

阳光人脸识别二次开发包（V3.91）说明书

<青铜版>：本版本为免费正式标准版，许可商用。

本版本发布日期：2015 年 6 月 30 日

作者简介：姓名朱伟，笔名清池，四川乐山人，1974 年生，毕业于现乐山师范学院（大专），财会专业，长年在上海的中小型软件公司打工，从事一线软件开发工作 15 年余。本 SDK 为个人待业期间的业余作品。因于 2011~2015 年期间，在家里的手机通话与上网电脑，长期被某黑客集团窃听窃取信息，且甚至这些信息被我所在公司的上级非法获得，常规办法无法抵御，**特提醒广大软件开发人员，于此留心，注重研发的保密工作，免做无用功。**

第一部分 产品的性能评估，应用说明，问题等。

第二部分 主要是 SDK 函数的使用说明。

<<目录>>

第一部分

- [一. 产品的版权](#)
- [二. 产品的版本](#)
- [三. 产品的重要特点](#)
- [四. 如何使用范例代码](#)
- [五. 如何采模板](#)
- [六. 如何设置相似度阈值](#)
- [七. 图像格式，大小，人脸大小怎样最佳](#)
- [八. 产品的适用人群](#)
- [九. 人脸特征数据的保存](#)
- [十. 如何鉴别两套人脸识别 SDK 的优劣](#)
- [十一. 在 DELPHI 下做二次开发的注意事项](#)
- [十二. 如何正确使用子功能](#)
- [十三. SDK 升级信息](#)
- [十三. 人脸检测性能指标](#)

第二部分

- [一. 输出数据-结构定义](#)
- [二. SDK 调用的-初始化部分](#)
- [三. 人脸识别主函数](#)
- [四. 赠送的 USB 视频函数](#)
- [五. 模板管理部分](#)
- [六. 输出调试变量](#)
- [七. 参数设置](#)
- [八. 活体识别](#)
- [九. 人工定位/第三方接口](#)
- [十. 背光模式下的人脸识别](#)
- [十一. 打瞌睡检测参考代码](#)
- [十二. 多线程范例](#)
- [十三. 人面搜索](#)

附录：

[Ver3.91 版的《人脸识别相似度阈值表》](#)

第一部分

一. 产品的版权

本版本为青铜版(核心算法 Ver3.911)分为两个子版本, 免费的标准版和收费的扩展版, 均可商用, 所有范例代码开源, 只要是基于本 SDK 进行二次开发, 做应用项目和做应用软件产品都行, 且无绑定的发行量限制。但是标准版的最大用户数限制为 1000, 且不提供任何技术服务支持。

本品系国人完全自主知识产权开发, 从底层开始即是自行编码, 未引用任何第三方技术如(人脸识别库/图像识别算法库/开源代码)等资源。更未引用其它公司的人脸识别 SDK 来做外壳调用型封装。

本品是 VC 开发, 但只有极小部分用到了 MFC 函数, 大部分代码均为标准 C++ 编写。

作者已于 2010 年在北京办理了著作权证书: 软件名称: 人脸检测与识别算法软件 登记号: 2010SR030391

注: 因国家软件版权申请现起禁止用 SDK/开发包/系统/模块/库 之类的字眼, 一律用软件两字做结尾。

特别申明: 作者至今未对外公开和销售过源代码, 若有则属黑客非法窃取的盗版。

且本核心技术的发行形式至今仅此一种, 所有将本核心技术改装成形如: OCX 控件, 云计算, 某专用 SDK, WEB 服务等变相的商业运营销售形式, 均属非法。

二. 产品的版本

SunLightFace.exe 是网站发布的演示程序, 用户可以通过这个程序来测试 SDK 的性能。

SunLightFace.dll 是 SDK, 其性能技术指标与演示程序是完全相同的。

本版本全称是:

阳光人脸识别开发包<青铜版>: 本版本为免费正式**标准版**, 许可商用, 无捆绑发行量限制。

适用领域是:

在一般光照条件下的, 同一物理环境下的, 用户主动配合的考勤, 门禁, PC 登录等的识别与验证。也可与 RFID 卡或声纹等进行组合的身份验证。

不适用的领域是:

光线混乱或变化大的环境, 用红外光进行识别; 行进中识别(非主动配合识别); 原始相机照片与现场本人的比对(非同一物理环境)。

三. 产品的重要特点

1. 人脸检测的高正确率, 误检, 漏检很少, 支持多脸 (max=32)。平面旋转高达 60 度, 并带鼻, 嘴定位, 及眼镜判断等功能。
2. 支持双目(双摄像头)/多目的 3D 维度识别, 增加了人脸的特征集, 再次提高识别精度, 并有效防止照片通过。
3. 人脸识别的高精度, 向用户**推荐**的识别阈值不仅能适应光线环境的变化, 具备满足实际应用的识别正确率。带眼镜或头发挡住眉毛都行。(但黑粗边眼镜的识别率相对低些, 即在较好的识别环境下, 黑粗边眼镜仍是 OK 的, 只要看得清眼球, 就对识别率没有任何影响)
4. 在背光模式下(看上去黑黑的一大片), 开启背光识别的开关后, 识别速度会减低, 把识别阈值调低一些, 就仍然能进行准确的识别。
5. 高精度的人眼定位, 对高清大图, 开启二次精定位开关后, 能准确地定位人眼中的瞳孔。准确的人眼定位, 使用户能在此基础上开发出多款有创意的延伸产品。

四. 如何使用范例代码

SunLightFace.dll 是按 WINDOWS API 的标准调用来生成的动态库, 并提供了 VC, C#, JAVA, VB 四类范例代码, 精简扼要, 如果因 VS 版本问题无法打开, 可以用其它文件编辑器打开.CPP, .H, .CS, .frm, .bas 等文件来进行分析。

用户实际做应用项目时, 可以直接粘贴范例代码到自己的工程中运行, 只是注意调节好适当的参数值即可。

注: 要把识别能力调好, 照搬范例代码是不够的, 需要仔细阅读本文档, 做一些参数的设置和调节。

五. 如何采模板

1. 因为是主动配合识别, 最好是采 5 张模板照片 (总量是 10 张), 正面 1 张, 侧面 2 张, 上仰下俯各 1 张, 偏角不宜大 (2-3 度即可)→**微偏**: 因为你在主动配合识别时不会故意地去偏着头识别, 而是正对识别, 但可能无心偏了一点, 这里要求你“微偏”的目的就是为了和你将来可能的“无心偏了一点”对应, 并**都眼看着画面中的自己**, 因为眼球是定准比对的重要依据, 所以无论是采模板还是识别, 都眼看着画面中的自己最佳。

2. 另外余下的 5 张模板照片, 可以空, 用于以后进行动态模板更新 ([参见动态模板更新](#)), 以适应人脸的长期发展变化。当然, 如果要求更稳定的识别效果, 并有充分的时间和配合度, 也可在每个方向上采上两个角度差不大的模板照片, 这样总共就用掉 9 张模板, 为了更好地适应光线, 还可再采不同光线的两套模板, 这样就用掉 18 或 27 个模板。个人模板的总量是可设置的, 请参看后面的设置函数。

注: 如果采的某几张模板照片在角度或光线上完全一样, 这是没有任何意义的, 等于只采了一张。

3. 采模板时, 光线不宜偏亮, 也不宜偏暗, 适中, 并充分地反映出人脸轮廓。这样的标准化模板才能更有效地去兼容那些人脸偏亮或偏暗的识别时段。要让识别更准确, **眼球附近**就必须最大可能地**清晰鲜明**, 同时眼球上最好不要反白光。

4. 由于各个摄像头的焦距不同, 可能对人脸轮廓的拉伸度也不同, 所以, 为达到最好的识别率, **建议**采集与识别完全用一模一样的摄像头。

5. 及时画出采集的人脸模板照片(具体功能函数可参见范例代码), 如果发现其中人眼定位不准确的, **建议**删除该模板重采, 因为这样的模板是对识别无效的。(具体的删除函数下文有述)

6. 也有的公司拿去只采了一个正面的模板就开始用, 但前提是光线环境好, 被识别者自愿配合度高, 打开了动态模板更新开关(至少应设置为保 1 变 1 的动态模板更新)。**注意:** 为了能**长期**地进行好 1C1, 1CN, 请务必打开动态模板更新的开关, 以保证库中的模板能随时间的推移与人脸同步变化。还有一种情况是本人不能亲自来采模板, 只有一张标准照片的情况下, 也可以**只用一张模板照片进行识别**, 因为拍照环境不一致, 故识别阈值应大幅下调。

六. 如何设置相似度阈值?

注: 下面的阈值是 SDK 的, 对于最终用户, 开发人员可以将这些阈值或相似度值“适当加大”输出。

当前版本的人脸检测**推荐**阈值为: 0.55

当前版本的人脸识别 1CN **推荐**阈值为: 0.80 (正确率 %99)

当前版本的人脸识别 1C1 **推荐**阈值为: 0.72 (正确率 %98)

注: 以上阈值仅供参考, 不同的光照环境, 不同的摄像头, 需用户自行总结测试来确定阈值。

比如双目识别时, 侧角度的副摄像头, 阈值就应再调低些。

注意: 要想调到识别效果最佳, 就在于视频或拍照的光线要打好, 人脸肤色既不要太暗黑, 也不要**曝光过度**, **让人的整个眼球看上去很黑, 眼白很白**, (实际高清拍摄的眼球可能有点偏黄橙色, 黄眼仁和黑瞳孔很分明, 但这并非人脸识别实际需要的最好效果, **不需要这种分明**), 总之使**黑眼球在图像上显得鲜明突出**, 不论你如何换摄像头或调光, 总之都要向着这个目标靠近。如果是在户外识别, **建议**要加上遮光罩和镜头滤光片, 以去除强阳光反射的干扰, 目前实测**柔和光线**是最好的。

1. 人脸识别或验证**最好连续进行 2 次以上**, 取最大值, 更可靠些。(如果时间许可)

2. 1C1 由于有个人 ID 的输入或打卡, 所以相似度阈值可略低些。

3. 假定每个 ID 都有 5 个模板, 且都有一张正面的 BMP 文件, 则把所有加入的 ID, 用它们对应的 BMP, 在内部做一遍 2 选的 1CN 相互识别。

这样, 2 选的结果: 会输出两个值, 第一个为张三的照片与张三的模板的相似度, 第二个是与张三最像的人的模板与张三照片的相似度, 我们关注第二个:

求出第二个的平均相似度及最高相似度，再参照 SDK 的**推荐**值来共同设定。

比如：

SDK 的**推荐**值=0.55

全部互识别的第二个结果的平均相似度=0.2

第二个结果的最高相似度=0.41（由于这两个人像）

那么，说明：最终阈值可定为：0.55

虽然目前取 0.43（比 0.41 多 0.02）的阈值也不会出错，但难保外人来识别时不会错，所以还是用**推荐**值好些。

再比如：

SDK 的**推荐**值=0.55

全部互识别的第二个结果的平均相似度=0.3

第二个结果的最高相似度=0.56（由于这两个人非常像）

那么，说明：最终阈值可定为：0.58（高于**推荐**值） 否则：这两个人就可能相互识别错。

七. 图像格式，大小，人脸大小怎样最佳？

1.图像格式可用 BMP,JPG,但最好用 BMP,因为最终分析数据还是 BMP 的。

2.图像大小**推荐**用 320*240（240*320 更好），最大可支持 1280*1280，但大了会使速度减慢。

3.人脸最佳采模板和识别 SIZE 是：眼距 40 像素。（两眼球中心连线的长度称为眼距）

建议：人脸在摄像画面中的占位大小最好是 1/8 左右,居中心(因周边可能有变形),故不宜太大,太小则丢特征,至少确保画面最上方要能现出全部头发,最下方要能现出脖子的底端。

八. 产品的适用人群

仅对人脸识别而言：（即不针对人脸检测）

最适用人群：深色眼球，眉毛与眼睛界限分明，鼻尖处明暗分明。

不适用人群：眼球颜色非常“浅色”的人脸。（除此外，黑黄蓝绿眼球都是可以的）

年纪限制：老少男女均可识别，无头发可，头发挡住眉毛也可。

佩戴限制：戴墨镜不行，戴帽子可以，但不能挡住眼，戴粗黑边眼镜时，识别效果比不戴眼镜稍差，但仍能识别，其它眼镜不影响识别，而且可以戴眼镜采模板，不带眼镜识别，反之亦可。

只要看得清眼球即可，如果镜片反光，使眼球看不见了，这种情形是无法识别的。

九. 人脸特征数据的保存

a) 一个模板大约需要 1MB 的硬盘空间。（**注：**启用数据库压缩后会少用一大半）

其中含有一张档案式人脸照片，可由 SDK 函数读出并画出。

b) 注意查看一个名为 DataBase.ini 的文件，里面可对数据库进行配置。

即可以由用户定制 OLEDB 连接串，目前主要兼容 Access 和 SqlServer 两种数据库。

（此项功能仅供开放式数据库接口的~扩展版用户，标准版暂不支持）

十. 如何鉴别两套人脸识别 SDK 的优劣？

名词解释：

1C1:

1C1 是已知 ID 号,从库中读出模板，和当前图做 1 对 1 的**比较**验证。

1CN:

1CN 则是在库中遍历式**比较**，找出与当前图最相似的前 N 个模板。

1CN 如果识别（比较）出错（把你认成了别人），就意味着 1C1 也会错，即这个误认的人用你的 ID 号就能识别通过，所以，应用 1CN 来比较人脸识别开发包（SDK）的性能，几个人采好模板后，在各种可能性的光线下都能刚好通过的 1CN 阈值，就是“性能比较阈值”，（各家 SDK 的阈值不同，这就相当于让两个 SDK 都处于相同的识别能力水平上。）然后想办法找百人照片，或千人照片，（且一个人只能有一张照片，不能重复，）越多越好，以每张照片一个 ID 的方式加入，从而构成大型人脸模板库，然后再一一用这些照片进行 1CN 的 5 选，当然第 1 名相似度应是 0.99 以上（自身像片），而第 2，3，4，5 个相似度（其他人的像片），应在“性能比较阈值”之下（否则就是实际上的认错人了），而且差距越大越好，然后统计出这 1000 次操作的平均结果，就可以知道哪个 SDK 识别能力强了。

十一. 在 DELPHI 下做二次开发的注意事项：

- a) 因为是 WIN32 API, 所以 delphi 调用应采用 stdcall 方式。
- b) VC 之 CHAR *对应的变量类型是 delphi 的 pchar 类型。
- c) 定义结构时，应选择 record 方式。

十一. 如何正确使用子功能：

用的子功能越多，速度就越慢，对于要求高速度的用户这就成了一个问题。

所以，对于人脸监控和录像的用户，只要做人脸检测即可，不用人眼定位等等。

对于虹膜识别的用户，只要做人脸检测，人眼定位，不用做面膜提取等等。

对于做面膜方面的用户，则无需进行眼镜检测及生成识别模板。

前面的三项，每一项又都是后一项的基础。所有子功能的开关，请参见下面函数部分。

十二. SDK 升级信息

- 1. Ver2.7 新增瞳孔定位功能，适用于高清照片的瞳孔定位或辅助虹膜识别。
- 2. Ver2.8 人脸识别的鲁棒性优化，使能识别过去三年内的照片，识别速度提高 20%。
- 3. Ver2.9 人脸识别的光线自适应性能改良。
- 4. Ver2.92 内存 BUG 的解决及相关操作说明文档的改进。
- 5. Ver2.95 人工定位功能的加入，编译器优化编译，整体速度提高约 15%。
- 6. Ver3.01 人脸检测中曲度特征的加入，眼球定位再次改良。
- 7. Ver3.05 侧脸识别小幅度改良。
- 8. Ver3.38 复杂背景图像的处理速度改进；
三寸小照片的人脸检测、识别性能改进；
人脸检测误识率 FER 的大幅度下降。
- 9. Ver3.51 在背光（逆光）环境下的人脸检测与识别的改进。
 - a. Ver3.68 新增部分人脸识别特征，提高了人脸识别的精度。
 - b. Ver3.86 新增高清摄像头支持等功能函数，改进内部数据处理性能。
 - c. Ver3.91 新增多路视频并发支持，修正了因数据文件超过 2GB 导致无法添加模板的 bug。

“三寸小照片”意义诠释：类似二代身份证照片，宽度小于或等于 160 像素点的图像。Width≤160

十三. 人脸检测性能指标

- 1. 人脸角度范围：上仰 30 度，下俯 30 度，左侧 30 度，右侧 30 度，平面旋转 60 度。
- 2. 光照范围：太阳光，室内光，暗室，逆光。
- 3. 人脸肤色范围：黄色人种，白色人种，棕色人种，黑色人种，女士化妆。

- 4. 其它支持：模糊照片，变色照片，褪色照片，戴眼镜，戴帽。
- 5. 图像大小：最大 1280*1280，推荐 320*240 (此项为 2010 年的老测试标准，用户可自行再测定)，类型：BMP/JPG。
- 6. 人脸大小：最小眼距 8 像素 (开启小脸检测开关) 或 12 像素，最大眼距 160 像素，推荐最佳眼距为 40 像素 (此项为 2010 年的老测试标准，用户可自行再测定)。

眼距是指人脸上两眼中心连线的像素长度，可代表图像中的人脸大小。

下表是本 SDK 在人脸检测中的人脸大小限制：

	图像宽度	最小眼距	最大眼距		图像宽度	最小眼距	最大眼距
1	小于 200 像素	大于 12 像素	小于图像宽度	4	541-800 像素	大于 16 像素	小于 140 像素
2	200-360 像素	大于 16 像素	小于 120 像素	5	801-1200 像素	大于 16 像素	小于 150 像素
3	361-540 像素	大于 16 像素	小于 130 像素	6	1200 像素以上	大于 16 像素	小于 160 像素

对于高大于宽的纵向图像，最大眼距的限制不同：是小于图宽的 1/2 而且小于 200 像素。

- 7. 人脸数量：最多可从一张图像中检出 32 张人脸。
- 8. 人脸检测正确率：(1500 张像片随机测试):99.5%, 其中误识率 0.01%, 漏识率 0.04%。
- 9. 人脸检测速度：图像大小 320*240, CPU:P4 2.4GHZ, 平均速度<200ms/张。
- a. 对眼距 12 像素以下的更小的脸，可启用补充模块做极小脸检测，但这种算法检出的脸做人脸识别无意义。

第二部分

- 注 1：**下文虽仅采用 VC 语言格式进行说明，但仍支持其它语言 (C#, JAVA, VB, DELPHI) 进行二次开发。
- 注 2：**一般地，本说明书的参数值（或返回值）LONG 型的 1 表示真，表示开启，表示开关打开，0 则表示假或关闭。返回值是负数，表示有错发生，可以用 GetLastError 函数取得具体信息。

一. 输出数据-结构定义：

```
//人脸检测的输出结果结构
struct DLL_OUT_FACE_STRUCT
{
    BYTE address;           //无意义，传址的
    LONG eye1_x;
    LONG eye1_y;

    LONG eye2_x;
    LONG eye2_y;           //两眼坐标,要进行了人眼定位才有效

    LONG left;
    LONG top;
    LONG right;
    LONG bottom;           //人脸矩形

    LONG angle;             //人脸平面角度（正面垂直时为 90 度）
    float tally;            //得分 ,100 分制,即人脸的置信度。

    LONG is_small_face;     //是小脸还是大脸，1 为小脸，是启用了小脸检测的结果，小脸不能进行人脸识别
}
```

//当是小脸时，只输出人脸矩形，即只有人脸矩形有效，其它值无效

//下面的值要进行了人眼定位与人脸检测的后期处理才有效

```
LONG skin_color_R;           //采样肤色 COLORREF。(RGB)
LONG skin_color_G;           //采样肤色 COLORREF。(RGB)
LONG skin_color_B;           //采样肤色 COLORREF。(RGB)
LONG skin_hd_bright;          //采样肤色的灰度亮度。
```

```
LONG left_face_len;           //从左眼开始计算的左脸估计长度。
LONG right_face_len;          //从右眼开始计算的右脸估计长度。
float face_width_rely;        //脸宽的信任度[0, 1)。
```

```
LONG nose_x;                  //在原图像中的鼻尖位置。
LONG nose_y;                  //在原图像中的鼻尖位置。
float nose_rely;              //鼻尖位置:可信度[0, 1)。
```

```
LONG month_x;                 //在原图像中的嘴中心位置。
LONG month_y;                 //在原图像中的嘴中心位置。
float month_rely;             //嘴心位置:可信度[0, 1)。
```

```
float glass_rely;             //可能眼镜的置信度[0, 1)。
```

```
LONG eye1_w;                  //人眼的两个眼球的宽与高, 人眼的两个眼球的宽与高, 但要求姿态端正,
LONG eye1_h;                  //定位精度随眼距的增大而增高
LONG eye2_w;
LONG eye2_h;
```

```
LONG CloseEyeBelievable; //闭眼的可能性系数输出, 值域[0, 1000]
```

//判断是睁眼还是闭眼，这个判断需要一定时长的序列连续 CloseEyeBelievable 数据才能统计分析出来。

```
}; //END STRUCT DEF
```

//1CN 的识别输出结构

```
struct DLL_1CN_RECOG_OUT_STRUCT
{
    BYTE address;              //无意义，传址的
    float value;               //相似度值域(0, 1)
    char Template_ID[33];      //模板的 ID
    char TemplateFileName[256]; //模板源文件名称
};
```

二. SDK 调用的-初始化部分

```
LONG __stdcall Initialize(CHAR *userkey);
```

//本 SDK 的全局初始化函数。

//装入 SDK 时最先执行且只执行一次。

//userkey 或 username，是作者发送给用户的一个字符串或密钥，输入正确，其它函数才能正确运行。

//免费版本的 userkey 串就是一通用说明字符串，就在附带的范例代码中，原样复制即可使用。

```
LONG __stdcall zSetDataMode(LONG imode);
```

//数据模式设置： imode=0 为人脸模板数据保存在本机（本地化数据）（不设置则默认是 0）

// imode=1 为 CS 结构或远程数据库，人脸模板数据保存在数据库服务器上。

//设置成功返回值=1，否则返回值=0。

//注：1. 本函数也是一个全局设置，只有紧接放在上一全局初始化函数之后执行才是有效的。

2. 此设置要保持长期一贯性，一经设置值，就不再改变值，否则先后数据的存储位置混乱。

```
LONG __stdcall UnInitialize(); //退出 SDK 时执行且只执行一次的。
```

```
LONG __stdcall GetLastError(LONG OID, CHAR *msg); //取得最后一次失败的原因信息。
```

```
LONG __stdcall CreateOneThreadObject(LONG IS_LOAD_TZLIB=1, CHAR *DataFilePath=NULL);
```

//创建一个人脸识别类的实例对象。

//参数 1 表示是否载入所有人脸特征到内存，只做慢速的 1C1(或只做人脸检测)就不用 LOAD 特征。

// 载入特征会消耗较大的内存，读到内存也要耗费 CPU 时间。

//参数 2 表示指明数据目录，数据与程序可以不在同一目录中，如果在同一目录，就默认为空。

// 数据目录中的主要数据文件是 Template.dat, 里面存放着所有人脸特征。

// 此函数调用后，数据文件将处于（独占的）打开状态。

// 在本 SDK 中，数据目录默认名称是 zData，（程序与数据是可分离的，可自由挂接的）。

// 数据目录可以自由复制、命名和变更位置。

//返回值，LONG 在线程上建立的对象 ID，对象 ID 值域[1, 64], 共可建 64 个对象, 0 表示失败。

//如果是多线程，则此函数必要在线程上运行，即，

//一个对象实例，从头到尾都只能在一个线程上建立，运行，销毁。

```
LONG __stdcall DeleteOneThreadObject(LONG OID); //返回值 0，失败，1，成功
```

//销毁上一条函数建立的对象，回收内存，关闭打开的数据文件。

三. 人脸识别主函数

//人脸定位的输出数组已按从高到低的得分排了序。

```
LONG __stdcall FaceLocate(LONG OID, //第一个函数返回的人脸识别实例对象 ID
    CHAR *FileName, //图像文件名, JPG, BMP
    LONG max_out_nums, //用户要求的最大人脸输出数
    FLOAT Threshold, //人脸置信度阈值, 高于这个阈值才会被输出
    DLL_OUT_FACE_STRUCT *dofs); //人脸输出结构数组
```

//返回值，实际输出的人脸数量

```
LONG __stdcall FaceLocate_BmpData( LONG OID, //第一个函数返回的人脸识别实例对象 ID
    BYTE *BmpData, INT width, INT height, INT bitcount,
    //标准 Windows 位图数据(lpData, 320, 240, 24)
    LONG max_out_nums, //用户要求的最大人脸输出数
    FLOAT Threshold, //人脸置信度阈值, 高于这个阈值才会被输出
    DLL_OUT_FACE_STRUCT *dofs); //人脸输出结构数组
```

//返回值，实际输出的人脸数量，如果一张照片上有 8 张人脸，则返回 8。

//注意：如果 max_out_nums 参数设置为 1，而实际的照片上有两个或更多人脸，则会仍只输出 1 个。

但这个人脸是相似度得分最高的那个，往往这张人脸在照片中面积最大，姿态最端正，五官最清晰。

```
LONG __stdcall FlagFace(
    LONG OID, LONG draw_window_hwnd, LONG order, LONG offset_x=0, LONG offset_y=0);
```


//在目标窗口上标志人脸矩形/眼位置(用户也可自行开发本函数)
 //传入参数分别是: OID, 窗口句柄, 脸的序号(多张脸时第1张脸序号是0),
 //最后两个参数为绘制人脸标志矩形时的偏移像素 offset_x, offset_y。
 //因为目标人脸矩形绘制时, 参照坐标为主窗口的客户区左上角点(0, 0),
 //如果视频窗口或图像显示窗口的左上角点不是(0, 0), 就需要做偏移, 否则就会画偏。

LONG __stdcall FaceLocate_FreeMemory(LONG OID); (用户可自行开发)
 //人脸检测完成后, 会把一张人脸标准化小照片及一些参数保存在人脸输出结构中, 作为人脸识别的数据源,
 //或用于画一些人脸标志, 当利用完毕后, 即可用此命令回收内存。

LONG __stdcall Recog1C1(LONG OID, CHAR *VID, LONG order, FLOAT *value, CHAR *TemplateFileName=NULL);
 LONG __stdcall Recog1C1_Fast
 (LONG OID, CHAR *VID, LONG order, FLOAT *value, CHAR *TemplateFileName=NULL);
 //VID 是指当前检测到的人脸和模板库中的哪一个 ID 的脸做验证。
 //order 是人脸检测中输出的编号, order>=0 order<人脸检测输出人脸数。
 //比如说要对一张照片上检测出的 8 张人脸全都做识别, order 从 0 到 7, 调用 8 次人脸验证或识别函数即可。

//value 是本函数输出的相似度, 值域[0, 1)
 //TemplateFileName 请先预置 256BYTE 的空间, 由于一个 ID 有多个模板,
 //这里是输出与当前脸最像的模板的源图文件名称。这个参数是一个输出参数。

//FAST_1C1 在速度上比 1C1 快并不是绝对的, 当库中模板数越多时 FAST_1C1 会渐渐变慢,
 //比如万人, 或十万人, 或更多, 有可能 FAST_1C1 反而比 1C1 慢。
 //因为 1C1 用的是标准 SQL 数据库查找模式(并比对 TID 对应的所有模板),
 //而 FAST_1C1 用的是内存中逐一检索模式(只比对 TID 中最相似的两三个模板)。

LONG __stdcall Recog1CN(LONG OID, LONG order, LONG max_out_num, DLL_1CN_RECOG_OUT_STRUCT *dros);
 //order 是人脸检测中输出的编号, order>=0 order<输出人脸数。
 //max_out_num 是指输出多少个与 ORDER 相似的人脸, 返回值就是实际输出数
 //dros 是输出结构

四. 赠送的 USB 视频函数(非本 SDK 所包含功能项)

LONG __stdcall UsbVideo_Init(LONG DEVICE_ID, LONG play_window_hwnd, LONG Base_Height);

//视频的播放和捉图的初始化函数

DEVICE_ID, 一般是 0, 也可选 1, 2, 3 等, 多视频设备的情形。这个编号顺序是 WINDOWS 自动分配的。

本 SDK 最多支持 16 路视频的播放和多线程捉图, DEVICE_ID 的范围为[0, 15]。

PLAY_WINDOW_HWND: 是播放视频的子窗口的句柄。其位置由设计者定, 其大小会被本函数修改。

Base_Height: 等于 0, 表示视频播放窗口的 SIZE 自适应设备的默认分辨率。

如果 Base_Height 不等于 0, 那么它就表示实际视频高度。如下, 可设置成 240:

240, 是默认值, 视频以 240P 来播放。也可设置为其它任意值, 如 480, 即 480P 的意思。

为适应软件 UI 界面自由设计, 也可以设置成例如 120, 200, 280, 720.....

视频的宽是从高等比例缩放计算出来的, 确保图像不会被拉伸变形, 所以宽不可设置。

注意: 如果视频设备默认分辨率为 320*240, 请不要设置为 480P(640*480), 这样图像会模糊。

LONG __stdcall UsbVideo_Get_Height(LONG DEVICE_ID); //取得实际视频高度
 LONG __stdcall UsbVideo_Get_Width(LONG DEVICE_ID); //取得实际视频宽度
 LONG __stdcall UsbVideo_CapOneBmp(LONG DEVICE_ID, CHAR *BmpFileName);

//从指定 (DEVICE_ID) 的视频设备上, (阻塞) 捕捉一张 BMP 图像, 高宽就是视频的高宽。

```
LONG __stdcall UsbVideo_EndAll(); //退出软件时在最后代码处调用
```

注 1: 若是多路视频, **建议**多路视频的初始化函数全在主窗口线程中调用, 各路捕捉函数则在各个线程中调用。

注 2: 凡支持 WINDOWS 视频 **WDM 协议** 的设备均可使用上面函数, 并不限于 USB 接口的视频设备。

五. 模板管理部分

```
LONG __stdcall AddFaceTemplate (LONG OID, CHAR *TID, LONG order);
```

//向数据目录下的 Template.dat 文件添加人脸模板。

//TID 是指要加入的模板 ID, 一个模板 ID 可以加入多张模板照片。

//order 是人脸检测结果中的脸序号 (一般取值 0)

//这个函数一般跟在 FaceLocate 函数之后执行。

//**注:** 标准版的用户数量限制为 1000, 即一个数据文件最多只能保存 1000 个人的 TID。

// 超过数量限制函数将返回失败, 需自行想办法动态更换数据目录或采用多线程并发的解决方案。

// 也可购买~扩展版, 采用 SqlServer 的 OLEDB 数据库接口, 无用户数量限制, 支持远程且方便管理模板。

//以下所有删除函数的返回值是删除的记录个数

```
LONG __stdcall DelTemplateA (LONG OID, CHAR *TID, CHAR *template_filename=NULL);
```

//如果没有提供模板照片文件名, 就删除这个模板 ID 对应的所有模板照片, 否则, 只删除那一张。

//模板照片文件名就是 FaceLocate 函数执行时, 传入的照片文件名。

```
LONG __stdcall DelTemplateB (LONG OID, CHAR *TID, LONG BH); //BH 是从 0 开始的, 不从 1 开始
```

//删除这个模板 ID 对应的时序中的第 BH 张模板照片, 也就是按入库的时间先后顺序号来删除。

```
LONG __stdcall DelAllTemplate (LONG OID);
```

//删除模板库的所有模板, 相当于清空模板库。

```
LONG __stdcall UpdateMemory (LONG OID);
```

//因为前面的操作都是对数据库的, 所以要让变更立即有效, 就要读库到内存,

//否则, 变更只有下次重启软件后才有效。

//返回值是当前内存中的模板总数

//如果只做慢速的 1C1, (或只做人脸检测), 则 CreateOneThreadObject (LONG IS_LOAD_TZLIB=0,...),

//让第一个参数=0, 也即不需要载入所有模板到内存, 可以省很多内存。

//同时, 不必以后使用任何的 UpdateMemory, 因为慢速 1C1 不使用载入内存的模板特征库。

//如果只新增了一个模板, 要让它立生效 (对 1C1_FAST 和 1CN), 则调用本函数。但如果是批量增删模板,

//则在大量增删之后, 再调用一次本函数为佳, 本函数会调用一次会占用较多的 IO 时间。

```
LONG __stdcall CountMemoryTidTotaleNums (LONG OID);
```

//统计内存中模板 ID 的总数。(但不是所有模板照片总数, 因为一个 TID (模板 ID) 可能对应多张模板照片)

//为确保本函数有效, 则 CreateOneThreadObject (LONG IS_LOAD_TZLIB=1,...), 即要把模板载入到内存。

```
LONG __stdcall EnumMemoryTid (LONG OID, CHAR TID_Byte_Array[]);
```

//枚举出载入到内存中的所有 TID 字符串, 送到已申请好长度空间的 TID 数组。

//返回值是 TID 的总个数。

//一般先调用 CountMemoryTidTotaleNums 这个函数取得数量, 再定接收数组的大小。

//数组字节大小为: 33 * Count (TID 总数)

//内容 (', '分隔符): tid1, tid2, tid3.....tidn + <C 语言的字符串结束符 0>

//输出顺序为字符串的 ASCII 顺序。

//如果是 UNICODE 模式的 C#, 接收到字节数组后, 需要转换为 unicode 的 String

//最后, 用 Splite 进行 ', '分隔符的字符串切割, 取出所有 TID 字符串。

```
LONG __stdcall EnumTid(LONG OID, CHAR TID_Byte_Array[]);
```

//相似于函数 EnumAllMemoryTid, 只不过来源是数据库, 而不是内存。

```
LONG __stdcall CountTemplateTotalNums(LONG OID, CHAR *TID=NULL);
```

//如果 TID 为空, 是统计数据库中的所有模板照片的总数。

//如果 TID 不为空, 则是统计这个 TID 所对应的模板照片的总数。

```
LONG __stdcall DrawOneTemplatePhoto(LONG OID,
```

```
CHAR *TID, LONG BH, LONG object_window_hwnd,
```

```
LONG start_x=0, LONG start_y=0, LONG IS_FLAG_EYE=1);
```

//此函数要与 CountTemplateTotalNums 函数结合使用, 这样才能知道 TID 对应的照片的张数

//TID 是模板 ID

//BH 是指画这个模板 ID 的第几张模板照片, 按入库的时间先后顺序排列从 0 开始。

//object_window_hwnd 是要画的目标窗口

//start_x, start_y 是模板照片的左上角坐标

//IS_FLAG_EYE 是指是否标出眼球, 如果一个模板照片上的眼球标错, 大可以删除这张模板照片,

//因为可能导致识别出错!

六. 输出调试变量

```
LONG __stdcall GetA(LONG OID, LONG PARA_NAME_ORDER);
```

//下面将返回最近一次的功能函数执行所用的时间: 单位 ms

//调用范例:

```
long usedms=zGetA(OID, de_out_recog_lcn_use_time);
```

//返回识别函数所用的毫秒数(不包含人脸检测的)

```
#define de_out_recog_lcn_use_time 1 //返回 1CN 用时
```

```
#define de_out_recog_lcl_use_time 2 //返回 1C1 用时
```

```
#define de_out_recog_fast_lcl_use_time 3 //返回快速 1C1 用时
```

```
#define de_out_find_face_use_time 4 //返回人脸检测用时
```

```
#define de_out_add_template_use_time 5 //返回添加模板用时
```

```
#define de_out_del_template_use_time 6 //返回任何一种删除模板 函数所用的时间
```

```
#define de_out_update_memory_use_time 7 //返回从库文件中重载模板特征到内存用时
```

//返回更新内存模板特征 函数所用的时间, **注意**: 模板越多, 速度越慢

//下面的输出变量, 是就返回的是 1, 否则返回的是 0

```
#define de_out_is_template_update_happen_in_recog 8
```

//最近的一次人脸识别或认证有无发生动态模板更新。

//如果后续进行的是 1CN 或 FAST_1C1, **建议**可以进行一次 UpdateMemory(), 使当前动态更新立即生效。

//但也要注意 UpdateMemory() 在模板上千后会变得很慢, 相当于重载模板特征到内存。

```
#define de_out_is_happen_backlighting 9 //背光算法 A 的检测结果 (人脸图像是否背光)
```

```
#define de_out_is_happen_backlighting_B 10 //背光算法 B 的检测结果 (人脸图像是否背光)
```

//如果查出发生了背光, **建议**动态降低人脸识别相似度的阈值

七. 参数设置

```
LONG __stdcall SetA(LONG OID, LONG PARA_NAME_ORDER, LONG VALUE); //BOOL 型及整型参数设置
```

```
LONG __stdcall SetB(LONG OID, LONG PARA_NAME_ORDER, FLOAT VALUE); //小数型的参数设置
//PARA_NAME_ORDER 是参数的宏名称（数字编号），用于指明设置的是哪个功能的开关或量。
//VALUE 是要设置的目标值，如果目标值是 BOOL 型及整型，用 SetA 函数，如果目标值是小数型的，
//则请用 SetB 函数。
```

//初始值：是指建立人脸识别对象时就默认已有的设置值，用户可根据实际情况作一些修改。

//调用范例：

```
Long ret=SetA(OID, de_is_auto_backlighting_repair,0); //关闭背光补偿 A 算法
Long ret=SetA(OID, de_is_second_locate_eye_infection_GETFACE,1); //开启面膜提取
```

//**注意**：对于采模板和识别这两种不同情形，在参数设置上最好保持一致。

//**注意**：以下行的**参数值**，作者**建议**按实际应用的需要去改变它们。

//#define de_is_second_locate_eye 5 //注：后面的这个 5，对用户没有意义，不要去管它！

//因为有些子功能不用，开着它们后多花 CPU 时间，使人脸检测和识别的速度变慢。

```
#define de_is_second_locate_eye 5 //进行人眼定位, LONG 型, 初始值 1
```

```
#define de_is_second_locate_eye_infection 6 // LONG 型, 初始值 1
```

//进行人眼定位后的后期计算（如是否戴眼镜的判断），只有上一个开关打开的情况下，本开关才有效。

//如果要进行人脸识别与验证，这两个开关都必要打开。

//如果只是尽可能多地捕捉人脸，这两个开关最好关掉，以节省时间。

//下面的六项子开关，依赖于上一个开关的打开才有效

```
#define de_is_second_locate_eye_infection_GETFACE 6001
```

//LONG 型, 初始值 0, 面膜提取, 只有上一个开关打开的情况下，本开关才有效。

//将会在数据目录下按序生成 PickUpFace_?.bmp 文件，青色点表达眼心坐标。

```
#define de_is_second_locate_eye_infection_GETFACE_Threshold 6002
```

//LONG 型, 初始值 64, 面膜边界切分阈值，但依赖于上一个开关的开启才有效。

```
#define de_is_second_locate_eye_infection_CALC_EYEWHITE 6003
```

//LONG 型, 初始值 0 是否启动眼白运算（若要进行打瞌睡检测）

//输出值：CloseEyeBelievable

实际结果要根据大量的时段统计分析数据来盼定，不能靠一次性的 CloseEyeBelievable。

```
#define de_is_second_locate_eye_infection_FOR_RECOG_OR_GLASS 6004
```

//LONG 型, 初始值 1 是否为的是：人脸识别或眼镜判定, 如果下一步要采模板或识别，此项必要。

```
#define de_is_second_locate_eye_infection_FACE_BORDER 6005
```

//LONG 型, 初始值 1 是否启动人脸边界查找。

```
#define de_is_second_locate_eye_infection_FACE_NOSE_MONTH 6006
```

//LONG 型, 初始值 1 是否启动鼻嘴查找。

```
#define de_vertical_angle_limit 7 //LONG 型, 初始值 60
```

//人脸检测中平面旋转的角度范围

//一般地，监控为 45 度，识别设置为 30 度即可，这样，歪着头也可识别。

//注：最大可设置到 80 度，但人脸检测的速度会相对变慢

想提高速度的用户，**建议**把这个值调到 15~30 度之间，以提高速度。

```
#define de_min_eye_distance 35 //人脸检测的最小眼距限制, LONG, 初值 12
```

```
#define de_max_eye_distance 36 //人脸检测的最大眼距限制, LONG, 初值 200
```

//这个与网站/说明书上的眼距表是交集关系, 不在这个范围内的人脸会被滤掉。

//特别是一些太小的脸，不利于进行正确的人脸识别，最好是通过调节上面的 size “窗口” 过滤掉。

//因为人眼二次重定位算法存在的原因，实际过滤结果和预设值可能存在少量的偏差。
//想提高速度的用户，**建议**把这个值调到 25~100 之间。(具体范围可进行现场测试)
//如人脸考勤，若要求走近了才进行识别，可以改大 de_min_eye_distance 的值，从而取得更好的识别效果。

```
#define de_is_color_filter 10
//LONG 型，初始值 1，肤色过滤，默认打开。
//当图像足够大且纹理背景足够复杂时，会在分析时排除掉图像中的青蓝色区域。
//对于背景简单纹理少的图像，这个开关打开或关闭都没有任何作用。
//图像纹理多（背景复杂），如风景人脸图像，建议打开这个开关，
//从而过滤掉青蓝色区域不进行分析，以提高人脸检测速度。
//图像纹理多（背景复杂），“而且”是在酒吧舞厅或劣质光线下，“人脸变色”，建议关闭这个开关，
//否则人脸检测不正确。
```

```
#define de_color_filter_min_bmp_width 11
//LONG 型，初始值 300，作肤色过滤的最小位图宽，因为对于一些小的位图，没有必要做肤色过滤。
```

```
//下面主要针对导出的 FlagFace 函数（在于在图像或视频上标志人脸外矩形等）
//目前，下面这些定位的精度是有条件的，只有在人脸正，光线良好的条件下才能保证准确度。
#define de_flagface_draw_eye 14 //LONG 型，初始值 1，是否画眼
#define de_is_draw_face_border 15 //LONG 型，初始值 1，是否画脸边界
#define de_is_draw_nose_month 16 //LONG 型，初始值 1，是否标志鼻嘴
#define de_draw_nose_month_rely 17 //FLOAT，初始值 0.3f，画出鼻嘴的最低置信度
#define de_draw_face_width_rely 18 //FLOAT，初始值 0.4f，画出脸宽(脸边界)的最低置信度
#define de_is_have_glass_threshold 19
//FLOAT，初始值 0.65f；画出眼镜标记的最低置信度，无开关，若不画，设置来高于 1.0f。
```

```
//下面是模板管理
//关于动态模板更新：对于 1C1，1CN 识别，如果被识别者是 1 年，2 年，3 年，N 年的长期识别，
//这种长期识别的情况，那么建议一定要打开动态模板更新的开关，使库中模板与现实人脸同步变化。
```

重要范例：下面是开启“保 1 变 1”的动态模板更新

```
zSetA(OID, de_is_template_recog_adding_update, 1); //启动增量更新
zSetA(OID, de_template_limite_num, 2); //个人总模板数 2
zSetA(OID, de_template_scroll_reserve_num, 1); //始终保留那一个第 1 次采的模板不变，
//以后每次识别如果相似度过了 0.85 以上，就自动更新掉第二个模板用当前识别图像。
```

```
#define de_is_template_recog_adding_update 23 //LONG 型，初始值 0，
//识别发生后，相似度过指定阈值，是否将当前人脸添加为模板。增量更新。（推荐）
#define de_is_template_recog_hit_update 24 //LONG 型，初始值 0，
//识别发生后，相似度过指定阈值，是否用当前人脸为模板替换掉与当前人脸最像的被“命中”模板。
//命中更新。命中更新的特点是每个人的总模板数不会增加。
//注：上面两个开关，是由二次开发人员保持其设置为互斥的，不能同时为 1。
```

```
#define de_template_scroll_threshold 25
//FLOAT，初值 0.85f；即人脸识别相似度大于了 85%，增量更新的指定阈值。
//在用户添加模板时，这个值无意义。这个值是出厂默认值，但最终应由用户根据实际情况而定。
#define de_template_hit_threshold 26
```



```
//FLOAT, 初值 0.85f;即人脸识别相似度大于了 85%,命中更新的指定阈值。  
//在识别时,相似度大于了更新阈值,才可以更新模板  
//END 关于模板更新
```

```
#define de_template_limite_num 21//LONG 型, 初始值 10,即每个人最多 10 个模板  
#define de_is_adding_template_scroll 20//LONG 型, 初始值 1  
//在添加模板时,人脸模板是否超过限制量就滚动更新。如果未启动滚动,过量就无法再加入。  
//如果启动了滚动更新,当到达最大数量时,会自动删除掉最早的模板,以维持模板总量不变。
```

```
#define de_template_scroll_reserve_num 22//LONG 型, 初始值 5  
//保持多少个最早的模板, 0, 不保持, 3, 则表示最早的 3 个模板始终不被滚动更新掉。
```

//注: 以下行的参数值, 作者**建议**按实际应用的需要去灵活改变它们。

//1. 一对多的过滤口径(1CN)

```
#define de_1CN_first_filter_reserve_bl 27 //FLOAT, 初值 0.02f  
//1CN 在内存特征中进行初次过滤后的保留比 // (2%)  
//在内存人脸特征的过滤中,只保留 2%的模板来做一一详细比对,这个值改小,会提速,  
//尤其是在千人库,万人库中,但漏识率会增加。
```

//2. 相似度输出值映射

```
#define de_enable_out_result_map 30 //LONG, 初值 1  
//启用的时机,门禁/考勤/身份验证, 目的在于区别出不同的人,结果是相似的人相似度差值也较大。  
//禁用的时机,例如从库中找出与当前人脸照片相似的前 20 名, 目的在于查找与一个人相似的人。  
//结果是相似的人相似度差值较小。  
//此开关的功能:相似度值映射,识别发生后,输出映射转换后的相似度值。  
//禁用它:则输出的是最原始的相似度(差值较小),由用户自行分析后,可进行分段缩放,  
//进行曲线变换等,以便输出最终客户认可和理解的相似度值。  
//作者推荐,有好的光照和硬件,想把识别精度提很高的用户,可关掉它自行设计输出值映射算法。
```

//3. 人眼二次精定位

```
#define de_locate_eye_jll 33 //LONG, 初值 0  
//即当较清晰图像的端正的人脸中的“眼球直径达到 15 个像素”以上时,可以用此算法。  
// (否则即便启用了算法,算法也会自动被跳过)  
//且眼球直径越大,本算法的效果就越好。(眼球直径 $\geq 30$ 而又端正的高清,作者推荐用此算法,  
//以提高眼球中心的定准率)  
//最好是高清图,眼白和眼球要有足够大的对比度,越分明效果越好,反之,则用本算法可能会有负作用。  
//眼球半径的上限是不能超过 40 个像素点。(否则即便启用了算法,算法也会自动被跳过)  
//本算法目的是定位标准圆的圆心,但未必就是瞳孔,用户可自行加入规律性的偏移量进行校正后输出。  
//一般建议关闭这个功能,符合上面的条件,又对眼球中心(瞳孔)的定位精度要求高的 USER,可以开启它。
```

//4. 以下值的参数设置 请参见本说明书中的“背光模式下的人脸识别”部分

```
#define de_BackLight_B 32 //背光算法 B //LONG, 初值 0  
#define de_BackLight_B_xnd 3201 //背光算法 B 对应的细腻度 //FLOAT, 初值 0.09f  
//这个值越小,越细腻,速度越慢
```

```
#define de_high_precision_work 31 //进行高精度工作 //LONG, 初值 0
```

//上面这个开关打开，有助于提高背光算法 B 的人脸识别正确率，但降低识别速度
#define de_high_precision_work_xnd 3101 //上一条对应细腻度 //FLOAT,初值 0.045f
//这个值越小，越细腻，速度越慢

//5. 非主动识别优化

#define de_eanble_eye_extand 34 //LONG,初值 0
//本版本是：主动识别可见光版，不是专门的：非主动识别版本
//非主动识别指：被识别者未注意自己正在被识别着。是一种不配合，不知情的状态。
//所以可能正面对着别处，或人面对镜头但眼睛瞟向别处。
//本开关的作用是针对于非主动识别的**优化**。甚至采模板的行为本身也可以是非主动的。
//开启后，会多用一倍以上的识别时间，对“非主动识别”（眼睛看着别处）可提高识别能力 1-5 个百分点。

//6. 活体识别（人脸不能太小，眼距应大于 40 像素）

#define de_check_eye_hd_change_bl 28 //FLOAT 型, 初值 0.45f;
//眼球的“变化率”必要大于这个值，才认为眨了眼。
//实测当一张照片不动时，静态噪音变化度冲到最高 0.23f，这主要决定于环境光的稳定性和摄像机质量。
//眼球的“亮度变化率”必要大于这个值，才认为眨了眼。
//当视频的对比度，清晰度越高时，为加大活体识别的可靠性，建议调高此值。
//如果明明闭了眼，却没有活动提示，最终超时（活体识别一直返回 0），说明这个值应调小，增大灵敏度
//反之，如果值太小，没动眼也会误认为动了眼（别人的照片抖一下就能过）（活体识别返回 1）。
//本活体识别算法的判断原理和依据是：因为人的眼皮比眼球的亮度值更高。

#define de_check_eye_face_change_Threshold 29 //FLOAT 型, 初值 0.33f;
//这是活体识别的第二个重要参数
//人脸的二值稳定度要大于这个值才行
//人脸的二值稳定度，即只准眼变，其它地方不许可变，被识别者在闭眼时不得晃动头部。
//目的在于防止：“人脸大照片”通过平移以形成闭眼的假相，若照片平移一下可通过，请加大这个参数值。
//但真的人脸明明未平移判断函数却返回-1，则要调低这个参数值。

//注：以下行的参数初始值，作者建议**不要改，除非你做了大量现场数据的实证测试，证明改后综合效果变好。**

#define de_is_auto_backlighting_repair 1 //LONG 型, 初始值 1
//是否开启自动逆光补偿（又名背光补偿 A 算法）功能。
//对于门禁，人脸电脑密码等，可能让摄像头背光（逆光）的地方，即人脸暗，但光线在背后，且很亮，采用。
//**注意**：在背景暗，人脸亮的地方采用，会起很大的负面效果。即正常光线情况下**建议**不采用。
//背光补充的有效区在画面的中下方，就是除掉画面的上 1/3，左 1/3，右 1/3 后的余下部分。
所认为的光源是在画面的上 1/3 处。
#define de_ZW_HALF_UP_LIGHT_LD_MIN_VALUE 1001
//LONG, 初始值 180, 当开启了背光补偿功能功能后，图像上方 1/3 最小亮度要大于这个值才能做背光补偿
#define de_ZW_HALF_UP_LIGHT_LD_MIN_BL 1002
//FLOAT, 初始值 1.8f, 当开启了背光补偿功能功能后，上 1/3 的亮度比中 1/3 的亮度高多少倍才能做背光补偿
#define de_backlighting_repair_base_value 2 //LONG, 初始值 33，最大值 100
//背光补偿:补多少？

#define de_is_green_eye_ball_optimize 3 //是否绿眼球化化, LONG 型, 初始值 0
#define de_is_blue_eye_ball_optimize 4 //是否蓝眼球优化, LONG 型, 初始值 0
//优点是利于这两种色的眼球的定位（有利于人眼定位的精确性，尤其是有非黑眼球人的情况下）

//在有外国人，但同时又没有绿/蓝光照的场合(如舞厅)，可以打开这两个开关以进行性能优化。

```
#define de_automode 8 //LONG 型, 初始值 1
```

//人脸检测自动模式, 自动模式下用的时间可以少些，但在多张脸同时存在时，有较小可能漏捉人脸。

```
#define de_is_run_smalle_face 9 //LONG 型, 初始值 0
```

//极小脸检测是否启动的“全局总开关”，

//当最终用户的要求及格分在 80 分以上时，**建议**关小脸检测，小脸检测无意义，浪费大量时间。

//因其分率只在 50-80 分段，上 70 分的都只是极少数。

//所以此时小脸检测自行关闭，以减少时间占用。

//当用于人脸识别时，**建议**关小脸，小脸检测只在进行高灵敏度的只捉人脸的监控时启用。

```
#define de_face_locate_zdz 12 //FLOAT 型, 初值 0.0783f （第一次查找人脸时的）
```

```
#define de_face_locate_bank_zdz 13 //FLOAT 型, 初值 0.0525f
```

//在人脸检测中，查找人脸内部轮廓的边缘检测阈值。

//有时一张图有人脸却找不到，略改一点儿就找到了，这两个预设值代表作者大多数测试的经验最优值。

//如果用户有条件，如有大量的范例照片，也可以微调上面两个值，来取得最好的全局人脸检测效果。

```
#define de_face_is_use_bank 131 //LONG 型, 初值 1, 是否启动第二次查找。
```

//第二次查找人脸时的，在自动人脸检测模式下，第一次未找到及格人脸，会发动第二次。

八. 活体识别

//下为活体识别的两条函数, 目的在于判别是真人脸还是人脸大照片

```
LONG __stdcall InitCheck(LONG OID, LONG order);
```

//初始化活体识别现场函数, order 是人脸检测结果序, 一般取第一个: 0

```
LONG __stdcall FrameCheck(LONG OID, CHAR *bmpFileName);
```

注意，这里传入的一定要是 BMP 格式的文件，不能用 JPG

帧检测函数，返回 0 时，请再执行，表示还未能判断出来，返回小于 0，表示活体识别不能通过，是作弊。返回 1，表示用户是活体。

具体细节流程请参见 VC 范例代码。(VC_sample 目录下的 P3.CPP 结尾部分, 记事本打开即可)

注意：活体识别与背光算法 B（及高精度处理）不能同时使用。

所以做活动识别，应关掉这两个开关(如果前面哪里有打开过)。

注意：要想活体验证判断准确，就要使光照充分，让眼球黑，让眼皮白。另外还有两个活体识别灵敏度参数可供调节，以适应不同光线条件，总之过于敏感和过于迟钝都不好，请参见本文参数设置部分。

作者在此还有另一种活体识别方法**建议**：用两个摄像头，一主一副，组成一个适当夹角，采两套模板（两个数据目录），双线程做识别，这样就是一个伪 3D 识别，因为本 SDK 具有识别的角度严格性，就图像而言人的正脸和侧脸差别是巨大的，所以本质上成了一个人，“两张脸”，特征集增加了。可排除大照片通过的情况，同时提高了识别的可靠性。要两个通道的相似度都在阈值之上后，才算通过，而两个通道则可能因硬件和角度条件的不同，采用不同的阈值，且应用多线程进行并发识别，具体实现可参考我的 **C#双目识别范例代码**。

九. 人工定位/第三方接口

人工定位：对于送来的照片，导入模板库后，进行模板照片查看时（请参见 zDrawOneTemplatePhoto 函数），发现人脸或人眼的定位错误，可以删除这个模板，（这件事应为软件操作员进行）用 PHOTOSHOP 或画笔等软件对原照片进行加工（这里一定要先转成 24 位色的 BMP 进行），在人脸的两个眼中心各画一个十字架进行标定，十字架的两线宽为 1，两线的长均应至少大于 3 个像素，最好是能有 9 个像素长（最长不能到眉），然后保存成 24 位色的 BMP 图像文件，对于这样的“已标定了人脸两眼心的 BMP 文件”，二次开发人员即可进行以下函数调用：

```
LONG __stdcall MakeFaceDataByHand(LONG OID, CHAR *FileName, LONG flag_color, LONG minlinelen);
```

//生成人工定位的人脸数据，可用于加入模板或被识别。

//第一个参数是线程 ID，

//第二个参数是文件名，是“已标定了人脸两眼心的 24 位色的 BMP 文件”

//第三个参数是十字架的两条线的颜色

//线的颜色最好是照片中不存在的颜色，这样可以让标定达到唯一性。

//最后一对 FF 是指：R(红分量)

//倒数第二对 FF 是指：G(绿分量)

//倒数第三对 FF 是指最后一对 FF 是指：B(蓝分量)

//和做网页的那个#FFFFFF 意义一样，但顺序刚好相反。

//如：0x00ffff00 表示青色

//第四个参数是最小线长，如果画的线小于这个长度，将无法定位到。长度单位是像素。

注：一次只能从一张照片中定位一张脸。

人脸在图像中的保留大小，最好是和本 SDK 的 zDrawOneTemplatePhoto 函数的输出图像的大小保持相同。

如果觉得麻烦，可以裁剪得略大些，就是宜大不宜小。

人工定位眼球来加入模板的 VC 范例代码：

(常规参数设置请参见范例代码，此处不再赘述。)

```
LONG ret=MakeFaceDataByHand(OID,"d:\\bmp_zt_flag\\6.bmp",0x00ffffff,5); //白色线
if(ret==1)
{
    AddFaceTemplate(OID,"张三",0); //加入模板库
    FaceLocate_FreeMemory(OID); //回收人脸检测内存
}
else ::AfxMessageBox("没有找到人工定位的两个十字");
```

人工定位眼球来进行识别的 VC 范例代码：

```
LONG ret1=MakeFaceDataByHand(OID,"d:\\bmp_zt_flag\\4.bmp",0x00ff00ff,3); //紫色线
if(ret1==1)
{
    LONG ret2=Recogn1CN(OID,0,5,rout); //一对多识别，5 选
    //第二个参数为 0，表示识别第一张脸，因为人工定位能且只能产生一张脸的数据
    .....//其它代码
    FaceLocate_FreeMemory(OID); //回收人脸检测内存
}
else ::AfxMessageBox("没有找到人工定位的两个十字");
```

第三方接口：有的二次开发用户已开发出了，或已拥有了，用于人脸检测和人眼定位的算法或软件，已经能从图片中准确找出人脸并定位人眼中心，那么可以通过编程的方式，对含人脸的图像做处理，实现和上述人工操作过程相同的**眼心标定**后，生成新的 BMP 文件，从而跳过本 SDK 的人脸检测和人眼定位，直接生成人脸数据（用 MakeFaceDataByHand 函数），最后调用本 SDK 的采模板函数或识别函数。

十. 背光模式下的人脸识别(因**光线不足**导致图像上人脸偏暗的情形同样适用背光 B 算法)

人脸背光的定义：人脸正对摄像头，但背对主强光源，画面中人脸发黑。

从而引发 SDK 找不到人脸或人脸识别率低的问题。

下面是解决方案：

1. 背光 A 的背光发生判断依据是，图像的上方亮，中部暗，判为背光。

zSetA(0ID, de_is_auto_backlighting_repair,1); //打开背光开关，默认就是打开的。

2. 背光 B 的背光发生判断依据是，原来找不到人脸，但进行背光处理后能找到，说明发生了背光。

(实测这个效果优于背光 A 的效果)

zSetA(0ID, de_BackLight_B,1); //打开背光 B 算法，默认是关闭的。

背光 B 模式开启后，定位速度会变慢，如果背光 B 细腻度值调得越小，则速度变得更慢。

背光 B 算法实际上是被动触发启动的，即当图片上未找到至少 1 个及格的人脸时。

3. 推荐开启背光 B 模式的应用场合：

- a) 批量照片的人脸检测，已确定了，照片中肯定至少有一张人脸存在的情况。
- b) 背光环境(或光线明显不足环境)下的 1C1, 1CN.
- c) 一半时间光线好，一半时间背光的 1C1, 1CN.

4. 禁用背光 B 算法的应用场合：

- a) 视频监控的高速度人脸检测，或批量照片的人脸检测，但不确定每帧或每张照片中一定有人脸。
- b) 光线充足，肯定没有背光的环境下进行 1C1, 1CN.
- c) 如果要进行活体识别，则不能开启背光 B.
- d) 如果要进行打瞌睡检测，则不能开启背光 B.
- e) 要求高速度进行的 1C1 和 1CN.

5. 如果识别环境是长期处于背光模式下，光线很差，但又不计较识别的速度，那下面这个高精度开关有助于提高背光模式的识别率：

zSetA(0ID, de_high_precision_work,1); //增强背光下识别人脸的能力，默认是关闭的。

高精度开启后，定位速度会变慢，如果它的细腻度值越小，则速度变得更慢。

光线好的环境，这个开关一定要关掉；若背光情况只是偶尔发生，也不建议启用这个功能。

6. 背光算法的范例代码：

(常规参数设置请参见范例代码，此处不再赘述。背光 A 算法因为用时少，默认就是打开的)

Float fz=0.72f; //定义相似度阈值

zSetA(0ID1, de_BackLight_B,1); //开背光 B 算法开关

//zSetA(0ID1, de_high_precision_work,1); //开背光 B 算法的高精度处理

//上面开启背光 B 算法和提高背光识别能力的高精度算法，以上只要做一次即可。

//**注意：**对于采模板和识别，上面的开关设置要保持一致性，否则性能大降！

//例如采模板时，开启了背光，但识别时又关闭，或相反。

```
int ret=FaceLocate( 0ID1,          //第一个函数返回的人脸识别实例对象 ID
                   "temp.bmp",      //图像文件名，JPG,BMP
                   1,                //用户要求的最大人脸输出数
                   55,               //人脸置信度阈值，高于这个阈值才会被输出
                   ofs);             //人脸输出结构数组
if(ret<=0)return; //未找到人脸，返回，下面是找到了的情况

//FlagFace(0ID1, (LONG) this->m_hWnd, 0); //在窗口上的图像上标出人脸方框等标记
LONG is_bk=0;
Is_bk += ZGetA(0ID1, de_out_is_happen_backlighting); //是否发生了背光 A?
Is_bk += ZGetA(0ID1, de_out_is_happen_backlighting_B); //或是发生了否背光 B?
float out_xsd=0;
Recog1C1(0ID1, "1001", 0, &out_xsd); //开始进行 1C1 识别
float cur_Fz=fz;
if(is_bk>0) cur_fz -= 0.12f; //注意：这里根据是否背光，动态调节相似度阈值
If(out_xsd>=cur_fz) {pass 伪代码!} //当输出的相似度大于了阈值，认为通过
FaceLocate_FreeMemory(0ID1); //回收人脸检测内存
```


十一. 打瞌睡检测参考代码

第一步：开关量设置

```
//下面打开眼白检测功能
zSetA(OID1, de_is_second_locate_eye_infection_CALC_EYEWHITE, 1);
//下面关闭背光 B 与高精度开关(下面两个是默认就是关着的，假如你前面打开了，下面就要关掉)
zSetA(OID1, de_BackLight_B, 0); //注意：打瞌睡检测要求高速度，所以此两个慢速度必关！
zSetA(OID1, de_high_precision_work, 0);
```

第二步：定时器中循环执行的代码(周期 500ms)

```
gm->dx_capture_one_file(&gm->m_dxcap, "temp.bmp"); //捉一张图
int ret=FaceLocate(OID1, "temp.bmp", 1, ofs, cur_sel_threshold); //找一张脸
if(ret<=0) return; //没找到脸
if(ofs[0].tally>=60 && 人脸眼距<=50) //要求人脸得分够高，眼距(人脸面积)足够大
zw_showmsgwindow("提示", "请把脸靠近摄像头一些.", 3000);

long v=ofs[0].CloseEyeBelievable; //读出值
FaceLocate_FreeMemory(OID1); //回收人脸检测内存
//注：需要开发人员做的事，就是把这个值放入一个序列中做时序上的折线统计图分析
//根据其折线图变化规律，来判断眼的睁闭状态。 调试模式时，这里可以绘制折线图。
```

十二. 多线程范例

下为在 VC 中进行多线程操作的范例代码：

- 注意：**
1. CreateOneThreadObject 应在新线程中进行。
一个对象实例，从头到尾都只能在一个线程上建立，运行，销毁。
 2. 普通 1c1 和人脸检测线程不要 LOAD 特征库。(FAST_1C1, 1CN 需要 LOAD 特征库)

```
LONG OID1, OID2, OID3; //各个线程的人脸实例 ID

UINT thr1(LPVOID);
UINT thr3(LPVOID);
UINT thr2(LPVOID);

void CP6::OnBnClickedButton1()
{
    // TODO: 在此添加控件通知处理程序代码

    AfxBeginThread(thr1, NULL, THREAD_PRIORITY_NORMAL);
    AfxBeginThread(thr2, NULL, THREAD_PRIORITY_NORMAL);
    AfxBeginThread(thr3, NULL, THREAD_PRIORITY_NORMAL);
}

BOOL CP6::OnInitDialog()
{
    CPropertyPage::OnInitDialog();
```

Initialize(); //注意：此函数必需运行且只运行一次,且只能在窗口主线程中进行。

```
return TRUE; // return TRUE unless you set the focus to a control
// 异常: OCX 属性页应返回 FALSE
}
```

UINT thr1(LPVOID)

```
{
    DLL_1CN_RECOG_OUT_STRUCT rout[10]; //假设最大 10 选
    DLL_OUT_FACE_STRUCT ofs[32];
    OID1=CreateOneThreadObject(1, "D:\\zData1");
    while(true)
    {
        Sleep(10);

        int ret=FaceLocate( OID1,           //第一个函数返回的人脸识别实例对象 ID
                           "temp.bmp",      //图像文件名, JPG, BMP
                           1,                //用户要求的最大人脸输出数
                           60,              //人脸置信度阈值, 高于这个阈值才会被输出
                           ofs);            //人脸输出结构数组

        // FlagFace(OID1, (LONG) this->m_hWnd, 0); //在窗口上画出人脸矩形等标志

        Recog1CN(OID1, 0, 1, rout);

        CString lxx;
        lxx.Format("OID:%d xsd:%0.4f id:%s \n", OID1, rout[0].value, rout[0].Template_ID );
        TRACE(lxx);
        FaceLocate_FreeMemory(OID1); //回收人脸检测内存
    }
    return 1;
}
```

UINT thr2(LPVOID)

```
{    DLL_OUT_FACE_STRUCT ofs[32];

    DLL_1CN_RECOG_OUT_STRUCT rout;
    OID2=CreateOneThreadObject(1, "D:\\zData2");
    while(true)
    {
        Sleep(10);

        int ret=FaceLocate( OID2,           //第一个函数返回的人脸识别实例对象 ID
                           "d:\\111.bmp",   //图像文件名, JPG, BMP
                           1,                //用户要求的最大人脸输出数
                           50,              //人脸置信度阈值, 高于这个阈值才会被输出
                           ofs);            //人脸输出结构数组
    }
```

```

// FlagFace(OID2, (LONG) this->m_hWnd, 0); //在窗口上画出人脸矩形等标志

Recog1CN(OID2, 0, 1, &rout);

TRACE3("OID:%d xsd:%0.4f id:%s \n", OID2, rout.value, rout.Template_ID);
FaceLocate_FreeMemory(OID2); //回收人脸检测内存
}
return 1;
}

UINT thr3(LPVOID)
{
    DLL_OUT_FACE_STRUCT ofs[32];
    DLL_1CN_RECOG_OUT_STRUCT rout;
    OID3=CreateOneThreadObject(1, "D:\\zData3");
    while(true)
    {
        Sleep(10);

        int ret=FaceLocate( OID3, //第一个函数返回的人脸识别实例对象 ID
            "d:\\bmp\\2. bmp", //图像文件名, JPG, BMP
            1, //用户要求的最大人脸输出数
            50, //人脸置信度阈值, 高于这个阈值才会被输出
            ofs); //人脸输出结构数组

        // FlagFace(OID3, (LONG) this->m_hWnd, 0); //在窗口上画出人脸矩形等标志

        Recog1CN(OID3, 0, 1, &rout);

        TRACE3("OID:%d xsd:%0.4f id:%s \n", OID3, rout.value, rout.Template_ID);

        FaceLocate_FreeMemory(OID3); //回收人脸检测内存
    }
    return 1;
}

//下为调试模式下的输出结果
//OID:1 xsd:0.7916 id:102
//OID:2 xsd:0.4155 id:104
//OID:3 xsd:0.2432 id:101

```

十三. 人面搜索

用途: 在海量的人脸照片中 (可能无人照, 也可能有合影), 找出与目标人脸最相似的前 n 张照片。

目前人面搜索的推荐阈值为: 0.5f

编程思路: 在空库中, 导入目标人脸的一张或几张有代表性的模板照片。

然后可参考下面的思路进行开发:

```

DLL_OUT_FACE_STRUCT ofs[32];
DLL_1CN_RECOG_OUT_STRUCT rout;
OID1=CreateOneThreadObject(1, "D:\\zData");

zSetA(OID1, de_BackLight_B, 1); //建议打开背光 B 算法, 可处理背光照片
//zSetA(OID1, de_high_precision_work, 1);
//打开背光 B 的高精度开关, 有助于背光图片的识别, 但速度会变慢很多

//枚举出目录中的所有文件名, 然后在一个大循环中做以下操作:
for(i=0; i<总文件数; i++)
{
    //1. 对每一张照片做多人脸定位
        int facenum=FaceLocate( OID1, //第一个函数返回的人脸识别实例对象 ID
            照片文件名, //图像文件名, JPG, BMP
            32, //用户要求的最大人脸输出数
            55, //人脸置信度阈值, 高于这个阈值才会被输出
            ofs); //人脸输出结构数组
    //2. 对每一张照片做多人脸识别
        for(int k=0; k<facenum; k++) //facenum 是上面人脸定位中发现的人脸总数
        {
            int ret=Recog1CN(OID1, k, 1, &rout); //进行 1CN 的 1 选(用 1C1 更好)
            if(ret>0 && rout.value>=指定阈值)
            {
                //这里做选出动作的编程, 当前照片上发现目标人脸!
                break; //本张照片不再查下一张脸了
            }
        }
        FaceLocate_FreeMemory(OID1); //回收人脸检测内存
} //总文件数循环

```

附录:

Ver 3.91 版的《人脸识别相似度阈值表》

相似度	正确率	识别能力	错认可能性	适应环境
0.87f	99.75%	+	+	理想环境
0.84f	99.5%	++	++	较理想环境, 比如说镜头增加了滤光片, 消除了不良反射光线的影响
0.80f	99%	+++	+++	光线, 镜头角度变化较小
0.75f	98.5%	++++	++++	光线, 角度有日常变化 (推荐值, 若背光, 此值应再下降 0.1 左右)
0.72f	98%	+++++	+++++	光线, 镜头角度变化较大
0.70f	97%	++++++	++++++	
0.68f	96%	+++++++	+++++++	

0.66f	95%	+++++++	+++++++	
0.60f	90%	+++++++	+++++++	
0.55f	85%	+++++++	+++++++	近似/疑似的，相似度大多分布在 0.5f 附近，并在 0.4f-0.6f 之间

(测试人数 400 人，每人一张正面模板)

要了解更多，请参见网站：<http://www.sunlightface.com/pro/>

作者邮箱：sunface@vip.163.com

——全文结束——