云从科技人脸 SDK 技术白皮书

First printing, July 2016

前言

作为国内最先进的人脸识别技术服务提供商，我们始终坚持聚焦战略，对图像识别基础算法、云平台和智能终端设备等持续进行研发投入，以尖端技术创新和客户需求为驱动，使公司始终处于国际领先水平，引领行业的发展，是我们一直努力的方向。

技术白皮书会为您提供 SDK 基本的开发指南和技术背景介绍，帮助您更好地使用我们的技术服务。

衷心感谢您对我们技术与产品的信任和支持！

修改记录

版本号	拟制/修改日期	主要更改内容
5.0.0	2018.09.09	新版本 SDK 首次撰写
5.1.0	2018.10.23	新增属性封装接口
5.2.0	2018.11.19	红外双目活体接口封装
5.3.0	2018.12.25	新增可见光单目活体接口

目录

1. 技术规格

1.1. 开发语言

SDK 目前提供了标准 C 语言接口，Windows，Linux，Android 平台下接口函数与参数定义完全一致。第四章 SDK 开发文档中给出 C 语言接口的具体介绍。

1.2. 平台支持

SDK 适用于 Windows，Linux，Android 等操作系统，支持 x86，x64，ARMv7a 和 ARMv8a 等架构的处理器，所有系统平台的接口均保持一致。本小节将介绍不同系统环境的推荐配置要求，如下表：

1.2.1. Windows

系统版本	Windows 7 or later 64/32bit
开发环境	Visual Studio 2012
CPU	主频 3.00GHz
内存	8GB

1.2.2. Linux

系统版本	CentOS7.2 64bit / CentOS7.3 64bit
开发环境	GCC 4.8.5
CPU	主频 3.00GHz
内存	8GB

1.2.3. Android

系统版本	arm-v7a or arm-v8a
系统版本	Android 4.0 or later
开发环境	Android SDK r14 or later
CPU	主频 1.80GHz
内存	3GB

1.3. 功能模块

1.3.1. 人脸检测跟踪

人脸检测		说明
输入	1、支持人脸图片	人脸瞳间距>15 像素
	2、支持实时、离线视频流	
	3、支持输入 ROI（人脸检测区域设定）	ROI 为矩形
输出	1、单图多人脸的位置输出	单图人脸数量<50 人
	2、人脸位置坐标（x,y,width,height）	
	3、人脸置信度	
	4、单图多人脸按人脸大小排序	可以根据客户需求按照人脸大小排序、人脸位置排序、人脸置信度排序。
人脸跟踪		说明
输入	1、视频流连续帧输入	
	2、视频流跳帧输入	跳帧时间间隔<1 秒
输出	1、连续帧多人脸的关联输出	关联人脸数量<10 人
	2、重复人脸 ID 合并	同一个人的许多帧人脸给出同 ID 信息

1.3.2. 质量评估

人脸质量评估		说明
输入	1、人脸图片	
	2、人脸图片+关键点坐标	
	3、视频帧人脸	
输出	1、人脸模糊程度	区间[0,1]，越小越模糊；越大越清晰；
	2、人脸光照明暗	区间[0,1]，越小越暗；越大越亮；
	3、人脸正面程度	区间[0,1]，越小越偏；越大越正；
	4、是否戴墨镜	区间[0,1]，越大表示有戴墨镜；
	5、人脸肤色分	区间[0,1]，越小表示黑白，越大代表正常肤色；
	5、人脸综合质量分数	区间[0,1]，越小质量越差；越大越好；

1.3.3. 比对识别

人脸比对		说明
输入	1、人脸图片 A 与人脸图片 B	A、B 均只包含一张人脸
输出	1、人脸图片 A、B 的特征值	
	2、人脸图片 A、B 的比对分数	区间[0,1],越大越相似
人脸识别		说明
输入	1、注册图片列表 A_1~A_n	注册 N 张人脸图片入库
	2、测试人脸图片 B	
输出	1、人脸图片 A_1~A_n、B 的特征值	
	2、人脸图片 B 与 A_1~A_n 的所有相似度分数	区间[0,1],越大越相似
	3、图片 B 与之最相似 A_i	

2. SDK 开发文档

2.1. 数据结构

2.1.1. 输入图像

2.1.1.1. 图像格式

名称：cw_img_form_t

功能：定义图像所采取的编码格式

声明：

```
typedef enum cw_img_form
{
    CW_IMAGE_GRAY8 = 0,
    CW_IMAGE_BGR888,
    CW_IMAGE_BGRA8888,
    CW_IMAGE_RGB888,
    CW_IMAGE_RGBA8888,
    CW_IMAGE_YUV420P,
    CW_IMAGE_YV12,
    CW_IMAGE_NV12,
    CW_IMAGE_NV21,
    CW_IMAGE_BINARY,
} cw_img_form_t;
```

描述：

CW_IMAGE_GRAY8：　单通道灰度图（1 byte / pixel）

CW_IMAGE_BGR888：　三通道 BGR 图（1 byte / pixel / channel）

CW_IMAGE_BGRA8888：四通道 BGRA 图（1 byte / pixel / channel）

CW_IMG_RGB888：　三通道 RGB 图（1 byte / pixel / channel）

CW_IMG_RGBA8888：　四通道 RGB A 图（1 byte / pixel / channel）

CW_IMAGE_YUV420P：　YUV 4:2:0 12bpp（双通道，一个是连续亮度通道，另一个是 U/V 分离交错通道）

CW_IMAGE_YV12：　YVU 4:2:0 12bpp（三通道，一个是连续亮度通道，另两个是连续 U 分量和 V 分量通道）

CW_IMAGE_NV12：　YUV 4:2:0 12bpp（双通道，一个是连续亮度通道，另一个是 U/V 分离交错通道）

CW_IMAGE_NV21：　YUV 4:2:0 12bpp（双通道，一个是连续亮度通道，另一个是 V/U 分离交错通道）

CW_IMAGE_BINARY：　图像文件的二进制流（jpg、bmp、png 等）

2.1.1.2. 图像旋转

名称：cw_img_angle_t

功能：定义图像需要旋转的角度，逆时针方向

声明：

```
typedef enum cw_img_angle {
    CW_IMAGE_ANGLE_0 = 0,
    CW_IMAGE_ANGLE_90,
    CW_IMAGE_ANGLE_180,
    CW_IMAGE_ANGLE_270
} cw_img_angle_t;
```

描述：

CW_IMAGE_ANGLE_0：　不旋转

CW_IMAGE_ANGLE_90：　逆时针旋转 90 度

CW_IMAGE_ANGLE_180：逆时针旋转 180 度

CW_IMAGE_ANGLE_270：逆时针旋转 270 度

2.1.1.3. 图像镜像

名称：cw_img_mirror_t

功能：定义图像的镜像方式

声明：

```
typedef enum cw_img_mirror
{
    CW_IMAGE_MIRROR_NONE = 0,
    CW_IMAGE_MIRROR_HOR,
    CW_IMAGE_MIRROR_VER,
    CW_IMAGE_MIRROR_HV
} cw_img_mirror_t;
```

描述：

CW_IMAGE_MIRROR_NONE： 不镜像

CW_IMAGE_MIRROR_HOR： 垂直镜像

CW_IMAGE_MIRROR_VER： 水平镜像

CW_IMAGE_MIRROR_HV： 垂直和水平镜像

2.1.1.4. 图像

名称：cw_img _t

功能：定义图像的数据和参数

声明：

```
typedef struct cw_img
{
    long long          frameId;
    char*              data;
    int                dataLen;
    int                width;
    int                height;
    cw_img_form_t      format;
    cw_img_angle_t     angle;
    cw_img_mirror_t    mirror;
} cw_img_t;
```

描述：

frameId： 帧号

data： 图像数据

dataLen: 数据长度，BINARY 格式必须设置，其他格式可不设

width： 宽，CW_IMAGE_BINARY 格式可不设，其他格式必须设置

height： 高，CW_IMAGE_BINARY 格式可不设，其他格式必须设置

format： 图像格式

angle： 图像旋转角度

mirror： 图像镜像

2.1.2. 人脸框

名称：cw_facepos_rect_t

功能：定义一个人脸框的位置及其尺寸

声明：

```
typedef struct cw_facepos_rect
{
    int    x;
    int    y;
    int    width;
    int    height;
} cw_facepos_rect_t;
```

描述：

x：　　人脸框左上角的 x 坐标

y：　　人脸框左上角的 y 坐标

width：人脸框的宽度

height：人脸框的高度

2.1.3. 对齐人脸图像

名称：cw_aligned_face _t

功能：定义对齐后的人脸图像

声明：

```
#define CW_ALIGNED_SIZE 128
typedef struct cw_aligned_face
{
    char   data[CW_ALIGNED_SIZE * CW_ALIGNED_SIZE];
    int    width;
    int    height;
    int    nChannels;
} cw_aligned_face_t;
```

描述：

CW_ALIGNED_SIZE：对齐后的人脸图像尺寸

width：　　对齐后的人脸图像宽度

height：　　对齐后的人脸图像高度

nChannels： 对齐后的人脸图像通道

2.1.4. 人脸质量

名称：cw_quality_t

功能：定义人脸的各项质量分

声明：

```
#define FACE_QUALITY_MAX_COUNT 10
typedef struct cw_quality
{
    cw_quality_errcode_t errcode;
    float scores[FACE_QUALITY_MAX_COUNT];
} cw_quality_t;
```

描述：

errcode: 质量分析错误码

scores： 质量分分数项，具体含义（根据数据下标顺序）：

0: 人脸质量总分，0~1.0 之间，越大人脸质量越好，推荐范围 0.65-1.0

1: 清晰度，越大表示越清晰，推荐范围 0.65-1.0

2: 亮度，越大表示越亮，推荐范围 0.2-0.8

3: 人脸角度，左转为正，右转为负

4: 人脸角度，抬头为正，低头为负

5: 人脸角度，顺时针为正，逆时针为负

6: 肤色接近真人肤色程度，越大表示越真实，推荐范围 0.5-1.0

7: 戴黑框眼镜置信度，越大戴黑框眼镜可能性越大，推荐范围 0.0-0.5

8: 戴墨镜的置信分，越大表示戴墨镜的可能性越大，推荐范围 0.0-0.5

2.1.5. 人脸综合信息

名称：cw_face _res_t

功能：定义人脸综合信息，包括人脸来源、人脸 ID、人脸框、关键点、对齐人脸、人脸质量等结果。

声明：

```
typedef struct cw_face_res
{
    long long              frameId;
    int                    detected;
    int                    trackId;
    cw_facepos_rect_t      faceRect;
    cw_aligned_face_t      faceAligned;
    cw_quality_t           quality;
} cw_face_res_t;
```

描述：

frameId：　人脸所在帧号

detected：　0: 跟踪到的人脸；

1: 检测到的人脸;

2: 检测到但不会被进行后续计算(关键点)的人脸;

3: 可能是静态误检框；

4: 大角度人脸;

5: 关键点错误;

6: 不需再处理的人脸;

7: 被估计为低质量人脸

trackId：　跟踪 ID，小于 0 时表示尚未进入跟踪

faceRect：　人脸框的位置和尺寸

faceAligned：　对齐后的人脸图像，只有 detected=1，才有对齐人脸数据

quality：　人脸质量结果，只有 detected=1，才有人脸质量

2.1.6. 错误码说明

2.1.6.1. 函数通用错误码

名称：cw_errcode_t

功能：接口返回的错误码

声明：

```
typedef enum cw_errcode
{
    CW_SDKLIT_OK = 0,

    CW_UNKNOWN_ERR = 20000,
    CW_DETECT_INIT_ERR,
    CW_KEYPT_INIT_ERR,
    CW_QUALITY_INIT_ERR,

    CW_DET_ERR,
    CW_TRACK_ERR,
    CW_KEYPT_ERR,
    CW_ALIGN_ERR,
    CW_QUALITY_ERR,
    CW_EMPTY_FRAME_ERR,
    CW_UNSUPPORT_FORMAT_ERR,
    CW_ROI_ERR,
    CW_UNINITIALIZED_ERR,
    CW_MINMAX_ERR,
    CW_OUTOF_RANGE_ERR,
    CW_UNAUTHORIZED_ERR,
    CW_METHOD_UNAVAILABLE,
    CW_PARAM_INVALID,
    CW_BUFFER_EMPTY
    CW_FILE_UNAVAILABLE,
    CW_DEVICE_UNAVAILABLE,
    CW_DEVICE_ID_UNAVAILABLE,
    CW_EXCEEDMAXHANDLE_ERR,

    CW_RECOG_FEATURE_MODEL_ERR,
    CW_RECOG_ALIGNEDFACE_ERR,
    CW_RECOG_MALLOCMEMORY_ERR,
    CW_RECOG_FEATUREDATA_ERR
    CW_RECOG_EXCEEDMAXFEASPEED,
    CW_RECOG_EXCEEDMAXCOMSPEED,
    CW_RECOG_GROUPSIZE_ERR,
     CW_RECOG_CONVERT_ERR,
     CW_RECOG_NOFACEDET,
```

```
    CW_LICENCE_JSON_CREATE_ERR,
    CW_LICENCE_DECRYPT_ERR,
    CW_LICENCE_HTTP_ERROR,
    CW_LICENCE_MALLOCMEMORY_ERR,
    CW_LICENCE_KEY_DEVICE_ERR,
    CW_LICENCE_KEY_LICENSE_ERR,
    CW_LICENCE_KEY_INSTALL_ERR,
} cw_errcode_t;
```

描述：

CW_SDKLIT_OK：　　0，成功

CW_UNKNOWN_ERR：　　20000，未知错误

CW_DETECT_INIT_ERR：　　20001，初始化人脸检测器失败

CW_KEYPT_INIT_ERR,：　　20002，初始化关键点检测器失败

CW_QUALITY_INIT_ERR：　　20003，初始化跟踪器失败

CW_DET_ERR：　　20004，人脸检测失败

CW_TRACK_ERR：　　20005，人脸跟踪失败

CW_KEYPT_ERR：　　20006，关键点检测失败

CW_ALIGN_ERR：　　20007，人脸对齐失败

CW_QUALITY_ERR：　　20008，人脸质量评估失败

CW_ EMPTY_FRAME_ERR：　　20009，空图像

CW_ UNSUPPORT_FORMAT_ERR：20010，不支持的图像格式

CW_ ROI_ERR：　　20011， ROI 设置错误

CW_UNINITIALIZED_ERR：　　20012，功能句柄尚未初始化

CW_ MINMAX_ERR：　　20013，最小/最大人脸尺寸设置错误

CW_OUTOF_RANGE_ERR：　　20014，数据范围错误

CW_ UNAUTHORIZED_ERR：　　20015，未授权

CW_METHOD_UNAVAILABLE：　　20016，方法无效

CW_PARAM_INVALID：　　20017，参数无效

CW_BUFFER_EMPTY： 20018， 缓冲区空

CW_FILE_UNAVAILABLE： 20019，文件不存在如模型不存在等

CW_DEVICE_UNAVAILABLE： 20020，设备不存在

CW_DEVICE_ID_UNAVAILABLE： 20021，设备 id 不存在

CW_EXCEEDMAXHANDLE_ERR： 20022，超过授权最大数量

CW_RECOG_FEATURE_MODEL_ERR：20023，加载特征识别模型失败

CW_RECOG_ALIGNEDFACE_ERR： 20024，对齐图片数据错误

CW_RECOG_MALLOCMEMORY_ERR：20025，预分配特征空间不足

CW_RECOG_FILEDDATA_ERR： 20026，特征数据错误

CW_RECOG_EXCEEDMAXFEASPEED：20027，超过授权最大提特征速度

CW_RECOG_EXCEEDMAXCOMSPEED 20028，超过授权最大比对速度

CW_RECOG_GROUPSIZE_ERR 20029， 底库 N 超过最大授权数

CW_RECOG_CONVERT_ERR, 20030， 特征转换失败

CW_RECOG_NOFACEDET, 20031， 未检测到人脸

CW_LICENCE_JSON_CREATE_ERR： 20032，Json 操作失败

CW_LICENCE_DECRYPT_ERR： 20033，加密失败

CW_LICENCE_HTTP_ERROR： 20034，HTTP 请求失败

CW_LICENCE_MALLOCMEMORY_ERR：20035，内存分配不足

CW_LICENCE_KEY_DEVICE_ERR： 20036，获取设备文件错误

CW_LICENCE_KEY_LICENSE_ERR： 20037，获取授权文件错误

CW_LICENCE_KEY_INSTALL_ERR： 20038，安装授权文件错误

CW_ATTRI_AGEGENDER_MODEL_ERR ：20039，加载属性模型失败

CW_ATTRI_EVAL_AGE_ERR： 20040 ，年龄识别失败

CW_ATTRI_EVAL_GENDER_ERR： 20041，性别识别失败

CW_ATTRI_EVAL_RACE_ERR： 20042，种族识别失败

2.1.6.2. 质量分析错误码

名称：cw_quality_errcode_t

功能：定义人脸质量分析错误码

声明：

```
typedef enum cw_quality_errcode
{
    CW_QUALITY_OK = 0,
    CW_QUALITY_NO_DATA = 20150,
    CW_QUALITY_ ERROR_UNKNOWN,
} cw_quality_errcode_t;
```

描述：

CW_QUALITY_OK： 质量分数据有效

CW_QUALITY_NO_DATA： 质量分数据无效，原因：尚未检测

CW_QUALITY_ ERROR_UNKNOWN： 未知错误

2.1.7. 函数接口相关

2.1.7.1. 人脸预处理功能开关

名称：OP

功能：定义功能选择开关。根据用户需求，每项功能都可与其他功能进行自由组合

声明：

```
#define CW_OP_DET                0
#define CW_OP_TRACK              2
#define CW_OP_ALIGN              8
#define CW_OP_QUALITY            16
#define CW_OP_ALL                30
```

描述：

CW_OP_DET： 人脸检测开关。基础功能，若不显示开启，系统内部也会隐式开启。此功能返回人脸框信息

CW_OP_TRACK： 人脸跟踪开关。若开启，会赋予人脸一个唯一 ID，并对视频流中的同一人物保持此 ID

CW_OP_ALIGN： 人脸对齐开关。若开启，会计算得到对齐后的人脸图像。否则，对应数据项无效

CW_OP_QUALITY： 人脸质量分评估开关

CW_OP_ALL： 所有开关综合

2.1.7.2. 人脸检测功能参数

名称： cw_det_param_t

功能： 定义功能参数，影响各个子功能的性能和效果

声明：

```
typedef struct cw_det_param
{
    int   roiX;
    int   roiY;
    int   roiWidth;
    int   roiHeight;
    int   minSize;
    int   maxSize;
    const char* pConfigFile;
} cw_det_param_t;
```

描述：

roiX： ROI（感兴趣区域）的左上角 x 坐标。默认为 0

roiY： ROI（感兴趣区域）的左上角 y 坐标。默认为 0

roiWidth： ROI 的宽度。默认为 0，表示输入图像的宽度

roiHeight： ROI 的高度。默认为 0，表示输入图像的高度

minSize： 检测出的最小人脸尺寸。PC 端默认 48，移动端默认 100，值越大检测速度越快

maxSize： 检测出的最大人脸尺寸。PC 端默认 600，移动端默认 400，大于该尺寸的人脸将被忽略

pConfigFile： 内部参数配置文件路径，此参数只能设置(set)

2.2. 人脸预处理接口

2.2.1. 创建检测器句柄

名称：cwCreateDetHandle

功能：创建检测器句柄

声明：

```
CW_FACE_API
void* cwCreateDetHandle(cw_errcode_t* errCode,
                        const char* pConfigFile,
                        const char* pLicence);
```

形参：

errCode：　　成功返回 CW_SDKLIT_OK，失败返回其他 [out]

pConfigFile：　　模型参数配置文件[in]

pLicence：　　授权码（仅用于安卓平台，PC 端传 NULL 即可）[in]

返回值：

检测器句柄

2.2.2. 销毁检测器句柄

名称：cwReleaseDetHandle

功能：释放检测器

声明：

```
CW_FACE_API
void cwReleaseDetHandle(void* pDetector);
```

形参：

pDetector：检测器句柄 [in]

返回值：

无

2.2.3. 获取人脸功能参数

名称：cwGetFaceParam

功能：获取检测器参数

声明：

```
CW_FACE_API
cw_errcode_t cwGetFaceParam(void* pDetector, cw_det_param_t* param);
```

形参：

pDetector：检测器句柄 [in]

param：　　检测器参数 [out]

返回值：

错误码

2.2.4. 设置人脸功能参数

名称： cwSetFaceParam

功能： 设置功能参数，（必须先调用 cwGetFaceParam 在使用此函数）

声明：

```
CW_FACE_API
cw_errcode_t cwSetFaceParam(void* pDetector, const cw_det_param_t* param);
```

形参：

pDetector：检测器句柄 [in]

param：　　参数 [in]

返回值：

错误码

2.2.5. 人脸检测功能操作

名称： cwFaceDetection

功能： 人脸检测跟踪接口

声明：

```
CW_FACE_API
cw_errcode_t cwFaceDetection(void* pDetector,
                             cw_img_t* pFrameImg,
                             cw_face_res_t* pFaceBuffer,
                             int iBuffLen,
                             int* nFaceNum,
                             int iOp);
```

形参：

pDetector：　检测器句柄 [in]

pFrameImg：被检测图像 [in]

pFaceBuffer：存放检测结果的数组 [out]

iBuffLen:　存放检测结果 pFaceBuffer 数组的元素个数[in]

nFaceNum：　实际被检测到的人脸数 [out]

iOp：　操作码[in]

返回值：

错误码

2.2.6. 清除检测跟踪状态信息函数

名称： cwResetDetTrackState

功能：清除检测跟踪状态信息函数

声明：

```
CW_FACE_API
cw_errcode_t cwResetDetTrackState (void* pDetector);
```

形参：

pDetector：检测器句柄 [in]

返回值：

错误码

2.3. 特征提取和比对接口

2.3.1. 创建识别句柄

名称：cwCreateRecogHandle

功能：加载模型文件，创建识别句柄

声明：

```
CW_FACE_API
void* cwCreateRecogHandle(cw_errcode_t* errCode ,
                          const char* pConfigurePath,
                          const char* pLicence,
                          cw_recog_pattern_t emRecogPattern);
```

形参：

errCode：　　错误码 [out]

pConfigurePath：　配置文件路径 [in]

pLicence：　　移动端授权码，PC 端传 0 即可 [in]

emRecogPattern：创建的句柄类型 [in]

返回值：

创建的识别句柄

2.3.2. 销毁识别句柄

名称：cwReleaseRecogHandle

功能：销毁识别句柄

声明：

```
CW_FACE_API
void cwReleaseRecogHandle(void* pRecogHandle);
```

形参：

pRecogHandle：识别句柄 [in]

返回值：

错误码

2.3.3. 获取特征长度

名称：cwGetFeatureLength

功能：获取特征长度

声明：

```
CW_FACE_API
int cwGetFeatureLength(void* pRecogHandle);
```

形参：

pRecogHandle：　识别句柄 [in]

返回值：

特征长度

2.3.4. 提取人脸特征

名称：cwGetFaceFeature

功能：提取人脸特征，一次只能提取一个特征

声明：

```
CW_FACE_API
cw_errcode_t cwGetFaceFeature( void* pRecogandle ,
                               cw_aligned_face_t*  alignedFaces,
                               void* pFeatueData);
```

形参：

pRecogHandle：　识别句柄 [in]

alignedFaces：　对齐人脸数据指针 [in]

pFeatueData：　返回的人脸特征，需要分配足够空间 [out]

返回值：

错误码

2.3.5. 人脸特征比对

名称：cwComputeMatchScore

功能：计算 1 个特征与 N 个特征的相似度,返回的相似度 scores 的个数为 N

声明：

```
CW_FACE_API
cw_errcode_t cwComputeMatchScore (void*        pRecogHandle ,
                                  const void*  pFea1,
                                  const void*  pFea2,
                                  int          iFea2Num,
                                  float*       pScores);
```

形参：

pRecogHandle：　识别句柄 [in]

pFea1：　特征 1，只能是一个特征[in]

pFea2：　特征 2，可以是 N 个特征 [in]

iFea2Num：　特征 2 的个数 [in]

pScores：　返回的相似度分数数组，长度为 iFea2Num，需要预先分配空间 [out]

返回值：

错误码

2.3.6. 人脸图片比对

名称：cwVerifyImageData

功能：比对两个图片中最大人脸特征，获取相似度

声明：

```
CW_FACE_API
cw_errcode_t cwVerifyImageData (void*        pDetector ,
                                const void*  pRecogHandle,
                                cw_img_t*    pFrameImg1,
                                cw_img_t*    pFrameImg2,
                                float        *fScore);
```

形参：

pDetector：　检测句柄 [in]

pRecogHandle：　识别句柄[in]

pFrameImg1：　图片 1 数据 [in]

pFrameImg2：　　图片 2 数据[in]

fScores：　　比对分数 [out]

返回值：

错误码

2.4. 人脸属性接口

2.4.1. 创建年龄段、性别、人种句柄

名称：cwCreateAttributeHandle

功能：加载模型文件，创建年龄、性别、人种属性句柄

声明：

```
CW_FACE_API
void* cwCreateAttributeHandle(cw_errcode_t* errCode,
                              const char* pConfigurePath,
                              const char* pLicence);
```

形参：

errCode：　　错误码 [out]

pConfigurePath：　配置文件路径 [in]

pLicence：　　移动端授权码，PC 端传 0 即可 [in]

返回值：

创建的属性句柄

2.4.2. 销毁属性句柄

名称：cwReleaseAttributeHandle

功能：释放年龄段、性别和人种属性句柄

声明：

```
CW_FACE_API
void cwReleaseAttributeHandle(void* pAttributeHangle);
```

形参：

pAttributeHangle：属性句柄 [in]

返回值：

无

2.4.3. 获取年龄段估计

名称：cwGetAgeEval

功能：获取年龄段估计

声明：

```
CW_FACE_API
cw_errcode_t cwGetAgeEval(void* pAttributeHandle,
                          cw_aligned_face_t* pAlignedFace,
                          int* pAgeGroup,
                          float* confidence);
```

形参：

pAttributeHandle：　属性句柄 [in]

pAlignedFace:　对齐人脸数据

pAgeGroup:　年龄段估计，0　小孩　1 成年人　2 老人

confidence:　置信度，0-1 之间小数，值越大可信度越高

返回值：

错误码

2.4.4. 获取性别估计

名称：cwGetGenderEval

功能：获取性别估计

声明：

```
CW_FACE_API
cw_errcode_t cwGetGenderEval(void* pAttributeHandle,
                             cw_aligned_face_t* pAlignedFace,
                             int* pGender,
                             float* confidence);
```

形参：

pAttributeHandle：　属性句柄 [in]

pAlignedFace:　对齐人脸数据

pGender:　性别估计，0 女性　1 男性

confidence:　置信度，0-1 之间小数，值越大可信度越高

返回值：

错误码

2.4.5. 获取人种估计

名称：cwGetRaceEval

功能：获取人种估计

声明：

```
CW_FACE_API
cw_errcode_t cwGetRaceEval(void* pAttributeHandle,
                           cw_aligned_face_t* pAlignedFace,
                           int* pRace,
                           float* confidence);
```

形参：

pAttributeHandle：　　属性句柄 [in]

pAlignedFace:　　对齐人脸数据

pRace:　　人种估计，0 黑人　1 白人　2 黄人

confidence:　　置信度，0-1 之间小数，值越大可信度越高

返回值：

错误码

2.5. 红外双目活体接口

2.5.1. 创建红外活体检测器

名称：cwCreateNirLivenessHandle

功能：创建红外活体检测器

声明：

```
CW_FACE_API
void* cwCreateNirLivenessHandle(cw_nirliveness_err_t *errCode
                                const char *pNirModelPath,
                                const char *pRecogModelPath,
                                const char *pPairFilePath,
                                const char * pLogPath,
                                const float  fSkinThresh,
                                const char  *pLicence);
```

形参：

errCode　　错误码 [out]

pNirModelPath：　红外活体检测器模型文件 [in]

pRecogModelPath：红外活体识别模型文件[in]

pPairFilePath：　匹配文件路径，只支持 640*480 以及 480*640 分辨率 [in]

pLogPath　　要保存 log 的目录 [in]

fSkinThresh：　肤色阈值，默认 0.35 [in]

pLicence：　　授权码（仅用于安卓平台，PC 端传 NULL 即可） [in]

返回值：

红外活体句柄　　成功返回句柄，失败返回 0

2.5.2. 释放红外活体句柄

名称：cwReleaseNirLivenessHandle

功能：释放红外活体句柄

声明：

```
CW_FACE_API
void cwReleaseNirLivenessHandle(void *pHandle);
```

形参：

pHangle：红外活体句柄 [in]

返回值：

无

2.5.3. 红外活体检测接口

名称：cwFaceNirLivenessDet

功能：红外活体检测，需先进行人脸检测

声明：

```
CW_FACE_API
cw_errcode_t cwFaceNirLivenessDet(void* pHandle,
                                  cw_nirliv_detinfo_t*  pNirLivDetInfo,
                                  Cw_nirliv_res_t*       pNirLivRes);
```

形参：

pHandle：　　红外活体句柄 [in]

pNirLivDetInfo:　　输入的红外和可见光图片及关键点等

pNirLivRes:　　红外活体检测结果，需事先分配内存

返回值：

错误码

2.5.4. 红外活体检测封装接口

名称：cwFaceNirByImageData

功能：判断可见光和红外活体中的最大人脸是否为活体

声明：

```
CW_FACE_API
cw_errcode_t cwFaceNirByImageData( void*            pDetector,
                                   void*            pNirHandle,
                                   cw_img_t*        pImgVis,
                                   cw_img_t*        pImgNir,
                                   cw_nirliv_res_t* pNirLivRes) ;
```

形参：

pDetector：	检测句柄 [in]
pNirHandle:	红外句柄[in]
pImgVis:	可见光图片数据[in]
pImgNir:	红外图片数据[in]
pNirLivRes:	存放红外活体检测结果的数组，需事先分配内存[out]

返回值：

错误码

2.6. 可见光单目活体接口

2.6.1. 创建单目活体检测器

名称：cwCreateAttLivenessHandle

功能：创建单目活体检测器

声明：

```
CW_FACE_API
void* cwCreateAttLivenessHandle (cw_attliveness_err_t *errCode
                                 cw_attliv_mode_t      attDetType,
                                 const char           *pModelPath,
                                 const char           *pLicence);
```

形参：

errCode　　错误码 [out]

attDetType：　　单目活体检测模式 [in]

pModelPath：　　单目活体检测器模型文件路径 [in]

pLicence：　　授权码（仅用于安卓平台，PC 端传 NULL 即可） [in]

返回值：

单目活体句柄　　成功返回句柄，失败返回 0

2.6.2. 释放单目活体句柄

名称：cwReleaseAttLivenessHandle

功能：释放单目活体句柄

声明：

```
CW_FACE_API
void cwReleaseAttLivenessHandle (void *pHandle);
```

形参：

pHangle：单目活体句柄 [in]

返回值：

无

2.6.3. 单目活体检测接口

名称：cwFaceAttLivenessDet

功能：单目活体检测

声明：

```
CW_FACE_API
cw_attliveness_err_t cwFaceAttLivenessDet(void  *pHandle,
                                          cw_att_liv_detinfo_t *pAttDetInfo,
                                          cw_attliv_det_rst_t  *pAttackType);
```

形参：

pHandle：　　单目活体句柄 [in]

pAttDetInfo:　　输入的图片及关键点等信息 [in]

pAttackType:　　攻击类型 [out]

返回值：

错误码

2.6.4. 重置单目活体检测器状态

名称：cwFaceAttLivenessReset

功能：重置单目活体检测器状态

声明：

```
CW_FACE_API
cw_attliveness_err_t cwFaceAttLivenessReset (void* pHandle) ;
```

形参：

pHandle：　　单目活体句柄 [in]

返回值：

错误码

2.7. SDK 版本及授权接口

2.7.1. 获取 SDK 当前版本信息

名称：cwGetSDKVersion

功能：获取 SDK 版本信息

声明：

```
CW_FACE_API
cw_errcode_t cwGetSDKVersion(char* pVersion, int iBuffLen);
```

形参：

pVersion： 版本信息，需事先分配内存 [out]

iBuffLen： 分配的空间大小 [in]

返回值：

错误码

2.7.2. 获取设备码信息

名称：cwGetDeviceInfo

功能：获取设备码信息

声明：

```
CW_FACE_API
cw_errcode_t cwGetDeviceInfo (char* pDeviceInfo, int iBuffLen, int *iUseLen);
```

形参：

pDeviceInfo： 返回的设备唯一码，需预先分配足够空间 [out]

iBuffLen： 分配的空间大小，不少于 200 字节 [in]

iUseLen： 输出的设备唯一码的长度 [out]

返回值：

错误码

2.7.3. 通过网络授权安装 licence

名称：cwInstallLicence

功能：通过网络授权安装 licence，生成 V2C 授权文件回保存在当前目录，用于 windows 和 Linux

声明：

```
CW_FACE_API
cw_errcode_t cwInstallLicence ( const char* sAppKey,
                                const char* sAppSecret,
                                const char* sProductId);
```

形参：

sAppKey：　　授权 AppKey，需要从云从科技获取 [in]

sAppSecret：　　授权 AppSecret，需从云从科技获取

sProductId：　　授权产品 Id，需要从云从科技获取 [in]

返回值：

错误码

2.7.4. 获取移动端授权码

名称：cwGetLicence

功能：获取移动端授权码 licence，PC 端无效

声明：

```
CW_FACE_API
cw_errcode_t cwGetLicence(const char* sAppKey,
                          const char* sAppSecret,
                          const char* sProductId,
                          char*       pLicence,
                          int         iBuffLen,
                          Int*        iUseLen);
```

形参：

sAppKey:　　授权 AppKey，需要从云从科技获取 [in]

sAppSecret：　　授权 AppSecret，需从云从科技获取

sProductId：　　授权产品 Id，需从云从科技获取 [in]

iBuffLen：　　分配字节长度，不低于 300 字节 [in]

pLicence：　　输出 licence 码，需事先分配内存，不低于 300 字节 [out]

iUseLen：　　输出的 licence 码的长度 [out]

返回值：

错误码

3. SDK 授权说明

3.1. 授权

本套 SDK 授权分 PC 端和移动端两种授权模式：PC 端需要安装授权 Ukey 才能正常运行，授权 Ukey 由云从科技提供；移动端需要授权码才能成功创建检测识别句柄，授权码由云从科技提供。

3.2. 并发控制与性能控制

本套 SDK 的所有函数接口**都不是**线程安全的。也即是说，使用同一检测器句柄，在多线程环境下同时进行人脸检测、跟踪、关键点提取、质量评估，会造成未定义错误甚至应用程序崩溃。为了达到并发的目的，您应该为每个线程创建独立的句柄以实现本 SDK 提供的功能，因此您能够创建的句柄总数由授权的并发数决定。

并发数是通过 Ukey 授权（PC 端）或者授权码（移动端）来进行控制，授权的最大并发数由云从科技提供。

性能控制主要包括可创建句柄数，底库 N 的数量，和每秒比对的次数，这三个参数都是通过 Ukey 授权来控制。授权 Ukey 详细性能参数如下图：

版本名称	底库数	句柄数	提特征数	比对数TPS	对应场景
v5.0_101	1	3	2/s	5	（1：1）低配版
v5.0_102	1	3	不限	5	（1：1）高配版
v5.0_200	200	3	不限	3	（1：N）200版
v5.0_5000	5000	3	不限	3	（1：N）5000版
v5.0_20000	20000	3	不限	3	（1：N）20000版

4. SDK 使用说明

4.1. Windows 环境

本套 SDK 的 Windows 版本在 VS2012 上编译，建议 Windows 平台 C++开发采用 VS2012 开发环境。不论使用哪种语言，都需要安装 vs20122 运行库。

4.2. 示例程序 Demo

本套 SDK 主机平台提供了两个 Demo，一个为 MFC 开发，需要使用 opencv，一个为控制台程序，不用 opencv，主要区别在于图片的解码方法。这两个 Demo 均只配置了 Release 环境，包括 32 位和 64 位。用户编译后可以直接运行。

Android 平台也提供了两个 Demo，一个 NDK 开发，目录为 Demo_NDK，直接用 NDK 编译，将编译好的可执行程序连同 so 库，模型文件一起拷贝到手机中即可运行。一个 APP 开发，目录为 Demo_APP，该 Demo 在 jni 层进行了封装，提供的 jni 接口可直接供 java 调用，用户可以根据需要自行封装 jni 层接口和对应的 java 接口。这两个 Demo 都需要云从提供的 licence 才能运行，licence 使用参考句柄创建接口说明。

4.3. 开发建议

关于线程和句柄的使用：建议程序启动时创建好句柄和线程，一个线程对应一个句柄，程序退出时销毁句柄和线程。不要程序里不停的创建和销毁句柄，会造成大量内存和时间的耗用。更不能多个线程使用同一个句柄，会造成无法预料的结果，如果一定要这样用，就必须要加锁。