# 合肥钛灵信息科技有限公司

版权所有 侵权必究

## 智慧相册 SDK接口文档

（**v2.1.43**）

|  |  |
| --- | --- |
| 系统名称 | Linux ARM |
| 负责人 | 李晨 |
| 文档发布日期 | 2023/03/18 |

## 一、SDK概述

### 1.1、SDK功能简介

1. 人脸聚类

2. 场景分类

### 1.2、SDK文件目录说明

smart\_album\_sdk/

├── 3third

│   ├── include

│   └── so // 库文件

├── bin

│   ├── data // 数据集

│   ├── models // 模型文件

├── CMakeLists.txt

├── doc // 开发文档

└── tools // 测试脚本

### 1.3、编程语言

Linux ARM  C/C++

### 1.4、运行环境

Linux平台与ARM 平台

### 1.5、使用方法

1) 鉴权（**InitParams）**

在调用人脸聚类、场景分类算法前，先调用鉴权接口。返回值为0表示鉴权成功。

2)人脸聚类算法

2.1)提取特征（ExtractImageFeats）

提取一张图片中所有人脸的特征，返回值为0表示成功。

2.2)人脸聚类（FaceCluster、PREFaceCluster）

在提取50~100张图片中人脸特征后，建议先用这些特征进行全量聚类。

继续提取50~100张图片中人脸特征在全量聚类基础上再进行增量聚类。

返回值为0表示聚类成功。

2.3)场景分类（SceneDetect）

调用SceneDetect接口获取场景编号。（编号对应的场景类别在下文中定义）

### 1.6、算法支持的视频/图象格式

1)人脸聚类

支持主流的图片格式

BMP DIB JPEG JPG JPE PNG PBM PGM PPM WEBP RAS TIFF TIF EXR HEIC HEIF

2)场景分类

支持绝大多数的视频格式

avi mp4 mvi mov mvi tty flv gif  vcd vivo webm webp xwma xmv yuv4mpegpipe yop

支持主流的图片格式

BMP DIB JPEG JPG JPE PNG PBM PGM PPM WEBP RAS TIFF TIF EXR HEIC HEIF

### 1.7 常见状态码

* + 10008 鉴权失败 10101 服务端连接失败
  + 10104 鉴权时间过期 10106 lisence文件不存在
  + 20001 模型错误 20002 输入图片为空
  + 20003 模型推理错误 20004 对象指针为空
  + 20005 输入通道数错误 20006 模型文件不存在
  + 20008 人脸区域太小 20010 图片格式不支持
  + 20011 视频太大 20012 输入视频为空
  + 20013 图片长宽比例不支持 20014 图片过大

## 二、C 接口参数说明

### 2.1 鉴权

函数声明：

int InitParams(const char\* lisence\_path, const char\* appId, const char\* appSecret ); int InitSNParams(const char\* lisence\_path, const char\* appId, const char\* appSecret , const char \*sn\_code );

函数功能：

鉴权，在其他算法之前调用。

参数列表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **类别** | **参数名称** | **是否必选** | **类型** | **默认值** | **描述** |
| 入参1 | license\_path | 是 | const char\* | 无 | 获取的license路径(不包括文件名称) |
| 入参2 | appId | 是 | const char\* | 无 | id |
| 入参3 | appSecret | 是 | const char\* | 无 | secret |
| 入参4 | sn\_code | 否 | const char\* | 有 | SN码仅支持字母和数字长度不超过32位 |
| 出参 | 返回值 |  | int |  | 状态码 |

### 2.2 特征提取

### 2.2.1 GetExtractHandle

函数声明

struct tiorb\_image\_feats\_info {

int num\_face = 0; // 检测到的人脸数量

int\* location = nullptr; // 人脸坐标

float\* quality = nullptr; // 人脸质量分

unsigned char\* img = nullptr; // 封面图片

int\* img\_size = nullptr; // 封面图片大小

float\* FaceFeature = nullptr; // 人脸特征

};

struct Handle\* GetExtractHandle();

函数功能

获取句柄，用于多线程场景。

### 2.2.2 ExtractInit

函数声明

int ExtractInit(Handle\* handle, const char\* ModelPath, int ForwardType, int ModelType);

参数列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **类别** | **参数名称** | **是否必选** | **类型** | **默认值** | **描述** |
| 入参1 | handle | 是 | Handle\* | 无 | 设置句柄 |
| 入参2 | ModelPath | 是 | const char\* | 无 | 设置模型路径(不包括模型名称) |
| 入参3 | ForwardType | 是 | const char\* | 无 | 设置0(CPU)或者1(GPU) |
| 入参4 | ModelType | 是 | const char\* | 无 | 设置0(大模型)或者1(小模型) |
| 出参 | 返回值 |  | int |  | 状态码 |

函数功能

初始化，设置模型路径、CPU或者GPU、模型类型

### 2.2.3 ExtractImageFeats

函数声明

int ExtractImageFeats(Handle\* handle, const unsigned char\* src, int size, tiorb\_image\_feats\_info\* feats\_info, float scale=1.5, int outsize=112);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | *handle* | *是* | *Handle\** | *无* | *设置句柄* |
| *入参2* | *src* | *是* | *const char\** | *无* | *二进制图片* |
| *入参3* | *size* | *是* | *int* | *无* | *图片大小* |
| *入参4* | *feats\_info* | *是* | *结构体指针* | *无* | *返回结果* |
| *入参5* | *scale* | *否* | *float* | *有* | *设置人脸比例，默认值1.5f* |
| *入参6* | *outsize* | *否* | *int* | *有* | *返回的人脸尺寸，默认值112* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

提取单张图片中所有人脸特征的接口

提取的人脸特征需要在数据库中保存

支持设置人物封面图片的比例和尺寸

### 2.2.3 ExtractDestroyStruct

函数声明

int ExtractDestroyStruct(tiorb\_image\_feats\_info\* feats\_info);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | feats\_info | *是* | *结构体指针* | *无* | *释放结构体指针* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

释放结构体所占内存

在每次提取特征后调用

### 2.2.4 ExtractDestroyModel

函数声明

int ExtractDestroyModel(Handle\* handle);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | handle | *是* | *指针* | *无* | *释放Handle指针* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

释放模型所占内存

释放模型后就无法提取人脸特征

## 2.3 人脸聚类

### 2.3.1 FaceCluster

函数声明

int FaceCluster(int num\_feats, float\* feats, tiorb\_face\_cluster\_info\* cluster\_info, float similarity=0.53f);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | *num\_feats* | *是* | *int* | *无* | *参与聚类的特征数量* |
| *入参2* | *feats* | *是* | *float\** | *无* | *参与聚类的特征* |
| *入参3* | *cluster\_info* | *是* | *结构体指针* | *无* | *返回聚类结果* |
| *入参4* | *similarity* | *否* | *float* | *无* | *相似度，默认值0.53* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

全量聚类,在聚类前需要提取一定数量的人脸特征（建议50~100张图片）

第一次聚类时建议使用全量聚类，后面使用增量聚类接口

### 2.3.2 PREFaceCluster

函数声明

int PREFaceCluster(int centernums, float\* centerfeats, float\* averagefeats, int\* centerlabels, int\* labelnums, int newnums, float\* newfeats, tiorb\_face\_cluster\_info\* cluster\_info, float similarity=0.53f);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **类别** | **参数名称** | **是否必选** | **类型** | **默认值** | **描述** |
| *入参1* | *centernums* | *是* | *int* | *无* | *现存的聚类数量* |
| *入参2* | *centerfeats* | *是* | *float\** | *无* | *每个聚类的中心向量* |
| *入参3* | *averagefeats* | *是* | *float\** | *无* | *每个聚类的平均向量* |
| *入参4* | *centerlabels* | *是* | *int\** | *无* | *现存的聚类类别* |
| *入参5* | *labelnums* | *是* | *int\** | *无* | *每个类别包含的数量* |
| *入参6* | *newnums* | *是* | *int* | *无* | *本次聚类参与的数量* |
| *入参7* | *newfeats* | *是* | *int\** | *无* | *本次聚类的特征* |
| *入参8* | *cluster\_info* | *是* | *结构体指针* | *无* | *返回聚类结果* |
| *入参9* | *similarity* | *否* | *float* | *有* | *相似度，默认值0.53* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

在聚类前需要提取一定数量的人脸特征（建议50~100张图片）

第一次聚类建议用全量聚类接口，后面使用增量聚类接口

cluster\_info 返回聚类结果（labels）中相同标签的就属于同一类

cluster\_info 返回聚类结果（labels）中前num\_cluster是已存在类别

### 2.3.3 ClusterDestroy

函数声明

int ClusterDestroy(tiorb\_face\_cluster\_info\* cluster\_info);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | *cluster\_info* | *是* | *结构体指针* | *无* | *释放结构体指针* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

释放结构体所占内存

释放后无法再进行聚类

## 2.4 聚合与拆分

### 2.4.1 MergeCenters

函数声明

float\* MergeCenters(float\* feat1, int num1, float\* feat2, int num2);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | feat1 | *是* | *float\** | *无* | 类别1的所有特征向量 |
| *入参2* | num1 | *是* | *int* | *无* | 类别1的特征数量 |
| *入参3* | feat2 | *是* | *float\** | *无* | 类别2的所有特征向量 |
| *入参4* | num2 | *是* | *int* | *无* | 类别2的特征数量 |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

在合并两个类别时，调用此接口合并两个类别的中心向量

### 2.4.2 MergeCentersDestroy

函数声明

int MergeCentersDestroy(float\* center\_descriptors);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | center\_descriptors | *是* | *float\** | *无* | 释放指针数组 |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能：

在合并两个类别时，调用此接口释放float\* 数组指针

### 2.4.3 UpdateCenters

函数声明

float\* UpdateCenters(int num\_feats, float\* feats);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | num\_feats | *是* | *int* | *无* | 特征数量 |
| *入参2* | feats | *是* | *float\** | *无* | 特征数组 |
| *出参* | *返回值* |  | *float\** |  | *float数组* |

函数功能

把一个类别拆分成两个或者以上类别时，调用此接口更新拆分后每个类别的中心向量

### 2.4.4 UpdateCentersDestroy

函数声明

int UpdateCentersDestroy(float\* center\_descriptors);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | center\_descriptors | *是* | *float\** | *无* | 释放指针数组 |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

把一个类别拆分成两个或者以上类别时，调用此接口释放float\* 数组指针

### 2.5 获取封面头像

### 2.2.1 GetPhotoHandle()

函数声明

struct tiorb\_face\_photo\_info {

unsigned char\* img = nullptr;

int img\_size = 0;

};

struct Handle\* GetPhotoHandle();函数功能

获取句柄，用于多线程场景。

### 2.2.2 PhotoInit

函数声明

int PhotoInit(Handle\* handle, const char\* ModelPath, int ForwardType);

参数列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **类别** | **参数名称** | **是否必选** | **类型** | **默认值** | **描述** |
| 入参1 | handle | 是 | Handle\* | 无 | 设置句柄 |
| 入参2 | ModelPath | 是 | const char\* | 无 | 设置模型路径(不包括模型名称) |
| 入参3 | ForwardType | 是 | const char\* | 无 | 设置0(CPU)或者1(GPU) |
| 出参 | 返回值 |  | int |  | 状态码 |

函数功能

初始化，设置模型路径、CPU或者GPU

### 2.2.3 PhotoExtract

函数声明

PhotoExtract(Handle\* handle, const unsigned char\* src, int size, int\* rect, tiorb\_face\_photo\_info\* photo\_info, float scale=1.5f, bool rotation=false);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | *handle* | *是* | *Handle\** | *无* | *设置句柄* |
| *入参2* | *src* | *是* | *const char\** | *无* | *二进制图片* |
| *入参3* | *size* | *是* | *int* | *无* | *图片大小* |
| *入参4* | *rect* | *是* | int\* | int\* | *人脸坐标* |
| *入参5* | photo\_info | *是* | *结构体指针* | *无* | *返回结果* |
| *入参6* | *scale* | *否* | *float* | *有* | *设置人脸比例，默认值1.5f* |
| *入参7* | rotation | *否* | bool | *有* | 旋转检测人脸 |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

根据人脸坐标生成封面图片

### 2.2.3 PhotoDestroyStruct

函数声明

PhotoDestroyStruct(tiorb\_face\_photo\_info\* photo\_info);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | photo\_info | *是* | *结构体指针* | *无* | *释放结构体指针* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

释放结构体所占内存

在每次提取特征后调用

### 2.2.4 PhotoDestroyModel

函数声明

PhotoDestroyModel(Handle\* handle);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | handle | *是* | *指针* | *无* | *释放Handle指针* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

释放模型所占内存

释放模型后就无法提取人脸特征

## 2.5 场景分类

### 2.5.1 GetExtractHandle

函数声明

struct tiorb\_scene\_detect\_info {

int num\_scenes = 0; // index数组长度

int \*index = nullptr; // index数组，场景

label float \*accuracy = nullptr; // 视频分类时，输出预测类别的概率

};

struct Handle \*GetSceneHandle();

函数功能

获取句柄，用于多线程场景。

### 2.5.2 SceneInit

函数声明

int SceneInit(Handle \*handle, const char \*model\_path, int forward\_type);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | *handle* | *是* | *Handle\** | *无* | *设置句柄* |
| *入参2* | *ModelPath* | *是* | *const char\** | *无* | *设置模型路径(不包括模型名称)* |
| *入参3* | *ForwardType* | *是* | *const char\** | *无* | *设置0(CPU)或者1(GPU)* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

初始化，设置模型路径、CPU或者GPU

### 2.5.3 SceneDetect

函数声明

int SceneDetect(Handle \*handle, const unsigned char \*src, int size, tiorb\_scene\_detect\_info \*scene\_info);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | *handle* | *是* | *Handle\** | *无* | *设置句柄* |
| *入参2* | *src* | *是* | *const char\** | *无* | *二进制图片* |
| *入参3* | *size* | *是* | *const char\** | *无* | *图片大小* |
| *入参4* | scene\_info | *是* | *结构体指针* | *无* | *返回结果* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

场景识别，获取输入图片的类别

### 2.5.4 VideoPathDetect

函数声明

int VideoPathDetect(Handle \*handle, const char \*src\_path, float thresh, tiorb\_scene\_detect\_info \*scene\_info);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | *handle* | *是* | *Handle\** | *无* | *设置句柄* |
| *入参2* | src\_path | *是* | *const char\** | *无* | 视频路径包括文件名称 |
| *入参3* | thresh | *否* | *const char\** | *有* | 阈值，默认0.4 |
| *入参4* | scene\_info | *是* | *结构体指针* | *无* | *返回结果* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

场景识别，获取输入视频路径的类别

### 2.5.5 VideoBufferDetect

函数声明

int VideoBufferDetect(Handle \*handle, const unsigned char \*src, int size, float thresh, tiorb\_scene\_detect\_info \*scene\_info);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | *handle* | *是* | *Handle\** | *无* | *设置句柄* |
| *入参2* | *src* | *是* | *const char\** | *无* | *二进制图片* |
| *入参3* | *size* | *是* | *const char\** | *无* | *图片大小* |
| *入参4* | scene\_info | *是* | *结构体指针* | *无* | *返回结果* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

场景识别，获取输入视频buffer的类别

只支持GIF和短视频类型

### 2.5.6 SceneDestroyStruct

函数声明

int SceneDestroyStruct(tiorb\_scene\_detect\_info \*scene\_info);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | scene\_info | *是* | *结构体指针* | *无* | *释放结构体指针* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

释放结构体所占的内存

每次推理后都需要调用

### 2.5.7 SceneDestroyModel

函数声明

int SceneDestroyModel(Handle \*handle);

函数参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***类别*** | ***参数名称*** | ***是否必选*** | ***类型*** | ***默认值*** | ***描述*** |
| *入参1* | *handle* | *是* | *Handle \** | *无* | *释放Handle 指针* |
| *出参* | *返回值* |  | *int* |  | *状态码* |

函数功能

释放模型所占的内存

释放后无法继续推理

### 2.5.8 场景识别支持的类别

 "瀑布",  "雪景",  "草原（草地）",  "水下",  "建筑",  "日出日落",  "天空",  "绿植",  "鲜花",  "夜景",

"自拍", "舞台", "烟花", "烛光", "家居室内", "文本", "二维码", "屏幕", "健身房", "篝火", "多人",

"桥", "蛋糕", "露营", "交通车辆", "卡通", "沙漠", "饮料", "喷泉", "水果", "游乐场", "儿童", "操场",

"球类运动", "街道", "地铁", "超市", "游泳", "蔬菜", "婚礼", "船", "其他动物", "狗", "江河",

"聚餐", "盆栽", "截图", "猫", "微距", "美食", "沙滩", "山峰", "单人"

## 三、技术支持与联系方式

地    址：合肥钛灵信息科技有限公司

联系邮箱：alineye@gravitylink.com