# ArcSoft Face Detection

开发指导文档



© 2017 ArcSoft Inc. All rights reserved.

ArcSoft Corporation
46601 Fremont Blvd.
Fremont, CA 94538
http://www.arcsoft.com

**Trademark or Service Mark Information**

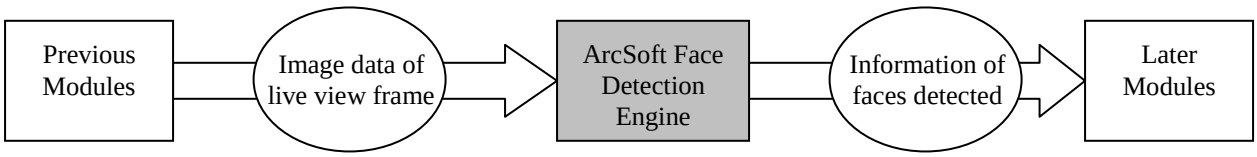
ArcSoft Inc. and ArcWare are registered trademarks of ArcSoft Inc.

Other product and company names mentioned herein may be trademarks and/or service marks of their respective owners. The absence of a trademark or service mark from this list does not constitute a waiver of ArcSoft Inc.'s trademark or other intellectual property rights concerning that trademark or service mark.

The information contained in this document is for discussion purposes only. None of the information herein shall be interpreted as an offer or promise to any of the substance herein nor as an agreement to contract or license, or as an implication of a transfer of rights. Any and all terms herein are subject to change at the discretion of ArcSoft. Copying, distributing, transferring or any other reproduction of these documents or the information contained herein is expressly prohibited, unless such activity is expressly permitted by an authorized representative of ArcSoft, Inc.

# Chapter 1: 概述

虹软人脸检测引擎工作流程图：




## 1.1. 运行环境

- Windows

## 1.2. 系统要求

- 64 位系统， Windows7 以上

## 1.3. 依赖库

- None

# Chapter 2: 结构与常量

## 2.1. 基本类型

```
typedef MInt32 AFD_FSDK_OrientPriority;
typedef MInt32 AFD_FSDK_OrientCode;
```

所有基本类型在平台库中有定义。 定义规则是在 ANSIC 中的基本类型前加上字母“M”同时将类型的第一个字母改成大写。例如“long” 被定义成“MLong”

## 2.2. 数据结构与枚举

### 2.2.1. AFD_FSDK_FACERES

**描述**

检测到的脸部信息

**定义**

```
typedef struct{
    MRECT     * rcFace;
    MLong     nFace;
    AFD_FSDK_OrientCode    * lfaceOrient;
} AFD_FSDK_FACERES, * LPAFD_FSDK_FACERES;
```

**成员变量**

| | |
|---|---|
| rcFace | 人脸矩形框信息 |
| nFace | 人脸个数 |
| lfaceOrient | 人脸角度信息 |

### 2.2.2. AFD_FSDK_VERSION

**描述**

SDK 版本信息

**定义**

```
 typedef struct
 {
    MInt32 lCodebase;
```

```
    MInt32 lMajor;
    MInt32 lMinor;
    MInt32 lBuild;
    MPChar Version;
    MPChar BuildDate;
    MPChar CopyRight;
} ArcSoft_Face_Detection_Version;
```

**成员描述**

| | |
|---|---|
| lCodebase | 代码库版本号 |
| lMajor | 主版本号 |
| lMinor | 次版本号 |
| lBuild | 编译版本号，递增 |
| Version | 字符串形式的版本号 |
| BuildDate | 编译时间 |
| CopyRight | copyright |

## 2.2.3. AFD_FSDK_OrientPriority

**描述**

定义脸部检测角度的优先级

**定义**

```
enum _AFD_FSDK_OrientPriority{
    AFD_FSDK_OPF_0_ONLY           = 0x1,
    AFD_FSDK_OPF_90_ONLY          = 0x2,
    AFD_FSDK_OPF_270_ONLY         = 0x3,
    AFD_FSDK_OPF_180_ONLY         = 0x4,
    AFD_FSDK_OPF_0_HIGHER_EXT     = 0x5
};
```

**成员描述**

| | |
|---|---|
| AFD_FSDK_OPF_0_ONLY | 检测 0 度方向 |
| AFD_FSDK_OPF_90_ONLY | 检测 90 度方向 |
| AFD_FSDK_OPF_270_ONLY | 检测 270 度方向 |
| AFD_FSDK_OPF_180_ONLY | 检测 180 度方向 |
| AFD_FSDK_OPF_0_HIGHER_EXT | 检测 0， 90， 180， 270 四个方向,0 度更优先 |

## 2.2.4. AFD_FSDK_OrientCode

**描述**

定义检测结果中的人脸角度

**定义**

```
enum _AFD_FSDK_OrientCode{
    AFD_FSDK_FOC_0   = 0x1,
    AFD_FSDK_FOC_90  = 0x2,
    AFD_FSDK_FOC_270 = 0x3,
    AFD_FSDK_FOC_180 = 0x4,
    AFD_FSDK_FOC_30  = 0x5,
    AFD_FSDK_FOC_60  = 0x6,
    AFD_FSDK_FOC_120 = 0x7,
    AFD_FSDK_FOC_150 = 0x8,
    AFD_FSDK_FOC_210 = 0x9,
    AFD_FSDK_FOC_240 = 0xa,
    AFD_FSDK_FOC_300 = 0xb,
    AFD_FSDK_FOC_330 = 0xc
};
```

**成员描述**

| | |
|---|---|
| AFD_FSDK_FOC_0 | 0 度 |
| AFD_FSDK_FOC_90 | 90 度 |
| AFD_FSDK_FOC_270 | 270 度 |
| AFD_FSDK_FOC_180 | 180 度 |
| AFD_FSDK_FOC_30 | 30 度 |
| AFD_FSDK_FOC_60 | 60 度 |
| AFD_FSDK_FOC_120 | 120 度 |
| AFD_FSDK_FOC_150 | 150 度 |
| AFD_FSDK_FOC_210 | 210 度 |
| AFD_FSDK_FOC_240 | 240 度 |
| AFD_FSDK_FOC_300 | 300 度 |
| AFD_FSDK_FOC_330 | 330 度 |

## 2.2.5. 支持的颜色格式

**描述**

颜色格式及其对齐规则

**定义**

| | |
|---|---|
| ASVL_PAF_I420 | 8-bit Y 层，之后是 8-bit 的 2x2 采样的 U 层和 V 层 |
| ASVL_PAF_YUYV | Y0, U0, Y1, V0 |
| ASVL_PAF_RGB24_B8G8R8 | BGR24, B8G8R8 |

# Chapter 3: API 说明

## 3.1. AFD_FSDK_InitialFaceEngine

**原型**

```
MRESULT AFD_FSDK_InitialFaceEngine(
       MPChar                          AppId,
       MPChar                          SDKKey,
       MByte                           *pMem,
       MInt32                          lMemSize,
       MHandle                         *pEngine,
       AFD_FSDK_OrientPriority         iOrientPriority,
       MInt32                          nScale,
       MInt32                          nMaxFaceNum
);
```

**描述**

初始化脸部检测引擎

**参数**

| | | |
|---|---|---|
| AppId | [in] | 用户申请 SDK 时获取的 App Id |
| SDKKey | [in] | 用户申请 SDK 时获取的 SDK Key |
| pMem | [in] | 分配给引擎使用的内存地址 |
| lMemSize | [in] | 分配给引擎使用的内存大小 |
| pEngine | [out] | 引擎 handle |
| iOrientPriority | [in] | 期望的脸部检测角度的优先级 |
| nScale | [in] | 用于数值表示的最小人脸尺寸 有效值范围[2,50] 推荐值 16 |
| nMaxFaceNum | [in] | 用户期望引擎最多能检测出的人脸数 有效值范围[1,100] |

**返回值**

成功返回 MOK，否则返回失败 code。失败 codes 如下所列:

| | |
|---|---|
| MERR_INVALID_PARAM | 参数输入非法 |
| MERR_NO_MEMORY | 内存不足 |

## 3.2. AFD_FSDK_StillImageFaceDetection

**原型**

```
MRESULT AFD_FSDK_StillImageFaceDetection(
```

```
    MHandle         hEngine,
    LPASVLOFFSCREEN pImgData,
    LPAFD_FSDK_FACERES     pFaceRes
);
```

**描述**

根据输入的图像检测出人脸位置，一般用于静态图像检测

**参数**

| | | |
|---|---|---|
| hEngine | [in] | 引擎 handle |
| pImgData | [in] | 带检测图像信息 |
| pFaceRes | [out] | 人脸检测结果 |

**返回值**

成功返回 MOK，否则返回失败 code。

## 3.3. AFD_FSDK_UninitialFaceEngine

**原型**

```
MRESULT AFD_FSDK_UninitialFaceEngine(
       MHandle       hEngine
);
```

**描述**

销毁引擎，释放相应资源

**参数**

| | | |
|---|---|---|
| hEngine | [in] | 引擎 handle |

**返回值**

成功返回 MOK，否则返回失败 code。失败 codes 如下所列:

| | |
|---|---|
| MERR_INVALID_PARAM | 参数输入非法 |

## 3.4. AFD_FSDK_GetVersion

**原型**

```
const AFD_FSDK_Version * AFD_FSDK_GetVersion(
       MHandle       hEngine
```

```
);
```

**描述**

获取 SDK 版本信息

**参数**

| | | |
|---|---|---|
| `hEngine` | `[in]` | 引擎 handle |

# Chapter 4: 示例代码

注意,使用时请替换申请的 **APPID SDKKEY**，并设置好文件路径和图像尺寸

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <Windows.h>
#include "arcsoft_fsdk_face_detection.h"
#include "merror.h"

#pragma comment(lib,"libarcsoft_fsdk_face_detection.lib")

#define WORKBUF_SIZE        (40*1024*1024)
#define INPUT_IMAGE_PATH "sample.bmp"
#define APPID       ""                  //APPID
#define SDKKey      ""                  //SDKKey

bool readBmp24(const char* path, uint8_t **imageData, int *pWidth, int *pHeight)
{
	if (path == NULL || imageData == NULL || pWidth == NULL || pHeight == NULL)
	{
		return false;
	}
	FILE *fp = fopen(path, "rb");
	if (fp == NULL)
	{
		return false;
	}
	fseek(fp, sizeof(BITMAPFILEHEADER), 0);
	BITMAPINFOHEADER head;
	fread(&head, sizeof(BITMAPINFOHEADER), 1, fp);
	*pWidth = head.biWidth;
	*pHeight = head.biHeight;
	int biBitCount = head.biBitCount;
	if (24 == biBitCount)
	{
		int lineByte = ((*pWidth) * biBitCount / 8 + 3) / 4 * 4;
		*imageData = (uint8_t *)malloc(lineByte * (*pHeight));
		uint8_t * data = (uint8_t *)malloc(lineByte * (*pHeight));
		fseek(fp, 54, SEEK_SET);
		fread(data, 1, lineByte * (*pHeight), fp);
		for (int i = 0; i < *pHeight; i++)
		{
			for (int j = 0; j < *pWidth; j++)
			{
				memcpy((*imageData) + i * (*pWidth) * 3 + j * 3, data +
(((*pHeight) - 1) - i) * lineByte + j * 3, 3);
			}
		}
		free(data);
	}
	else
	{
```

```
                fclose(fp);
                return false;
        }
        fclose(fp);
        return true;
}
int main()
{
        /* 初始化引擎和变量 */
        MRESULT nRet = MERR_UNKNOWN;
        MHandle hEngine = nullptr;
        MInt32 nScale = 16;
        MInt32 nMaxFace = 10;
        MByte *pWorkMem = (MByte *)malloc(WORKBUF_SIZE);
        if (pWorkMem == nullptr)
        {
                return -1;
        }
        nRet = AFD_FSDK_InitialFaceEngine(APPID, SDKKey, pWorkMem, WORKBUF_SIZE,
&hEngine, AFD_FSDK_OPF_0_HIGHER_EXT, nScale, nMaxFace);
        if (nRet != MOK)
        {
                return -1;
        }
        /* 打印版本信息 */
        const AFD_FSDK_Version * pVersionInfo = nullptr;
        pVersionInfo = AFD_FSDK_GetVersion(hEngine);
        fprintf(stdout, "%d %d %d %d\n", pVersionInfo->lCodebase, pVersionInfo-
>lMajor, pVersionInfo->lMinor, pVersionInfo->lBuild);
        fprintf(stdout, "%s\n", pVersionInfo->Version);
        fprintf(stdout, "%s\n", pVersionInfo->BuildDate);
        fprintf(stdout, "%s\n", pVersionInfo->CopyRight);

        /* 读取静态图片信息，并保存到ASVLOFFSCREEN结构体 （以ASVL_PAF_RGB24_B8G8R8格式
为例） */
        ASVLOFFSCREEN offInput = { 0 };
        offInput.u32PixelArrayFormat = ASVL_PAF_RGB24_B8G8R8;
        offInput.ppu8Plane[0] = nullptr;
        readBmp24(INPUT_IMAGE_PATH, (uint8_t**)&offInput.ppu8Plane[0],
&offInput.i32Width, &offInput.i32Height);
        if (!offInput.ppu8Plane[0])
        {
                fprintf(stderr, "Fail to ReadBmp(%s)\n", INPUT_IMAGE_PATH);
                AFD_FSDK_UninitialFaceEngine(hEngine);
                free(pWorkMem);
                return -1;
        }
        else
        {
                fprintf(stdout, "Picture width : %d , height : %d \n",
offInput.i32Width, offInput.i32Height);
        }
        offInput.pi32Pitch[0] = offInput.i32Width * 3;

        /* 人脸检测 */
        LPAFD_FSDK_FACERES   FaceRes = nullptr;
```

```
	nRet = AFD_FSDK_StillImageFaceDetection(hEngine, &offInput, &FaceRes);
	if (nRet != MOK)
	{
		fprintf(stderr, "Face Detection failed, error code: %d\n", nRet);
	}
	else
	{
		fprintf(stdout, "The number of face: %d\n", FaceRes->nFace);
		for (int i = 0; i < FaceRes->nFace; ++i)
		{
			fprintf(stdout, "Face[%d]: rect[%d,%d,%d,%d], Face
orient: %d\n", i, FaceRes->rcFace[i].left, FaceRes->rcFace[i].top, FaceRes-
>rcFace[i].right, FaceRes->rcFace[i].bottom, FaceRes->lfaceOrient[i]);
		}
	}

	/* 释放引擎和内存 */
	nRet = AFD_FSDK_UninitialFaceEngine(hEngine);
	if (nRet != MOK)
	{
		fprintf(stderr, "UninitialFaceEngine failed , errorcode is %d \n",
nRet);
	}
	free(offInput.ppu8Plane[0]);
	free(pWorkMem);
	return 0;
}
```