

ArcSoftFace

C# Demo 说明文档

目录

1. 简介.....	1
1.1 运行环境.....	1
1.2 系统要求.....	1
1.3 开发工具.....	1
1.4 环境要求.....	1
1.5 支持的颜色空间格式.....	1
1.6 产品功能简介.....	1
1.6.1 人脸检测/人脸追踪	1
1.6.2 年龄检测.....	1
1.6.3 性别检测.....	1
1.6.4 人脸识别.....	2
1.6.5 RGB 活体检测	2
1.6.6 IR 活体检测.....	2
1.6.7 图像质量检测.....	2
1.6.8 口罩检测.....	2
1.6.9 额头区域检测.....	2
2. 快速上手.....	2
3. 接入指南.....	3
3.1 示例代码.....	3
3.1.1 引擎在线激活.....	3
3.1.2 引擎离线激活.....	3
3.1.3 初始化引擎.....	3
3.1.4 人脸检测.....	4
3.1.5 口罩检测.....	4
3.1.6 图像质量检测.....	4
3.1.7 提取特征.....	4
3.1.8 人脸比对.....	5
3.1.9 RGB 活体检测	5
3.1.10 IR 活体检测.....	5
3.1.11 额头区域检测.....	5
3.2 通用方法.....	6
3.2.1 从 Bitmap 中读取 BGR 数据	6
3.2.2 从 Bitmap 中读取 GRAY 数据	6
4. 常见问题.....	6
4.1 常见问题问答.....	6
4.2 其他帮助.....	7

1. 简介

1.1 运行环境

Windows 平台

最低硬件配置

Intel® Core™ i5-2300@2.80GHz 或者同级别芯片

推荐硬件配置

Intel® Core™ i7-4600U@2.1GHz 或者同级别芯片

1.2 系统要求

Windows7 及以上

1.3 开发工具

VS2013 以上版本、USB 摄像头

1.4 环境要求

.Net Framework 4.5.1 以上

Microsoft Visual C++ 2013 Redistributable 环境包

1.5 支持的颜色空间格式

支持图像的颜色空间格式：BGR24、ASVL_PAF_GRAY

1.6 产品功能简介

1.6.1 人脸检测/人脸追踪

通过 `ASFDetectFaces` 或 `ASFDetectFacesEx` 方法从图片中检测人脸信息，获取图片中人脸框的个数、位置坐标信息和角度信息。不同的使用场景，初始化时使用不同模式：

1. 图片检测模式（适用于静态图片识别）：`ASF_DETECT_MODE_IMAGE`
2. 视频检测模式（适用于摄像头预览，视频文件识别）：`ASF_DETECT_MODE_VIDEO`

1.6.2 年龄检测

对图片中对应的人脸图片信息数据进行年龄检测。对应方法：通过 `ASFProcess` 方法从图片中检测人脸信息，通过 `ASFGetAge` 方法获取年龄检测结果。

1.6.3 性别检测

对图片中对应的人脸图片信息数据进行性别检测。对应方法：通过 `ASFProcess` 或 `ASFProcessEx` 方法从图片中检测人脸信息，通过 `ASFGetGender` 方法获取性别检测结果。

1.6.4 人脸识别

通过 `ASFFaceFeatureExtract` 或 `ASFFaceFeatureExtractEx` 方法从图片中提取人脸特征信息，通过人脸识别 SDK 中人脸比对的方法：`ASFFaceFeatureCompare`，对两个特征值进行比较，通过返回的相似度判断两个人是否是一个人。

1.6.5 RGB 活体检测

对图片中对应的人脸图片信息数据进行活体检测。对应方法：通过 `ASFProcess` 或 `ASFProcessEx` 方法从图片中检测人脸信息，通过 `ASFGetLivenessScore` 方法获取活体检测结果。

1.6.6 IR 活体检测

对图片中对应的人脸图片信息数据进行活体检测。对应方法：通过 `ASFProcess_IR` 或 `ASFProcessEx_IR` 方法从图片中检测人脸信息，通过 `ASFGetLivenessScore_IR` 方法获取活体检测结果。

1.6.7 图像质量检测

对图片中相应的人脸框区域进行图像质量检测。对应方法：通过 `ASFImageQualityDetectEx` 方法获取图像质量检测结果。

1.6.8 口罩检测

对图片中对应的人脸图片信息数据进行口罩检测。对应方法：通过 `ASFProcess` 或 `ASFProcessEx` 方法从图片中检测人脸信息，通过 `ASFGetMask` 方法获取口罩检测结果。

1.6.9 额头区域检测

对图片中对应的人脸图片信息数据进行额头区域检测。对应方法：通过 `ASFProcess` 或 `ASFProcessEx` 方法从图片中检测人脸信息，通过 `ASFGetFaceLandMark` 方法获取额头区域检测结果。

2.快速上手

1. 安装 VS2013 环境安装包（`vcredist_x86_vs2013.exe`），确认本地 .NET Framework 版本
2. 从虹软开发者中心官网(<https://ai.arcsoft.com.cn/ucenter/user/userlogin>) 下载 ArcFace 4.0 SDK 包(x86 或 x64，平台请根据项目编译环境选择)并解压
3. 将 libs 中的文件拷贝到工程 bin 目录的对应平台的 Debug 或 Release 目录下
4. 将对应 APP_ID、SDK_KEY、ACTIVE_KEY 替换 App.config 文件中对应内容
5. 在 Debug 或者 Release 中选择配置管理器，选择对应的平台
6. 按 F5 启动程序
7. 点击“注册人脸”按钮增加人脸库图片
8. 点击“选择识别图”按钮增加人脸图片
9. 点击“开始匹配”按钮进行比较
10. 根据下面文本框查看相关信息

3.接入指南

3.1 示例代码

3.1.1 引擎在线激活

```
retCode = FaceEngine.ASFOnlineActivation(appId, is64CPU ? sdkKey64 : sdkKey32, is64CPU ?
    activeCode64 : activeCode32);
```

3.1.2 引擎离线激活

```
string deviceInfo;
//获取设备信息
retCode = imageEngine.ASFGetActiveDeviceInfo(out deviceInfo);
if (retCode != 0)
{
    MessageBox.Show("获取设备信息失败, 错误码:" + retCode);
}
else
{
    File.WriteAllText("ActiveDeviceInfo.txt", deviceInfo);
}
//官网执行离线激活操作, 将生成的离线授权文件,再执行以下操作
string offlineActiveFilePath = "离线激活文件路径";
//离线激活
retCode = imageEngine.ASFOfflineActivation(offlineActiveFilePath);
```

3.1.3 初始化引擎

图片模式引擎初始化

```
//初始化引擎
DetectionMode detectMode = DetectionMode.ASF_DETECT_MODE_IMAGE;
//Video 模式下检测脸部的角度优先值
ASF_OrientPriority videoDetectFaceOrientPriority = ASF_OrientPriority.ASF_OP_ALL_OUT;
//Image 模式下检测脸部的角度优先值
ASF_OrientPriority imageDetectFaceOrientPriority = ASF_OrientPriority.ASF_OP_ALL_OUT;
//人脸在图片中所占比例, 如果需要调整检测人脸尺寸请修改此值, 有效数值为 2-32
int detectFaceScaleVal = 16;
//最大需要检测的人脸个数
int detectFaceMaxNum = 5;
//引擎初始化时需要初始化的检测功能组合
int combinedMask = FaceEngineMask.ASF_FACE_DETECT | FaceEngineMask.ASF_FACERECOGNITION | FaceE
ngineMask.ASF_AGE | FaceEngineMask.ASF_GENDER | FaceEngineMask.ASF_FACE3DANGLE;
//初始化引擎, 正常返回值为 0
retCode = imageEngine.ASFInitEngine(detectMode, imageDetectFaceOrientPriority,
    detectFaceScaleVal, detectFaceMaxNum, combinedMask);
```

视频模式初始化

```
DetectionMode detectModeVideo = DetectionMode.ASF_DETECT_MODE_VIDEO;
int combinedMaskVideo = FaceEngineMask.ASF_FACE_DETECT | FaceEngineMask.ASF_FACERECOGNITION;
retCode = videoEngine.ASFInitEngine(detectModeVideo, videoDetectFaceOrientPriority,
```

```
detectFaceScaleVal, detectFaceMaxNum, combinedMaskVideo);
```

初始化时要设置需要用的算法功能。

3.1.4 人脸检测

使用人脸检测功能需要在初始化引擎时将人脸检测方法类型 (`FaceEngineMask.ASF_FACE_DETECT`) 做初始化, 将图片文件 (图片文件一般为 RGB 格式图像数据) 作为参数传入 `FaceEngine.ASFDetectFaces()` 或 `FaceEngine.ASFDetectFacesEx()` 的人脸检测方法即可, 示例代码如下:

```
//人脸检测
MultiFaceInfo multiFaceInfo = new MultiFaceInfo();
faceEngine.ASFDetectFacesEx(image, out multiFaceInfo);
```

3.1.5 口罩检测

口罩检测需要在初始化引擎时设置口罩检测算法功能。使用 `FaceEngine.ASFProcessEx()` 或者 `FaceEngine.ASFProcess()` 方法检测人脸信息, 然后通过 `FaceEngine.ASFGetMask()` 方法获取口罩检测结果, 示例代码如下:

```
MaskInfo maskInfo = new MaskInfo();
//人脸信息处理
retCode = faceEngine.ASFProcessEx(image, multiFaceInfo, FaceEngineMask.ASF_MASKDETECT);
if (retCode == 0)
{
    //获取 IR 活体检测结果
    retCode = faceEngine.ASFGetMask(out maskInfo);
}
```

3.1.6 图像质量检测

图像质量检测需要在初始化引擎时设置图像质量检测算法功能, 当人脸检测完成后, 将图像和人脸检测结果、口罩检测结果传入 `FaceEngine.ASFImageQualityDetectEx()` 方法来提取人脸的图像质量, 示例代码如下:

```
//图像质量检测
float confidenceLevel;
retCode = faceEngine.ASFImageQualityDetectEx(image, multiFaceInfo, out confidenceLevel, faceIndex, isMask);
```

3.1.7 提取特征

提取特征功能需要在初始化引擎时将人脸识别功能类型初始化 (`FaceEngineMask.ASF_FACERECOGNITION`), 将图像、人脸检测结果、口罩检测结果作为参数传入 `FaceEngine.ASFFaceFeatureExtract()` 或 `FaceEngine.ASFFaceFeatureExtractEx()` 方法来提取人脸特征信息, 示例代码如下:

```
//特征提取
FaceFeature faceFeature = new FaceFeature();
retCode = faceEngine.ASFFaceFeatureExtractEx(image, multiFaceInfo, registerOrNot, out faceFeature, faceIndex, isMask);
```

3.1.8 人脸比对

人脸比对功能是通过对比两个人脸特征信息，返回两者的相似程度。通过人脸检测，提取特征后，通过 `FaceEngine.ASFFaceFeatureCompare()` 的人脸比对方法对比两个人脸特征信息，获取它们的相似度，示例代码如下：

```
float similarity = 0f;
int retCode = videoRGBImageEngine.ASFFaceFeatureCompare(feature, imagesFeatureList[i],
    out similarity);
```

3.1.9 RGB 活体检测

RGB 活体检测功能是通过 RGB 图像检测图像中的人是否为活体。使用 `FaceEngine.ASFProcessEx()` 或者 `FaceEngine.ASFProcess()` 方法检测 RGB 图像的人脸信息，然后通过 `FaceEngine.ASFGetLivenessScore()` 方法获取 RGB 活体检测结果，示例代码如下：

```
LivenessInfo livenessInfo = new LivenessInfo();
//人脸信息检测
retCode = faceEngine.ASFProcessEx(image, multiFaceInfo, FaceEngineMask.ASF_LIVENESS);
if (retCode == 0)
{
    //获取活体检测结果
    retCode = faceEngine.ASFGetLivenessScore(out livenessInfo);
}
return livenessInfo;
```

3.1.10 IR 活体检测

IR 活体检测功能是通过红外摄像头获取人脸检测数据，根据检测数据判断图片中是否有活体。使用 `FaceEngine.ASFProcess_IR()` 或 `FaceEngine.ASFProcessEx_IR()` 方法检测 IR 图像的人脸信息，然后通过 `FaceEngine.ASFGetLivenessScore_IR()` 方法来获取 IR 活体检测结果，示例代码如下：

```
LivenessInfo livenessInfo = new LivenessInfo();
//人脸信息处理
retCode = faceEngine.ASFProcessEx_IR(image, multiFaceInfo, FaceEngineMask.ASF_IR_LIVENESS);
if (retCode == 0)
{
    //获取 IR 活体检测结果
    retCode = faceEngine.ASFGetLivenessScore_IR(out livenessInfo);
}
return livenessInfo;
```

3.1.11 额头区域检测

额头区域检测需要在初始化引擎时设置额头区域检测算法功能，当人脸检测完成后，使用 `FaceEngine.ASFProcess_IR()` 或 `FaceEngine.ASFProcessEx_IR()` 方法检测图像的人脸信息，然后通过 `FaceEngine.ASFGetFaceLandMark()` 方法来获取 IR 活体检测结果，示例代码如下：

```
LandMarkInfo landMarkInfo = new LandMarkInfo();
//人脸信息检测
```

```
retCode = faceEngine.ASFProcessEx(image, multiFaceInfo, FaceEngineMask.ASF_FACELANDMARK);  
if (retCode == 0)  
{  
    //获取额头区域检测结果  
    retCode = faceEngine.ASFGetFaceLandMark(out landMarkInfo);  
}
```

3.2 通用方法

3.2.1 从 Bitmap 中读取 BGR 数据


从 Bitmap 中读取 BGR 数据的方法比较复杂,可以参考 [ImageUtil.ReadBMP\(Image image\)](#) 方法。

3.2.2 从 Bitmap 中读取 GRAY 数据

从 Bitmap 中读取 GRAY 数据的方法比较复杂,可以参考 [ImageUtil.ReadBMP_IR\(Image image\)](#) 方法。

4. 常见问题

4.1 常见问题问答

问题	参考回复
启动后引擎初始化失败	<ol style="list-style-type: none">1. 请选择对应的平台, 如 x64, x86 2. 删除 bin 下面对应的 asf_install.dat;3. 请确保 App.config 下的 APPID、SDKKEY 与当前 SDK 的平台、版本保持一致。
SDK 支持那些格式的图片人脸检测?	目前 SDK 支持的图片格式有 Jpg、Jpeg、Png、Bmp 等。
使用人脸检测功能对图片大小有要求吗?	推荐的图片大小最大不要超过 2M, 因为图片过大会使人脸检测的效率不理想, 当然图片也不宜过小, 否则会导致无法检测到人脸。
使用人脸识别引擎提取到的人脸特征信息是什么?	人脸特征信息是从图片中的人脸上提取的人脸特征点, 是 byte[] 数组格式。
SDK 人脸比对的阈值设为多少合适?	推荐值为 0.8, 用户可根据不同场景适当调整阈值。
可不可以将人脸特征信息保存起来, 等需要进行人脸比对的时候直接拿保存好的人脸特征进行比对?	可以, 当人脸个数比较多时推荐先存储起来, 在使用时直接进行比对, 这样可以大大提高比对效率。存入数据库时, 请以 Blob 的格式进行存储, 不能以 string 或其他格式存储。

VS 中调试激活时，返回 90113 SDK 激活失败, 请打开读写权限	1、调试环境下：当前 VS 没有权限，请使用 管理员身份运行 2、IIS 环境部署下的 Web 服务：请将 SDK 文件夹添加 IUSR 和 IIS_USRS、NETWORK_SERVICE 用户的写入、修改权限, 并在 IIS 应用程序池中将高级设置中的标识设为 LocalSystem
在 .Net 项目中出现堆栈溢出问题	.Net 平台设置的默认堆栈大小为 256KB, SDK 中需要的大小为 512KB 以上，推荐调整堆栈的方法为： <pre>new Thread(new ThreadStart(delegate { ASF_MultiFaceInfo multiFaceInfo = FaceUtil.DetectFace(pEngine, imageInfo); })), 1024 * 512).Start();</pre>
在 .Net 项目中出现 x64 不能加载 SDK 的问题	首先使用 bool is64 = Environment.Is64BitProcess; 查看当前是否是 x64 位编译器，如果不是则按在 Visual Studio 中选择菜单“ 工具>选项>项目和解决方案>Web 项目 ”，在对话框中勾选“对网站和项目使用 IIS Express 的 64 位版本”。
X86 模式下批量注册人脸有内存溢出或图片空指针	请增加虚拟内存或每次批量注册人脸控制在 20 张图片范围内
图片中有人脸，但是检测时未检测到人脸	1. 请调整 detectFaceScaleVal 的值； 2. 请确认图片的宽度是否为 4 的倍数； 3. 请确认图片是否通过 ImageUtil.ReadBMP 方法进行数据调整。
尝试读取或写入受保护的内存	尝试读取或写入受保护的内存是内存操作不当导致，请查看您程序中是否有内存使用后未释放，或者使用了已经释放的内存！
销毁引擎时程序报错	请先确认销毁引擎的时候，引擎是否处于被占用的状态。
多人脸活体检测，调用 ASFProcess() 接口，返回值为 9	错误码 9 表示缓冲上溢。建议在活体检测引擎初始化时，将 detectFaceMaxNum 值设置为不小于实际要检测人脸的值。
多线程调用注意事项	1. 同一个引擎可以使用多线程调用不同算法； 2. 多线程调用同一个算法接口需要启用不同的引擎

更多常见问题请访问 <https://ai.arcsoft.com.cn/manual/faqs.html>。

4.2 其他帮助

如您想要了解更多虹软的产品，请访问虹软官网 <http://www.arcsoft.com.cn/>，或者您在开发的过程中遇到了问题，或者对我们的人脸识别 SDK 有什么意见或建议，欢迎在虹软官方论坛 <https://ai.arcsoft.com.cn//bbs/portal.php> 上发帖提问，我们的工作人员会竭力为您解答。