ArcSoft LiveFace Recognition

Developer’s Guide

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.

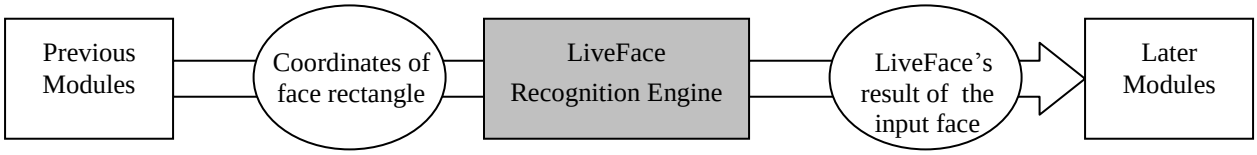
Other product and company names mentioned herein may be trademarks and/or service marks of their respective owners. The absence of a trademark or service mark from this list does not constitute a waiver of ArcSoft Inc.'s trademark or other intellectual property rights concerning that trademark or service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

Version	Date	Modifier	Summary of changes
1.0	10/28/2017	ArcSoft	Initial version
1.4.0.38	04/05/2018	ArcSoft	Interfaces Updated
1.6.0.10	05/08/2018	ArcSoft	Interfaces Updated

Chapter 1: Overview

This document provides a description for ArcSoft Automatic LiveFace Recognition library. This library provides the ability to distinguish living face by the given coordinates of face rectangles.



1.1. Deliverables

- Header files
- Developer guide document
- Library file(s)
- Sample code and/or simple test app

1.2. Platforms

- COACH
- Symbian
- Ti DM350
- Green Hills
- ADS
- Win32
- Wince2005

1.3. System requirements

- Memory: 30MB
- CPU : 100MHz or above
- ROM Space: 17MB

1.4. Dependencies

- ArcSoft Platform Library

Note: Please also include the platform header file and LiveFace recognition library in the "input" folder. If there is no corresponding platform library for some platform, it means that Click library is independent to this platform.

- asvloffscreen.h

Note: In order to detailedly understand the "asvloffscreen.h" file, which defines ASVLOFFSCREEN structure and the general color formats, please refer to the corresponding documents. In the current LiveFace recognition version, only" ASVL_PAF_NV21" is supported.

Chapter 2: Structures and Constants

2.1. About basic data types

```
typedef MVoid*                    ASAE_ENGINE;
```

All basic data types defined in the facial liveface recognition header file are redefined in the platform SDK. The general re-naming rule is that one prefix "M" is added to the basic types of ANSIC and the first character is made capitalized. For example, "long" is redefined to "MLong". Please refer to the ArcSoft Platform Library SDK API reference for more details.

2.2. Data structures

2.2.1. ArcSoft_LiveFace _Version

Description

This structure describes the version information of the facial LiveFace recognition library.

Definitions

```
typedef struct{
        MLong lCodebase;
        MLong lMajor;
        MLong lMinor;
        MLong lBuild;
        MTChar Version[50];
        MTChar BuildDate[20];
        MTChar CopyRight[50];
} ArcSoft_LiveFace _Version;
```

Member description

lCodebase	Codebase version number
lMajor	Major version number
lMinor	Minor version number
lBuild	Build version number, increasable only
Version	Version in string form
BuildDate	Version in string form
CopyRight	Copyright

2.2.2. ASAE_LiveFaceRESULT

Description

This structure defines the facial LiveFace result of the detected faces on the whole input image.

Definitions

```
typedef struct{
        MInt32 *pLFResultArray;
        MInt32 *pGenderResultArray;
        MLong  lFaceNumber;
        MFloat *cl_conf;
        MInt32 FaceAngle[3];
        MInt32 lightvalue;
        MRECT  pFaceRectArray;
        MFloat closedeyelevel[2];

}ASAE_LFRESULT, *LPASAE_LFRESULT;
```

Member description

pLFResultArray;	[output]The LF result array with same length as pFaceRectArray
pGenderResultArray	[output]The gender result array with same length as pFaceRectArray, reserved
lFaceNumber	[output]It is the same as lFaceNumber in ASAE_FACEINPUT
cl_conf	[output] the confidence of one face,length = 8
FaceAngle[3]	[output] roll，yaw，pitch
lightvalue	[output] the value of lightness
pFaceRectArray	[output] get rect from landmark
closedeyelevel[2]	[output] the level of left eye and right eye closed

2.2.3. ASAE _FACEINPUT

Description

This structure defines the input face information.

Definitions

```
Typedef struct{
        MRECT               *pFaceRectArray;
        MLong               *pFaceOrientArray;
        MLong               lFaceNumber;
        MInt32          *faceID;
} ASAE_FACEINPUT, *LPASAE_FACEINPUT;
```

Member description

pFaceRectArray	Bounding boxes of input faces
pFaceOrientArray	Orientations of input faces. Can be set as one item of ASAE_ OrientCode
lFaceNumber	Number of detected faces
FaceID	Face id

2.3. Enumeration

2.3.1. ASAE_ORIENTCODE

Description

This structure defines the orientation of the face in anti-clockwise sequence.

Definitions

```
enum ASAE_OrientCode {
              ASAE_FOC_0          = 0x1,
              ASAE_FOC_90         = 0x2,
              ASAE_FOC_270        = 0x3,
              ASAE_FOC_180        = 0x4,
              ASAE_FOC_30         = 0x5,
              ASAE_FOC_60         = 0x6,
              ASAE_FOC_120        = 0x7,
              ASAE_FOC_150        = 0x8,
              ASAE_FOC_210        = 0x9,
              ASAE_FOC_240        = 0xa,
              ASAE_FOC_300        = 0xb,
              ASAE_FOC_330        = 0xc
};
```

Member description

ASAE_FOC_0	0 degree
ASAE_FOC_90	90 degree
ASAE_FOC_270	270 degree
ASAE_FOC_180	180 degree
ASAE_FOC_30	30 degree
ASAE_FOC_60	60 degree
ASAE_FOC_120	120 degree
ASAE_FOC_150	150 degree
ASAE_FOC_210	210 degree
ASAE_FOC_240	240 degree
ASAE_FOC_300	300 degree
ASAE_FOC_330	330 degree

Chapter 3: Supporting API Reference

3.1. Facial LiveFace APIs

3.1.1. ASAE_InitLFEngine

Prototype

```
EXP_API MRESULT ASAE_InitLFEngine (
        MHandle             hMemMgr,
        ASAE_ENGINE         *pEngine,
);
```

Description

This function is used to initialize the facial liveface recognition engine, and please initialize (`*pEngine`) to "MNull" when (`*pEngine`) is defined.

Parameters

`hMemMgr`	[in]	Handle of memory manager
`pEngine`	[out]	Pointer pointing to the liveface recognition engine

Return value

Return MOK if succeed and return error codes if fails. Error codes are listed as follows:

MERR_INVALID_PARAM	Invalid input parameters
MERR_NO_MEMORY	Memory is not enough
MERR_ENGINE_NONZERO	(*pEngine) is not equal to Null before ASAE_InitExpEngine is called, and it is used to avoid multi-initialization.

3.1.2. ASAE_ LiveFace_StaticImage

Prototype

```
MRESULT ASAE_LiveFace _StaticImage(
        MHandle                         hMemMgr,
        ASAE_ENGINE                     pEngine,
        LPASVLOFFSCREEN                 pImginfo,
        LPASAE_THRESHOLDPARM            PThreshold,
        LPASAE_FACEINPUT                pFaceRes,
        LPASAE_LFRESULT                 pLFRes ,
        ASAE_FNPROGRESS                 fnCallback,
        MVoid                           *pParam
);
```

Description

This function is used to estimate the liveface of the detected face throughout the input image.

Parameters

hMemMgr	[in]	Handle of memory manager
pEngine	[in]	Liveface Estimation engine
PImginfo	[in]	Pointer pointing to an ASAE_OFFSCREEN structure containing the information of input image
PThreshold	[in]	The threshold of two models
pFaceRes	[in]	Pointer pointing to an ASAE_FACEINPUT structure containing the cooridnates of face rectangles
pLFRes	[out]	Pointer pointing to an ASAE_LFRESULT structure containing the results of facial LiveFace estimate
fnCallback	[in]	User defined callback function, can be set to NULL
pParam	[in]	Parameters for callback function, can be set to NULL

Return value

Return MOK if succeed and return error codes if fail. Error codes are listed as follows:

MERR_INVALID_PARAM	Invalid input parameters
MERR_NO_MEMORY	Memory is not enough
MERR_USER_CANCEL	User cancels

3.1.3. ASAE_LiveFace _Preview

```
MRESULT ASAE_ExpEstimation_Preview (
        MHandle                         hMemMgr,
        ASAE_ENGINE                     pEngine,
        LPASVLOFFSCREEN                 pImginfo,
        LPASAE_THRESHOLDPARM            PThreshold,
        LPASAE_FACEINPUT                pFaceRes,
        LPASAE_LFRESULT                 pLFRes ,
        ASAE_FNPROGRESS                 fnCallback,
        MVoid                           *pParam
);
```

Description

This function is used to estimate the liveface of the detected faces on preview mode automatically.

Parameters

hMemMgr	[in]	Handle of memory manager
pEngine	[in]	LiveFace Estimation engine
PImginfo	[in]	Pointer pointing to an ASAE_OFFSCREEN structure containing the information of input image

PThreshold	[in]	The threshold of two models
pFaceRes	[in]	Pointer pointing to an ASAE_FACEINPUT structure containing the cooridnates of face rectangles
pLFRes	[out]	Pointer pointing to an ASAE_LFRESULT structure containing the results of facial LiveFace estimate
fnCallback	[in]	User defined callback function, can be set to NULL

Return value

Return MOK if succeed and return error codes if fail. Error codes are listed as follows:

MERR_INVALID_PARAM	Invalid input parameters
MERR_NO_MEMORY	Memory is not enough
MERR_USER_CANCEL	User cancels

3.1.4. ASAE_UninitExpEngine

Prototype

```
MRESULT ASAE_UninitExpEngine (
      MHandle          hMemMgr,
      ASAE_ENGINE      *pEngine
);
```

Description

This function is used to release the facial liveface recognition engine and please set Engine to “MNull” after `ASAE_UninitLFEngine` is called.

Parameters

hMemMgr	[in]	Handle of memory manager
Engine	[in]	facial Liveface engine

Return value

Return MOK if succeed and return error codes if fails. Error codes are listed as follows:

MERR_INVALID_PARAM	Invalid input parameters

3.2. Callback functions

3.2.1. ASAE_FNPROGRESS

Prototype

```
typedef MRESULT (*ASAE_FNPROGRESS) (
      MVoid *pParam1,
      MVoid *pParam2
```

```
);
```

Description

It defines the callback function for the progressive processing information.

Parameters

pParam1	[in]	Not used
pParam2	[in]	Caller-defined data

Return value

User controls the return value.

Return 1 to cancel the process and return 0 to make the process continue.

3.3. Version Information API

3.3.1. ArcSoft_LiveFace _GetVersion

Prototype

```
const ArcSoft_LiveFace _Version* ArcSoft_LiveFace _GetVersion();
```

Description

This function is used to get the version information of the facial liveface recognition library.

3.4. Sample Codes

Here is the sample code for reference only.

```
#define LF_DETECTION
#ifdef LF_DETECTION
#include "arcsoft_liveface.h"
#pragma comment(lib, "libarcsoft_antisproof_liveface.lib")
#endif


//support one face
void main(IplImage * img, ALT_FACE_INFOR* outline)
{

#ifdef LF_DETECTION
        MVoid* LFMemBuffer;
        MHandle LFMemHandle;
        ASVLOFFSCREEN LFImageInfo;
        ASAE_ENGINE LFEngine;
```

```
        ASAE_FACEINPUT LFFaceInput;
        ASAE_RACERESULT LFResult;

//initial
        LFEngine = 0;
        LFMemBuffer = malloc(1024*1024*100);
        LFMemHandle = MMemMgrCreate(LFMemBuffer,1024*1024*100);
        int res = ASAE_InitLFEngine(LFMemHandle, &LFEngine);
        if (res != MOK)
        {
                printf("error when load LF model\n");
         }
        LFFaceInput.lFaceNumber = 0;
        LFFaceInput.pFaceRectArray = (MRECT*)malloc(MAXFACE*sizeof(MRECT));
        LFFaceInput.pFaceOrientArray = (MLong*)malloc(MAXFACE*sizeof(MLong));

        LFImageInfo.i32Width = img->width;
        LFImageInfo.i32Height = img->height;
        LFImageInfo.ppu8Plane[0] = (unsigned char*)img->imageData;
        LFImageInfo.ppu8Plane[1] = 0;
        LFImageInfo.ppu8Plane[2] = 0;
        LFImageInfo.ppu8Plane[3] = 0;
        LFImageInfo.pi32Pitch[0] = img->widthStep;
        LFImageInfo.pi32Pitch[1] = 0;
        LFImageInfo.pi32Pitch[2] = 0;
        LFImageInfo.pi32Pitch[3] = 0;
        LFImageInfo.u32PixelArrayFormat = ASVL_PAF_RGB24_B8G8R8;


        //preview
        //int res = ASAE_LiveFace_Preview(LFMemHandle, LFEngine, &LFImageInfo, PThreshold,
（ASAE_FACEOUTLINE_INPUT*）outline，&LFResult, 0, 0);
        //static image
        PThreshold.thresholdmodel1 = 0.5;
        PThreshold.thresholdmodel2 = 0.5;
        MInt32 parm[2] = {0,0};
        int res = ASAE_LiveFace_StaticImage(LFMemHandle, LFEngine, &LFImageInfo, PThreshold，
（ASAE_FACEOUTLINE_INPUT*）outline, &LFResult, 0, 0);
        if (res != MOK)
        {
                printf("detect error!\n");
                return;
        }
```

```
        // result
        if(LFResult.pLFResultArray[0]==0)
                Fate++;
        else if(LFResult.pLFResultArray[0]==1)
                Live++;
        else
                unknown++;
        Printf("FateFace_1:%.4f", LFResult. cl_conf[0]); \\paper_FateFace
        Printf("LiveFace_1:%.4f", LFResult. cl_conf[1]); \\paper_LiveFace
        Printf("FateFace_2:%.4f", LFResult. cl_conf[2]); \\screen_FateFace
        Printf("LiveFace_2:%.4f", LFResult. cl_conf[3]); \\screen_LiveFace


//release

        if (LFEngine)
        {
                ASAE_UninitLFEngine(LFMemHandle, &LFEngine);
        }

        if (LFFaceInput.pFaceOrientArray)
        {
                free(LFFaceInput.pFaceOrientArray);
                LFFaceInput.pFaceOrientArray = 0;
        }
        if (LFFaceInput.pFaceRectArray)
        {
                free(LFFaceInput.pFaceRectArray);
                LFFaceInput.pFaceRectArray = 0;
        }

        if (LFMemHandle)
        {
                MMemMgrDestroy(LFMemHandle);
                LFMemHandle = 0;
        }
        if (LFMemBuffer)
        {
                free(LFMemBuffer);
                LFMemBuffer = NULL;
        }
#endif
```

```
}
```