数

- **calib** (rgb_calibration*): 用于保存读取到的 RGB 相机标定信息。

返回值

- true: RGB 相机标定读取成功。
- false: RGB 相机标定读取失败。

说明

该函数从设备的 RGB 相机模块读取标定信息并存储到 calib 中。如果读取成功，则返回 true，否则返回 false。

58. 语音识别

入口代码位于 MainActivity.java 中

```
mBtStartASR.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
//        AsrUtils.getInstance().startAsrRecord(mContext,mBtStartASR,mTvResult);  
        setupAitalk(mContext);  
    }  
});
```

59. bool xv_switch_display_state(int eye_type, bool isOpen)

功能描述：

打开/关闭设备的光学模组使能。

参数说明：

- eye_type: 目标眼类型，用于指定操作的光学模组。取值范围：
 - 1: 左眼光学模组

- 2: 右眼光学模组
- 3: 两眼光学模组
- isOpen: 是否打开光学模组, true 表示打开, false 表示关闭。

注意事项:

- 如果 eye_type 参数无效 (即非 1, 2, 3), 函数将返回 false, 不执行任何操作。
 - 设备对象必须有效, 否则函数将返回 false, 不执行任何操作。
 - 操作成功后, 设备将根据输入参数调整光学模组的状态。
-

60. bool xv_set_electrochromic_level(int level)

功能描述:

设置设备的电致变色模组等级, 用于调整颜色变换的深浅程度。

参数说明:

- level: 电致变色等级, 范围为 0-20 (0 表示最浅, 20 表示最深)。

注意事项:

- 如果 level 参数超出范围 (小于 0 或大于 20), 函数将返回 false, 不执行任何操作。
- 设备对象必须有效, 否则函数将返回 false, 不执行任何操作。
- 调用此函数后, 设备将根据输入的 level 值调整电致变色模组的状态。

61. void xslam_gaze_set_config_path(const char* coe_path)

功能描述:

设置眼动模块初始化所需的配置 JSON 文件路径。

参数说明:

- `coe_path`: 配置文件的路径 (UTF-8 编码的字符串), 通常为 `.json` 格式, 包含眼动系统的初始化参数。

注意事项:

- 请确保配置文件路径合法且文件存在, 否则初始化可能失败。
-

62. void xslam_set_gaze_configs(int width, int height, float ipdDist, int srValue, int etWidth, int etHeight, int loplenth, float etFoclen, float etOccupy, float ftFoclen, int hiValue)

功能描述:

设置眼动追踪系统的相关初始化参数, 包括图像尺寸、焦距、瞳距等。

参数说明:

- `width / height`: 输入图像分辨率 (宽 / 高)
- `ipdDist`: 瞳距, 单位毫米
- `srValue`: 采样频率 (单位: Hz)
- `etWidth / etHeight`: 眼动相机图像的分辨率
- `loplenth`: 校准点数量
- `etFoclen`: 眼动相机焦距 (单位: mm)
- `etOccupy`: 眼动图像占据视野比例 (0~1)
- `ftFoclen`: 人脸追踪焦距
- `hiValue`: 是否启用高精度模式 (0 否 / 1 是)

注意事项:

- 必须在调用 `xslam_start_gaze` 前设置。
-

63. bool xslam_start_gaze()

功能描述：

启动眼动追踪系统。

返回值：

- true 表示启动成功
- false 表示启动失败

注意事项：

- 启动前需先设置配置文件和初始化参数。
-

64. bool xslam_set_exposure(int leftGain, float leftTimeMs, int rightGain, float rightTimeMs)

功能描述：

设置眼动摄像头的曝光时间和增益参数。

参数说明：

- leftGain / rightGain：左右眼摄像头的增益值
- leftTimeMs / rightTimeMs：左右眼摄像头的曝光时间（单位：毫秒）

注意事项：

- 曝光和增益的取值范围受底层驱动限制。
-

65. bool xslam_set_bright(int eye, int led, int brightness)

功能描述：

设置眼动系统中指定眼、指定 LED 的亮度。

参数说明：

- eye: 0 表示左眼, 1 表示右眼
- led: LED 编号
- brightness: 亮度值 (0~255)

注意事项:

- 不支持的 LED 编号将被忽略。
-

66. int xslam_set_gaze_callback(fn_gaze_callback cb)

功能描述:

注册一个回调函数, 用于接收眼动追踪数据。

参数说明:

- cb: 回调函数指针, 类型为 fn_gaze_callback, 签名例如:

```
void fn_gaze_callback(int eyeIndex, float x, float y, float confidence);
```

返回值:

- 0 表示注册成功, 非 0 表示失败

注意事项:

- 若需要取消回调, 可使用 xslam_unset_gaze_callback。
-

67. int xslam_set_pref(int etIdx, const uint8_t* data, int size)

功能描述:

加载和设置已保存的校准数据。

参数说明:

- etIdx: 眼动设备索引
- data: 校准数据的字节数组

- size: 数据长度

返回值:

- 0 表示成功, 其他为错误码

注意事项:

- 校准数据格式需与获取时保持一致。
-

68. void xslam_set_usr_eye_ready()

功能描述:

设置用户状态为“已准备好进行眼动追踪”。

注意事项:

- 建议在用户面部进入识别区域时调用此接口。
-

69. int xslam_gaze_calibration_begin(int et_idx)

功能描述:

启动眼动追踪校准过程。

参数说明:

- et_idx: 眼动设备索引

返回值:

- 0 表示成功, 其他为错误码

注意事项:

- 必须在设备准备就绪后再调用。
-

70. int xslam_gaze_set_conf_gaze(int et_idx, int s_idx, float conf_gaze_o[3])

功能描述:

设置某个校准点的期望注视方向。

参数说明：

- et_idx: 设备索引
- s_idx: 校准点序号
- conf_gaze_o: 长度为 3 的方向向量 (XYZ)

返回值：

- 0 表示成功，其他为错误码

注意事项：

- 确保调用时 conf_gaze_o 是单位向量。
-

71. int xslam_gaze_calibration_end(int et_idx)

功能描述：

结束眼动校准过程，并保存结果。

参数说明：

- et_idx: 设备索引

返回值：

- 0 表示成功，其他为错误码
-

72. int xslam_get_pref(int etIdx, uint8_t* data, int* size)

功能描述：

获取当前的眼动校准数据。

参数说明：

- etIdx: 眼动设备索引

- data: 数据缓冲区指针
- size: 输入为缓冲区大小, 输出为实际写入大小

返回值:

- 0 表示成功, 其他为错误码

注意事项:

- 调用前需分配足够大的缓冲区。
-

73. void xslam_gaze_enable_dump(bool enable)

功能描述:

启用或禁用眼动追踪数据集的保存。

参数说明:

- enable: true 启用, false 禁用
-

74. bool xslam_unset_gaze_callback()

功能描述:

取消已注册的眼动数据回调函数。

返回值:

- true 表示成功
 - false 表示失败或未注册
-

75. bool xslam_stop_gaze()

功能描述:

停止眼动追踪系统。

返回值:

- true 表示成功
- false 表示失败或未运行

注意事项:

- 停止前应确保不再有活跃的回调或数据读取行为。

三、android demo 开发

1 语音识别定制开发（非定制客户不需要）

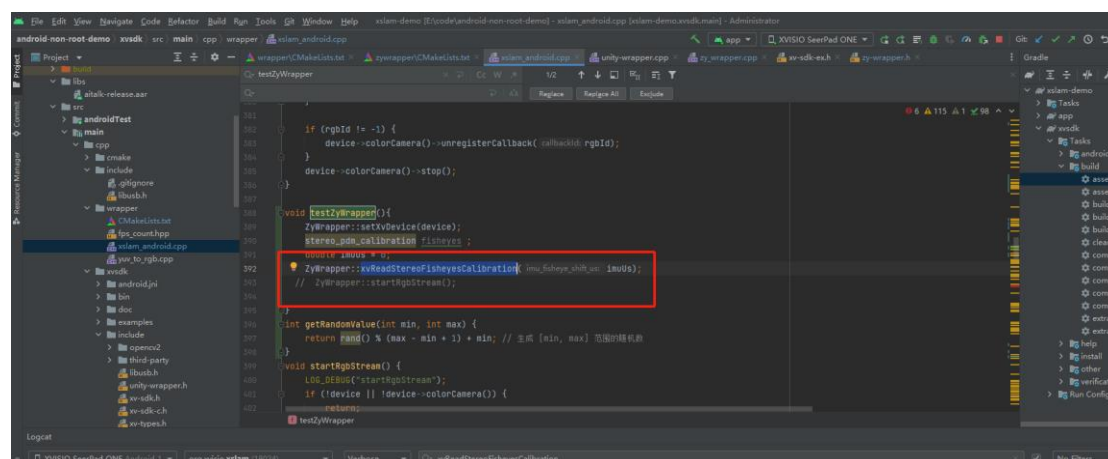
首先将 model 文件放置到 android-non-root-demo\asrSdk\src\main\assets\model 目录, 然后打开 app 模块中 MainActivity.java 中 66 行 isCustomerAsr 改为 true

2 android 原生应用简单测试 native 接口方法

Androidstudio 版本请到

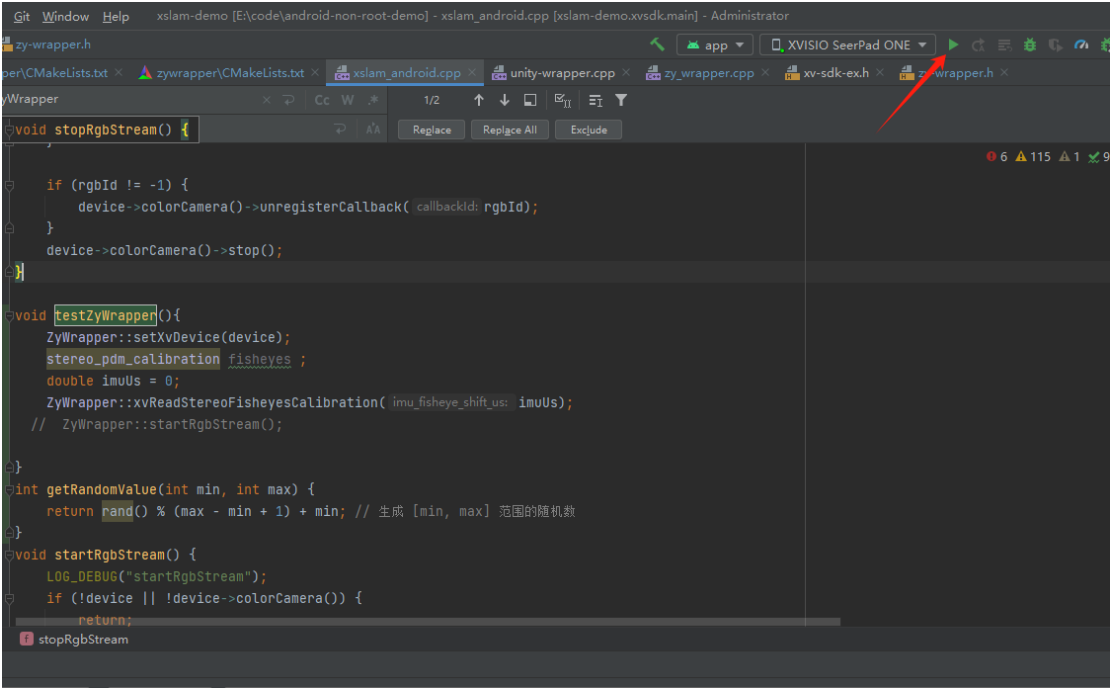
https://www.androiddevtools.cn/android-studio.html#google_vignette 下载安装 Android studio2021.3.1，保持版本一致。

在 xslam android.cpp 中如下位置增加测试接口



然后点击安装 xslam 应用到盒子上，目前界面暂时只有基本功能显示，其他功能

可以通过打印 log 方式查看结果



```
void stopRgbStream() {  
    if (rgbId != -1) {  
        device->colorCamera()->unregisterCallback( callbackId: rgbId);  
    }  
    device->colorCamera()->stop();  
}  
  
void testZyWrapper(){  
    ZyWrapper::setXvDevice(device);  
    stereo_pdm_calibration fisheyes ;  
    double imuUs = 0;  
    ZyWrapper::xvReadStereoFisheyesCalibration( imu_fisheye_shift_us: imuUs);  
    // ZyWrapper::startRgbStream();  
}  
  
int getRandomValue(int min, int max) {  
    return rand() % (max - min + 1) + min; // 生成 [min, max] 范围的随机数  
}  
  
void startRgbStream() {  
    LOG_DEBUG("startRgbStream");  
    if (!device || !device->colorCamera()) {  
        return;  
    }  
    stopRgbStream
```



```
* @brief 读取鱼眼标定参数  
*/  
bool xvReadStereoFisheyesCalibration(double imu_fisheye_shift_us) {  
    std::vector<xv::CalibrationEx> feCalib;  
    auto m_FECalibration : vector<CalibrationEx> = std::dynamic_pointer_cast<xv::FisheyeCamerasEx>( m_device->fisheyeCameras()->calibration);  
    // FishEyeSeucmCalibration fishEyeSeucmCalibration;  
    if(m_FECalibration.size() > 0 && m_FECalibration[0].seucm.size() > 0){  
        for (int i = 0; i < m_FECalibration.size(); i++) {  
            #ifdef ANDROID  
                __android_log_print( prio: ANDROID_LOG_WARN, tag: "XVXR",  
                    fmt: "eddy xvReadStereoFisheyesCalibration entry %f,%f,%f,%f,%f,%f,%f,%f,%f",m_FECalibration[i].seucm[0].fx,m_FECalibration[i].seucm[0].fy,m_FECalibration[i].seucm[0].u0,m_FECalibration[i].seucm[0].eu,m_FECalibration[i].seucm[0].ev,m_FECalibration[i].seucm[0].alpha,m_FECalibration[i].seucm[0].beta);  
            #endif  
        }  
    }  
}
```



```
402      return;
      stopRgbStream

slam (18490)  Verbose  xvReadStereoFisheyesCalibration
org.xvisio.xslam W/xvvr: eddy xvReadStereoFisheyesCalibration entry 276.041290,276.041290,320.251465,202.875732,320.059937,202.553589,0.589281,1.156389,0.000000,0.000000
org.xvisio.xslam W/xvvr: eddy xvReadStereoFisheyesCalibration entry 279.364746,279.364746,312.954651,201.887894,314.269409,201.322296,0.603690,1.106606,0.000000,0.000000
```