Megvii Face++ SDK

v0.5.0

**变更和修订历史记录**

**文档编号：**

|  |  |  |
| --- | --- | --- |
| **版本号** | **完成日期** | **修改记录** |
| 0.1.0 | 2016-08-24 | 初稿 |
| 0.1.1 | 2016-08-25 | 将查询过期时间的接口移至算法相关的头文件中 |
| 0.2.0 | 2016-09-14 | 增加3DPose检测功能 |
| 0.3.0 | 2016-12-01 | 增加了106点的人脸关键点检测 |
| 0.3.1 | 2016-12-07 | 增加GetAlgorithmInfo接口  增加TRACKING\_SMOOTH的检测方式 |
| 0.3.3 | 2016-12-30 | 增加人脸比对方法 |
| 0.4.1 | 2017-1-12 | 提升了算法性能 |
| 0.4.4 | 2017-2-23 | 提示了算法性能，修复了部分参数不生效的bug |
| 0.4.7 | 2017-3-17 | 增加了 FAST 和 ROBUST 的 Tracking模式 |
| 0.4.7 | 2017-6-27 | 增加Shutdown函数 |
| 0.5.0 | 2017-10-10 | 1. 增加 ResetTrack 接口  2. 增加 GetFaceConfidenceFilter 接口  3. 增加 SetFaceConfidenceFilter 接口  4. 增加 GetJenkinsNumber 接口  5. 增加以下接口替换 GetAlgorithmInfo 接口  5.1 getSDKBundleId  5.2 GetSDKAuthType  5.3 GetAbility |

目录

[Megvii Face++ SDK 1](#_Toc495407916)

[v0.5.0 1](#_Toc495407917)

[目录 2](#_Toc495407918)

[1 通用接口 4](#_Toc495407919)

[1.1 类型 4](#_Toc495407920)

[1.1.1 MG\_RETCODE 4](#_Toc495407921)

[1.1.2 MG常用基础变量 4](#_Toc495407922)

[1.1.3 MG\_POINT 5](#_Toc495407923)

[1.1.4 MG\_RECTANGLE 5](#_Toc495407924)

[1.1.5 MG\_IMAGEMODE 5](#_Toc495407925)

[1.1.6 MG\_FACELANDMARKS 6](#_Toc495407926)

[1.1.7 MG\_3DPOSE 6](#_Toc495407927)

[1.1.8 MG\_FACE 6](#_Toc495407928)

[1.1.9 MG\_ALGORITHMINFO 7](#_Toc495407929)

[2 人脸算法 7](#_Toc495407930)

[2.1 类型 7](#_Toc495407931)

[2.1.1 MG\_FPP\_DETECTIONMODE 7](#_Toc495407932)

[2.1.2 MG\_FPP句柄 8](#_Toc495407933)

[2.1.3 MG\_FPP\_APICONFIG 8](#_Toc495407934)

[2.2 函数 9](#_Toc495407935)

[2.2.1 API函数类型与实例 9](#_Toc495407936)

[2.2.2 CreateApiHandle 9](#_Toc495407937)

[2.2.3 ReleaseApiHandle 10](#_Toc495407938)

[2.2.4 GetJenkinsNumber 10](#_Toc495407939)

[2.2.5 GetApiVersion 10](#_Toc495407940)

[2.2.6 GetApiExpiration 10](#_Toc495407941)

[2.2.7 GetSDKBundleId 11](#_Toc495407942)

[2.2.8 GetSDKAuthType 11](#_Toc495407943)

[2.2.9 GetAbility 11](#_Toc495407944)

[2.2.10 GetDetectConfig 12](#_Toc495407945)

[2.2.11 SetDetectConfig 12](#_Toc495407946)

[2.2.12 Detect 12](#_Toc495407947)

[2.2.13 GetFaceInfo 13](#_Toc495407948)

[2.2.14 GetLandmark 13](#_Toc495407949)

[2.2.15 GetAttribute 14](#_Toc495407950)

[2.2.16 CreateImageHandle 15](#_Toc495407951)

[2.2.17 ReleaseImageHandle 15](#_Toc495407952)

[2.2.18 SetImageData 15](#_Toc495407953)

[2.2.19 ExtractFeature 16](#_Toc495407954)

[2.2.20 GetFeatureData 16](#_Toc495407955)

[2.2.21 FaceCompare 17](#_Toc495407956)

[2.2.22 ResetTrack 17](#_Toc495407957)

[2.2.23 GetFaceConfidenceFilter 18](#_Toc495407958)

[2.2.24 SetFaceConfidenceFilter 18](#_Toc495407959)

[2.2.25 Shutdown 18](#_Toc495407960)

[2.2.26 GetAlgorithmInfo 19](#_Toc495407961)

[3 授权管理 20](#_Toc495407962)

[3.1 类型 20](#_Toc495407963)

[3.1.1 MG\_LICMGR\_DURATION 20](#_Toc495407964)

[3.2 函数 20](#_Toc495407965)

[3.2.1 API函数类型与实例 20](#_Toc495407966)

[3.2.2 GetContext 20](#_Toc495407967)

[3.2.3 SetLicence 21](#_Toc495407968)

[4 SDK的集成 23](#_Toc495407969)

# 通用接口

## 类型

### MG\_RETCODE

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_RETCODE | |
| **名称** | 函数返回值类型 | |
| **所属头文件** | MG\_Common.h | |
| **类型说明** | Enum 枚举类型 | |
| **说明** | 大部分函数返回的类型，用于表示函数运行是否正常。 | |
| **枚举名** | **说明** | **值** |
| MG\_RETCODE\_OK | 正确运行程序 | 0 |
| MG\_RETCODE\_INVALID\_ARGUMENT | 传入了非法的参数 | 1 |
| MG\_RETCODE\_INVALID\_HANDLE | 传入了非法的句柄（handle） | 2 |
| MG\_RETCODE\_INDEX\_OUT\_OF\_RANGE | 传入了非法的索引（index） | 3 |
| MG\_RETCODE\_EXPIRE | SDK已过期，函数无法正常运行 | 101 |
| MG\_RETCODE\_INVALID\_BUNDLEID | 检测到包名与SDK所限制的包名不符 | 102 |
| MG\_RETCODE\_INVALID\_LICENSE | 传入了错误的证书（license） | 103 |
| MG\_RETCODE\_INVALID\_MODEL | 传入了错误的模型（model） | 104 |
| MG\_RETCODE\_FAILED | 算法内部错误 | -1 |

### MG常用基础变量

|  |  |  |
| --- | --- | --- |
| **名称** | 常用基础变量 | |
| **所属头文件** | MG\_Common.h | |
| **类型说明** | 基础变量 | |
| **说明** | 为统一不同平台字长而定义的常用基础变量 | |
| **类型名** | **说明** | **位数** |
| MG\_INT8 | 有符号8位整数 | 8 |
| MG\_INT16 | 有符号16位整数 | 16 |
| MG\_INT32 | 有符号32位整数 | 32 |
| MG\_INT64 | 有符号64位整数 | 64 |
| MG\_UINT8 | 无符号8位整数 | 8 |
| MG\_UINT16 | 无符号16位整数 | 16 |
| MG\_UINT32 | 无符号32位整数 | 32 |
| MG\_UINT64 | 无符号64位整数 | 64 |
| MG\_SINGLE | 单精度浮点数 | 32 |
| MG\_DOUBLE | 双精度浮点数 | 64 |
| MG\_BYTE | 字节 | 8 |
| MG\_BOOL | 布尔 | 32/64 |

### MG\_POINT

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_POINT | |
| **名称** | 坐标点类型 | |
| **所属头文件** | MG\_Common.h | |
| **类型说明** | Struct结构体类型 | |
| **说明** | 表示一个二维平面上的坐标（笛卡尔坐标系）。 | |
| **变量名** | **说明** | **类型** |
| x | 坐标点x轴的值 | MG\_SINGLE |
| y | 坐标点y轴的值 | MG\_SINGLE |

### MG\_RECTANGLE

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_RECTANGLE | |
| **名称** | 图像中平行于量坐标轴的矩形框 | |
| **所属头文件** | MG\_Common.h | |
| **类型说明** | Struct结构体类型 | |
| **说明** | 在图像中表示一个双边平行于坐标轴的矩形框，用(right-left)和(bottom-top)可以计算出矩形的宽和高。 | |
| **变量名** | **说明** | **类型** |
| left | 矩形框最左边的坐标值 | MG\_INT32 |
| top | 矩形框最上边的坐标值 | MG\_INT32 |
| right | 矩形框最右边的坐标值 | MG\_INT32 |
| bottom | 矩形框最下边的坐标值 | MG\_INT32 |

### MG\_IMAGEMODE

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_IMAGEMODE | |
| **名称** | 图像格式类型 | |
| **所属头文件** | MG\_Common.h | |
| **类型说明** | Enum 枚举类型 | |
| **说明** | 表示图像数据格式的枚举类型，支持几种常见的图像格式。 | |
| **枚举名** | **说明** | **值** |
| MG\_IMAGEMODE\_GRAY | 灰度图像 | 0 |
| MG\_IMAGEMODE\_BGR | BGR图像 | 1 |
| MG\_IMAGEMODE\_NV21 | YUV420（nv21）图像 | 2 |
| MG\_IMAGEMODE\_RGBA | RGBA图像 | 3 |
| MG\_IMAGEMODE\_COUNT | 支持图像总数 | 4 |

### MG\_FACELANDMARKS

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_FACELANDMARKS | |
| **名称** | 人脸关键点类型 | |
| **所属头文件** | MG\_Common.h | |
| **类型说明** | Struct结构体类型 | |
| **说明** | 可以记录所有人脸关键点的类型。 | |
| **变量名** | **说明** | **类型** |
| point | 记录关键点的数组 | MG\_POINT[] |

### MG\_3DPOSE

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_3DPOSE | |
| **名称** | 人脸关键点类型 | |
| **所属头文件** | MG\_Common.h | |
| **类型说明** | Struct结构体类型 | |
| **说明** | 记录三维属性的类型 | |
| **变量名** | **说明** | **类型** |
| pitch | 一个弧度，表示物体顺时针饶x轴旋转的弧度。 | MG\_SINGLE |
| yaw | 一个弧度，表示物体顺时针饶y轴旋转的弧度。 | MG\_SINGLE |
| roll | 一个弧度，表示物体顺时针饶z轴旋转的弧度。 | MG\_SINGLE |

### MG\_FACE

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_FACE | |
| **名称** | 记录人脸信息的类型 | |
| **所属头文件** | MG\_Common.h | |
| **类型说明** | Struct结构体类型 | |
| **说明** | 记录了人脸所有属性信息，关键点信息的类型。 | |
| **变量名** | **说明** | **类型** |
| track\_id | 人脸的跟踪标记。如果只对单张图做人脸检测则固定返回-1，否则在不同帧中相同的track\_id表示同一个人脸。每次初始化后track\_id的值为从0开始依此递增。 | MG\_INT32 |
| rect | 人脸在图像中的位置，以一个矩形框来刻画。 | MG\_RECTANGLE |
| points | 人脸关键点信息。 | MG\_FACELANDMARKS |
| confidence | 人脸置信度，为一个0~1之间的浮点数。超过0.5表示这确实是一个人脸。 | MG\_SINGLE |
| Pose | 人脸三维旋转角度。 | MG\_3DPOSE |

### MG\_ALGORITHMINFO

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_ALGORITHMINFO | |
| **名称** | 算法相关的信息 | |
| **所属头文件** | MG\_Common.h | |
| **类型说明** | Struct结构体类型 | |
| **说明** | 记录了算法相关信息的类型 | |
| **变量名** | **说明** | **类型** |
| expire\_time | 一个时间戳，表示过期时间 | MG\_UINT64 |
| auth\_type | SDK的授权类型（联网授权或者非联网授权） | MG\_SDKAUTHTYPE |
| ability | 提供人脸算法的能力  这是一些属性值的bit值的或和，可以参考以MG\_FPP\_ATTR\_开头的宏定义名。 | MG\_UINT64 |

# 人脸算法

## 类型

### MG\_FPP\_DETECTIONMODE

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_FPP\_DETECTIONMODE | |
| **名称** | 人脸检测模式类型 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | Enum 枚举类型 | |
| **说明** | 目前支持单张图片检测和对视频跟踪检测两种检测模式。 | |
| **枚举名** | **说明** | **值** |
| MG\_FPP\_DETECTIONMODE\_NORMAL | 单张图片人脸检测模式 | 0 |
| MG\_FPP\_DETECTIONMODE\_TRACKING | 视频人脸跟踪模式 | 1 |
| MG\_FPP\_DETECTIONMODE\_TRACKING\_SMOOTH | 特殊的视频人脸跟踪模式。此模式下人脸检测与跟踪会更平均的使用CPU计算资源。 | 2 |
| MG\_FPP\_DETECTIONMODE\_TRACKING\_FAST | 牺牲了关键点贴合度，提高了人脸跟踪的速度 | 3 |
| MG\_FPP\_DETECTIONMODE\_TRACKING\_ROBUST | 提高了人脸关键点的贴合度，但跟踪速度比较慢 | 4 |

### MG\_FPP句柄

|  |  |  |
| --- | --- | --- |
| **名称** | 人脸检测相关的句柄 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 句柄（handle） | |
| **说明** | 无 | |
| **类型名** | **说明** | **类型检测** |
| MG\_FPP\_APIHANDLE | 算法句柄 | \_MG\_FPP\_API\* |
| MG\_FPP\_IMAGEHANDLE | 图像句柄 | \_MG\_FPP\_IMAGE\* |

### MG\_FPP\_APICONFIG

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_FPP\_APICONFIG | |
| **名称** | 人脸检测算法配置类型 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | Struct结构体类型 | |
| **说明** | 可以对人脸检测算法进行配置。 | |
| **变量名** | **说明** | **类型** |
| min\_face\_size | 最小检测人脸的尺寸（人脸尺寸一般是指人脸脸颊的宽度），数值越大检测用的耗时越少。 | MG\_UINT32 |
| rotation | 输入图像的重力方向，必须是90的倍数。表示输入图像顺时针旋转rotation度之后为正常的重力方向。 | MG\_UINT32 |
| interval | 在跟踪模式下才有效，表示每隔多少帧进行一次全图的人脸检测。其余时间只对原有人脸进行跟踪。 | MG\_UINT32 |
| detection\_mode | 人脸检测模式。 | MG\_FPP\_DETECTIONMODE |
| roi | 一个矩形框，表示只对图像中roi所表示的区域做人脸检测。（如果人脸在roi中被检测到，且移动到了roi之外的区域，依然可以被跟踪） | MG\_RECTANGLE |
| one\_face\_tracking | 检测人脸时只跟踪单张人脸。 | MG\_BOOL |

## 函数

### API函数类型与实例

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_FACEPP\_API\_FUNCTIONS\_TYPE | |
| **名称** | 人脸算法函数集合 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | Struct结构体类型 | |
| **实例** | mg\_facepp | |
| **说明** | 所有的算法函数都表示为该类型的一个变量，可以用形如：  mg\_facepp.Function(…)  的形式进行调用。 | |
| **函数名** | **说明** | **类型** |
| CreateApiHandle | 创建一个算法句柄 | 函数指针 |
| ReleaseApiHandle | 释放一个算法句柄 | 函数指针 |
| GetApiVerrsion | 获取算法版本信息 | 函数指针 |
| GetApiExpiration | 获取算法的过期时间 | 函数指针 |
| GetDetectConfig | 获取算法当前配置信息 | 函数指针 |
| SetDetectConfig | 设置算法配置信息 | 函数指针 |
| Detect | 检测一张图片 | 函数指针 |
| GetFaceInfo | 获取上一次检测的人脸信息 | 函数指针 |
| GetLandmark | 获取人脸关键点信息 | 函数指针 |
| CreateImageHandle | 创建一个图像句柄 | 函数指针 |
| ReleaseImageHandle | 释放一个图像句柄 | 函数指针 |
| SetImageData | 设置图像信息 | 函数指针 |
| GetAlgorithmInfo | 获取算法相关信息 | 函数指针 |

### CreateApiHandle

|  |  |  |
| --- | --- | --- |
| **函数名** | CreateApiHandle | |
| MG\_RETCODE (\*CreateApiHandle)  (*JNIEnv\* env, jobject jobj,*  const MG\_BYTE \*model\_data, MG\_INT32 model\_length,  MG\_FPP\_APIHANDLE \_OUT \*api\_handle\_ptr) | | |
| **名称** | 创建人脸算法句柄（handle） | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 传入算法模型数据，创建一个算法句柄。 | |
| **变量名** | **说明** | **类型** |
| env *Android Only* | Android jni的环境变量 | JNIEnv\* |
| jobj *Android Only* | Android调用的上下文 | jobject |
| model\_data | 算法模型的二进制数据 | const MG\_BYTE \* |
| model\_length | 算法模型的字节长度 | MG\_INT32 |
| **输出变量名** |  | |
| api\_handle\_ptr | 算法句柄的指针，成功创建后会修改其值 | MG\_FPP\_APIHANDLE \* |

### ReleaseApiHandle

|  |  |  |
| --- | --- | --- |
| **函数名** | ReleaseApiHandle | |
| MG\_RETCODE (\*ReleaseApiHandle) (MG\_FPP\_APIHANDLE api\_handle) | | |
| **名称** | 释放人脸算法句柄（handle） | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 释放一个算法句柄。 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |

### GetJenkinsNumber

|  |  |
| --- | --- |
| **函数名** | GetJenkinsNumber |
| const char\* (\*GetJenkinsNumber)() | |
| **名称** | 获取算法Jenkins版本信息 |
| **所属头文件** | MG\_Facepp.h |
| **类型说明** | 函数 |
| **返回值类型** | **说明** |
| const char\* | 返回SDK打包时的jenkins号 |

### GetApiVersion

|  |  |
| --- | --- |
| **函数名** | GetApiVersion |
| const char\* (\*GetApiVersion)() | |
| **名称** | 获取算法版本信息 |
| **所属头文件** | MG\_Facepp.h |
| **类型说明** | 函数 |
| **说明** | 无 |
| **返回值类型** | **说明** |
| const char\* | 一个字符串表示算法的版本信息。 |

### GetApiExpiration

|  |  |  |
| --- | --- | --- |
| **函数名** | GetApiExpiration | |
| MG\_UINT64 (\*GetApiExpiration)(*JNIEnv\*, jobject*) | | |
| **名称** | 查看算法授权的过期时间 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 无 | |
| **变量名** | **说明** | **类型** |
| env *Android Only* | Android jni的环境变量 | JNIEnv\* |
| jobj *Android Only* | Android调用的上下文 | jobject |
| **返回值** |  | |
| 过期时间戳 | 给出算法授权过期的时间戳。如果没有授权过该算法，则返回0。 | MG\_UINT64 |

### GetSDKBundleId

|  |  |
| --- | --- |
| **函数名** | GetSDKBundleId |
| const char\* (\*GetSDKBundleId)() | |
| **名称** | 获取SDK限制的BundleId |
| **所属头文件** | MG\_Facepp.h |
| **类型说明** | 函数 |
| **返回值类型** | **说明** |
| const char\* | 返回SDK限制的BundleId |

### GetSDKAuthType

|  |  |
| --- | --- |
| **函数名** | GetSDKAuthType |
| MG\_SDKAUTHTYPE (\*GetSDKAuthType) | |
| **名称** | 获取SDK授权类型 |
| **所属头文件** | MG\_Facepp.h |
| **类型说明** | 函数 |
| **返回值类型** | **说明** |
| MG\_SDKAUTHTYPE | 返回SDK授权类型 |

### GetAbility

|  |  |  |
| --- | --- | --- |
| **函数名** | GetAbility | |
| MG\_RETCODE (\*GetAbility)  (const MG\_BYTE\* model\_data, MG\_INT32 model\_length, MG\_ABILITY \*ability\_info) | | |
| **名称** | 获取算法包含的能力 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 无 | |
| **变量名** | **说明** | **类型** |
| model\_data | 算法模型的二进制数据 | const MG\_BYTE\* |
| model\_lenght | 算法模型的字节长度 | MG\_INT32 |
| **输出变量名** |  | |
| ability\_info | 算法能力 | MG\_ABILITY |

### GetDetectConfig

|  |  |  |
| --- | --- | --- |
| **函数名** | GetDetectConfig | |
| MG\_RETCODE (\*GetDetectConfig)  (MG\_FPP\_APIHANDLE api\_handle, MG\_FPP\_APICONFIG \_OUT \*config) | | |
| **名称** | 获取当前算法的配置信息 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 获取算法句柄对应的配置信息。 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |
| **输出变量名** |  | |
| config | 算法配置信息 | MG\_FPP\_APICONFIG \* |

### SetDetectConfig

|  |  |  |
| --- | --- | --- |
| **函数名** | SetDetectConfig | |
| MG\_RETCODE (\*SetDetectConfig)  (MG\_FPP\_APIHANDLE api\_handle, const MG\_FPP\_APICONFIG \*config) | | |
| **名称** | 设置算法配置信息 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 将算法的配置信息设置到算法句柄中。 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |
| config | 算法配置信息 | const MG\_FPP\_APICONFIG \* |

### Detect

|  |  |  |
| --- | --- | --- |
| **函数名** | Detect | |
| MG\_RETCODE (\*Detect)  (MG\_FPP\_APIHANDLE api\_handle, MG\_FPP\_IMAGEHANDLE image\_handle,  MG\_INT32 \_OUT \*face\_nr) | | |
| **名称** | 检测图像中的人脸 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 检测一张图像，并返回检测到的人脸个数。  人脸检测使用的是灰度图，传入灰度的图像数据可以减少算法做图像格式转换的时间。 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |
| image\_handle | 图像句柄 | MG\_FPP\_IMAGEHANDLE |
| **输出变量名** |  | |
| face\_nr | 检测到的人脸个数，人脸以0~face\_nr-1编号。 | MG\_INT32 \* |
| **说明** | Detect函数是在做人脸计算时调用最频繁的函数，也是整个算法SDK提供的最核心的功能。在调用Detect函数时，算法已经完成了对Landmark的计算/跟踪。  在骁龙800（Android）上，在720p的图像上，跟踪检测一张人脸的计算耗时为13~15ms。  在iPhone 6p（iOS）上，在720p的图像上，跟踪检测一张人脸的计算耗时为5~6ms。 | |

### GetFaceInfo

|  |  |  |
| --- | --- | --- |
| **函数名** | GetFaceInfo | |
| MG\_RETCODE (\*GetFaceInfo)  (MG\_FPP\_APIHANDLE api\_handle, MG\_INT32 idx, MG\_FACE \_OUT \*face) | | |
| **名称** | 获取人脸信息 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 通过人脸标号获取人脸信息。 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |
| idx | 人脸编号（人脸以0~face\_nr-1编号） | MG\_INT32 |
| **输出变量名** |  | |
| face | 人脸信息 | MG\_FACE \* |

### GetLandmark

|  |  |  |
| --- | --- | --- |
| **函数名** | GetLandmark | |
| MG\_RETCODE (\*GetLandmark)  (MG\_FPP\_APIHANDLE api\_handle,  MG\_INT32 idx, MG\_BOOL is\_smooth, MG\_INT32 nr,  MG\_POINT \_OUT \*points); | | |
| **名称** | 获取人脸关键点信息 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 可以通过参数控制，获取不同个数的关键点，也可以获取平滑过的关键点。 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |
| idx | 人脸编号（人脸以0~face\_nr-1编号） | MG\_INT32 |
| is\_smooth | 是否需要进行平滑处理。选择平滑处理可以让前后帧关键点相对比较稳定。 | MG\_BOOL |
| nr | 获取的关键点个数，目前只有3种数值是合理的，分别是81点、101点和106点，并且定义了对应常量。  MG\_FPP\_GET\_LANDMARK106=106  MG\_FPP\_GET\_LANDMARK101=101  MG\_FPP\_GET\_LANDMARK81=81 | MG\_INT32 |
| **输出变量名** |  | |
| points | 获取的人脸关键点 | MG\_POINT \* |

### GetAttribute

|  |  |  |
| --- | --- | --- |
| **函数名** | GetAttribute | |
| MG\_RETCODE (\*GetAttribute)  (MG\_FPP\_APIHANDLE api\_Handle, MG\_FPP\_IMAGEHANDLE image\_handle,  MG\_INT32 idx, MG\_INT32 attribute\_mode,  MG\_FACE \_OUT \* face) | | |
| **名称** | 计算一张人脸的属性 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 通过attribute\_mode控制计算哪些人脸属性，将计算的结果写入到face的具体字段当中。 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |
| image\_handle | 图像句柄 | MG\_FPP\_IMAGEHANDLE |
| idx | 人脸编号（人脸以0~face\_nr-1编号） | MG\_INT32 |
| attribute\_mode | 需要计算的属性类型，将需要计算的属性用或（”|”）符号链接，传入即可  MG\_FPP\_ATTR\_POSE3D=0x01 | MG\_INT32 |
| **输出变量名** |  | |
| face | 人脸信息，根据attribute\_mode的不同，会计算对应人脸属性。 | MG\_FACE \* |
| **说明** | GetAttribute函数是在做人脸计算时一个比较重要的函数，其调用也十分频繁。在调用GetAttribute函数时，算法会计算人脸对应的属性，计算耗时会根据计算属性的多少发生变化。  **计算3DPose：**  在骁龙800（Android）上，在720p的图像上，计算一张人脸的3DPose耗时为5~7ms。  在iPhone 6p（iOS）上，在720p的图像上，计算一张人脸的3DPose耗时为2~3ms。 | |

### CreateImageHandle

|  |  |  |
| --- | --- | --- |
| **函数名** | CreateImageHandle | |
| MG\_RETCODE (\*CreateImageHandle)  (MG\_INT32 width, MG\_INT32 height,  MG\_FPP\_IMAGEHANDLE \_OUT \*image\_handle\_ptr) | | |
| **名称** | 创建图像句柄（handle） | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 图像句柄是表示的是一张图像，创建时需要确定其宽和高，且后续不能修改。如果是对连续的视频图像进行图像检测，可以使用同一个图像句柄，而不用反复释放。 | |
| **变量名** | **说明** | **类型** |
| width | 图像的宽 | MG\_INT32 |
| height | 图像的高 | MG\_INT32 |
| **输出变量名** |  | |
| image\_handle\_ptr | 图像句柄的指针，成功创建后会修改其值 | MG\_FPP\_IMAGEHANDLE \* |

### ReleaseImageHandle

|  |  |  |
| --- | --- | --- |
| **函数名** | ReleaseImageHandle | |
| MG\_RETCODE (\*ReleaseImageHandle) (MG\_FPP\_IMAGEHANDLE image\_handle) | | |
| **名称** | 释放图像句柄（handle） | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 释放一个图像句柄。 | |
| **变量名** | **说明** | **类型** |
| image\_Handle | 图像句柄 | MG\_FPP\_IMAGEHANDLE |

### SetImageData

|  |  |  |
| --- | --- | --- |
| **函数名** | SetImageData | |
| MG\_RETCODE (\*SetImageData)  (MG\_FPP\_IMAGEHANDLE image\_handle, const MG\_BYTE \*image\_data,  MG\_IMAGEMODE image\_mode) | | |
| **名称** | 设置图像数据 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 传图的图像数据的内存，需要保证在图像句柄被释放前都是有效的。如果一个图像句柄被反复调用该函数，则以最后一次设置的图像数据为准，且不会再用到上一次设置的数据。 | |
| **变量名** | **说明** | **类型** |
| image\_handle | 图像句柄 | MG\_FPP\_IMAGEHANDLE |
| image\_data | 图像数据，其大小应该有图像大小及格式决定。（如：一张YUV的图，其数据大小应该为witdh\*height\*1.5，且witdh和height必须都是2的倍数） | const MG\_BYTE \* |
| image\_mode | 图像格式 | MG\_IMAGEMODE |

### ExtractFeature

|  |  |  |
| --- | --- | --- |
| **函数名** | ExtractFeature | |
| MG\_RETCODE (\*ExtractFeature)(  MG\_FPP\_APIHANDLE api\_handle,  MG\_FPP\_IMAGEHANDLE image\_handle,  MG\_INT32 idx,  MG\_INT32\* feature\_length\_ptr); | | |
| **名称** | 抽取人脸特征 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 抽取图像中特定人脸的特征，特征以单精度浮点数存储 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |
| image\_handle | 图像句柄 | MG\_FPP\_IMAGEHANDLE |
| idx | 人脸编号（人脸以0~face\_nr-1编号） | MG\_INT32 |
| **输出变量名** |  | |
| feature\_length\_ptr | 存储特征需要的单精度浮点数长度 | MG\_INT32\* |

### GetFeatureData

|  |  |  |
| --- | --- | --- |
| **函数名** | GetFeatureData | |
| MG\_RETCODE (\*GetFeatureData)(  MG\_FPP\_APIHANDLE api\_handle,  MG\_SINGLE \_OUT \*feature\_data,  MG\_INT32 feature\_length); | | |
| **名称** | 获取人脸特征数据 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 在调用 ExtractFeature 后，调用此函数获取特征数据 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |
| feature\_length | 特征的长度，通过 ExtractFeature 函数获得 | MG\_SINGLE |
| **输出变量名** |  | |
| feature\_data | 人脸特征数据，务必保证其内存大小不低于 feature\_length | MG\_SINGLE\* |

### FaceCompare

|  |  |  |
| --- | --- | --- |
| **函数名** | FaceCompare | |
| MG\_RETCODE (\*FaceCompare)(  MG\_FPP\_APIHANDLE api\_handle,  const MG\_SINGLE\* feature\_data1,  const MG\_SINGLE\* feature\_data2,  MG\_INT32 feature\_length,  MG\_DOUBLE \_OUT \*score); | | |
| **名称** | 获取人脸比对的分数 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 传入两个特征，比对两个特征后产生对应分数。传入的特征顺序课交换。  阈值如下：  - 1e-2: 63.07  - 1e-3: 73.43  - 1e-4: 79.79  - 1e-5: 84.02 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |
| feature\_data1 | 参与比对的特征1 | const MG\_SINGLE\* |
| feature\_data2 | 参与比对的特征2 | const MG\_SINGLE\* |
| feature\_length | 特征的长度，通过 ExtractFeature 函数获得 | MG\_INT32 |
| **输出变量名** |  | |
| score | 人脸比对的分数 | MG\_DOUBLE |

### ResetTrack

|  |  |  |
| --- | --- | --- |
| **函数名** | ResetTrack | |
| MG\_RETCODE (\*ResetTrack)(MG\_FPP\_APIHANDLE api\_handle) | | |
| **名称** | 清除当前track模式下的缓存信息 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 快速切换track模式时可能残留上次track的信息，造成关键点错误，此时需要调用此接口 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |

### GetFaceConfidenceFilter

|  |  |  |
| --- | --- | --- |
| **函数名** | GetFaceConfidenceFilter | |
| MG\_RETCODE (\*GetFaceConfidenceFilter)(MG\_FPP\_APIHANDLE api\_handle, MG\_SINGLE \*filter) | | |
| **名称** | 获取算法内对人脸置信度的过滤阈值 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 人脸置信度小于阈值的将不再返回 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |
| **输出变量名** |  | |
| filter | 算法人脸置信度过滤阈值 | MG\_SINGLE |

### SetFaceConfidenceFilter

|  |  |  |
| --- | --- | --- |
| **函数名** | SetFaceConfidenceFilter | |
| MG\_RETCODE (\*GetFaceConfidenceFilter)(MG\_FPP\_APIHANDLE api\_handle, MG\_SINGLE filter) | | |
| **名称** | 设置算法内部人脸置信度过滤阈值 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 人脸置信度小于阈值的将不再返回。 | |
| **变量名** | **说明** | **类型** |
| api\_handle | 算法句柄 | MG\_FPP\_APIHANDLE |
| filter | 算法人脸置信度过滤阈值 | MG\_SINGLE |

### Shutdown

|  |  |
| --- | --- |
| **函数名** | Shutdown |
| MG\_RETCODE (\*Shutdown)() | |
| **名称** | 释放算法占用的内存资源 |
| **所属头文件** | MG\_Facepp.h |
| **类型说明** | 函数 |
| **说明** | 算法在运算是会占用额外的计算资源，需要在所有的算法句柄（handle）释放后再调用此函数 |

### GetAlgorithmInfo

|  |  |  |
| --- | --- | --- |
| **函数名** | GetAlgorithmInfo | |
| MG\_RETCODE (\*GetAlgorithmInfo)  (const MG\_BYTE\* model\_data, MG\_INT32 model\_length,  MG\_ALGORITHMINFO \*algorithm\_info); | | |
| **名称** | 获取算法相关信息 | |
| **所属头文件** | MG\_Facepp.h | |
| **类型说明** | 函数 | |
| **说明** | 读取模型中相关参数，返回当前SDK的所使用的算法的相关信息。 | |
| **变量名** | **说明** | **类型** |
| model\_data | 算法模型的二进制数据 | const MG\_BYTE \* |
| model\_length | 算法模型的字节长度 | MG\_INT32 |
| algorithm\_info | 算法相关信息 | MG\_ALGORITHMINFO |

# 授权管理

## 类型

### MG\_LICMGR\_DURATION

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_LICMGR\_DURATION | |
| **名称** | 单次授权时长类型 | |
| **所属头文件** | MG\_LicenseManager.h | |
| **类型说明** | Enum 枚举类型 | |
| **说明** | 目前只支持单次授权30天或365天，具体收费标准参看官网相关内容。如果自行传入了非这两个值之外的其他值，函数不会运行成功。 | |
| **枚举名** | **说明** | **值** |
| MG\_LICMGR\_DURATION\_30DAYS | 单次授权30天 | 30 |
| MG\_LICMGR\_DURATION\_365DAYS | 单次授权365天 | 365 |

## 函数

### API函数类型与实例

|  |  |  |
| --- | --- | --- |
| **类型名** | MG\_LICENSE\_MANAGER\_API\_FUNCTIONS\_TYPE | |
| **名称** | 授权管理函数集合 | |
| **所属头文件** | MG\_LicenseManager.h | |
| **类型说明** | Struct结构体类型 | |
| **实例** | mg\_licmgr | |
| **说明** | 所有的算法函数都表示为该类型的一个变量，可以用形如：  mg\_licmgr.Function(…)  的形式进行调用。 | |
| **函数名** | **说明** | **类型** |
| GetContext | 获取一个用于授权请求的上下文 | 函数指针 |
| SetLicence | 设置一个许可证，进行授权 | 函数指针 |
| GetExpiration | 查看当前授权过期时间 | 函数指针 |

### GetContext

|  |  |  |
| --- | --- | --- |
| **函数名** | GetContext | |
| MG\_RETCODE (\*GetContext)  (*JNIEnv\* env, jobject jobj,*  MG\_LICMGR\_DURATION duration, const char\* uuid,  const char \_OUT \*\*context\_data, MG\_INT32 \_OUT \*context\_length,  ...) | | |
| **名称** | 获取一个用于授权请求的上下文信息 | |
| **所属头文件** | MG\_LicenseManager.h | |
| **类型说明** | 函数 | |
| **说明** | 可以获取一个同时对一个或多个MegviiCloud的算法进授权的上下文信息。 | |
| **变量名** | **说明** | **类型** |
| env *Android Only* | Android jni的环境变量 | JNIEnv\* |
| jobj *Android Only* | Android调用的上下文 | jobject |
| duration | 申请的授权时长（以当前时间开始计算，向后30或365天） | MG\_LICMGR\_DURATION |
| uuid | 标示不同用户的唯一id，可以为空。  如果uuid有具体意义，则可以享受由MegviiCloud提供的各种统计服务。 | const char\* |
| … | 传入需要授权的算法GetAPIVersion的指针，并以MG\_END\_ARG结束。  例如需要对mg\_facepp算法进行授权，则传入参数为  mg\_licmgr.GetContext(  duration,  uuid,  &context\_data,  &context\_length ,  mg\_facepp.GetApiVersion,  MG\_END\_ARG) | 可变参数 |
| **输出变量名** |  | |
| context\_data | 获取的上下文信息，成功创建后会修改其值 | const char \*\* |
| context\_length | 获取的上下文信息的长度，成功创建后会修改其值 | MG\_INT32 \* |

### SetLicence

|  |  |  |
| --- | --- | --- |
| **函数名** | SetLicence | |
| MG\_RETCODE (\*SetLicence)  (*JNIEnv\* env, jobject jobj,*  const char \*license\_data, MG\_INT32 license\_length) | | |
| **名称** | 设置一个许可证，进行授权 | |
| **所属头文件** | MG\_LicenseManager.h | |
| **类型说明** | 函数 | |
| **说明** | 将GetContext获取的上下文信息，发送给MegviiCloud的授权API，获取license信息后，通过该函数对算法进行授权。  授权完成后可以用GetExpiration函数查看授权结果。  授权请求涉及到设备的网络权限，需要开发者自己完成。授权相关的Web API文档见如下网址：  <https://cloud.megvii.com/documents/5672315> | |
| **变量名** | **说明** | **类型** |
| env *Android Only* | Android jni的环境变量 | JNIEnv\* |
| jobj *Android Only* | Android调用的上下文 | jobject |
| license\_data | 联网获得的license数据 | const char \* |
| license\_length | License数据的长度 | MG\_INT32 |

# SDK的集成

本算法的SDK接口均为C接口，为了方便iOS和Android开发者的使用，我们在GitHub上开源了部分代码，对C接口做了封装。

iOS：

* 人脸算法SDK：https://github.com/FacePlusPlus/MegviiFacepp-iOS-SDK
* 授权管理SDK：https://github.com/FacePlusPlus/MegviiLicMgr-iOS-SDK

Android：

* 人脸算法SDK：https://github.com/FacePlusPlus/MegviiFacepp-Android-SDK
* 授权管理SDK：https://github.com/FacePlusPlus/MegviiLicMgr-Android-SDK