FaceTech 얼굴인식 SDK 기능정의서

（V7.1）

금호석유화학주식회사

목 차

1 FaceTech SDK 개요

1.1 문서 개정 내역

1.2 개요

금호석유화학주식회사(이하 KKPC) 얼굴인식 SDK (FaceTech ®) 는 얼굴인식과 관련된 많은 기능들을 도구화 (얼굴검출, 눈의 위치, 얼굴 특징점 추출 그리고 얼굴인식) 하고 있다.

1.3 함수

FaceTech ® SDK 는 아래와 같은 기능들을 제공한다.

1.3.1 얼굴 검출

영상 또는 사진 이미지(이하 이미지)를 입력한 후 이미지로부터 얼굴의 위치를 검출할 수 있다. 그리고 이미지로부터 얼굴의 개수와 각 얼굴의 좌표를 사각형 형태의 값(Left, Top, Right, Bottom)으로 추출할 수 있다. 사각형 형태의 값은 아래 그림 1.1 에서 알 수 있듯이 Left, Top 을 시작으로 Right , Bottom 까지를 의미한다. 이 사각형 형태의 값에 대한 자세한 사항은 데이터 구조체 HS_FaceCandidate 를 참조한다.

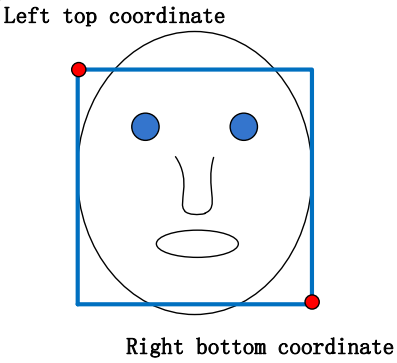

그림 1.1. 얼굴검출의 사각형 좌표

FaceTech ® SDK 는 얼굴검출에 있어 다음과 같은 특징들을 갖는다.

a) 얼굴의 크기 : 이미지상에서 얼굴검출 범위는 20 Pixel 부터 이미지상에서

검출가능영역까지의 짧은 면에서 이루어진다.

b) 회전 각 : 회전 각은 -30° 부터 30°

c) 고객 숙임 각 : 고개 숙임 각은 -10° 부터 10°

d) 기울기 각 : 기울기 각은 -15° 부터 15°

e) 하나의 이미지에서 하나 이상의 복수개의 얼굴을 검출할 수 있다.

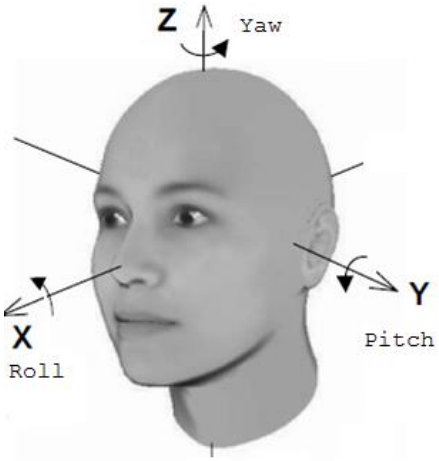

그림 1.2 얼굴의 회전, 기울기, 고개 숙임 각도에 대한 정의

1.3.2 눈 좌표

얼굴검출단계로부터 구해진 사각형 형태의 얼굴좌표를 사용함으로써 눈 중심점을 찾을 수 있다. 더 자세한 정보는 데이터 구조체 HS_EyePoint 를 참조한다. 눈의 중심은 눈의 상단 경계선과 하단 경계선 사이의 중심점과 눈의 안쪽과 바깥쪽 코너들 사이 중심점을 연결하는 점을 참조한다. 여기서 중요한 사항은 눈의 중심점이 눈동자의 중심이 아니다.

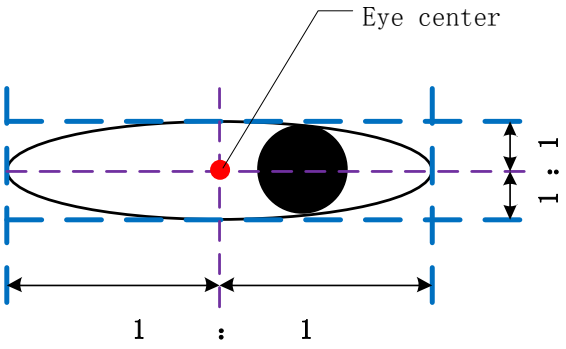

그림 1.3 눈의 중앙

FaceTech ® SDK 는 눈 좌표 파악에 있어 다음과 같은 특징 및 제한들을 갖는다.

a) 안경 : 선글라스를 제외한 안경들은 눈 좌표 파악에 영향을 주지는 않는다.
b) 눈 감음 : 눈 좌표 파악에 영향이 없다. 얼굴 인증할 경우 매칭 점수가 낮게 나올 가능성은 있다.
c) 얼굴크기 : 눈 검출함에 있어 요구 되어지는 얼굴의 최소 크기는 30*30 pixels 이다.
d) 회전 각 : 회전각 범위는 -25° 부터 25°
e) 고개 숙임 각 : 고개 숙임각 범위는 -10° 부터 10°
f) 기울기 각 : 기울기 각의 범위는 -15° 부터 15°

1.3.3 얼굴 특징점 추출

눈의 중심을 구한 후 얼굴로부터 생체 특징 점을 추출할 수 있다.

1.3.4 얼굴인식

얼굴 이미지들로부터 개별적인 얼굴 특징 점들을 추출함으로써 등록된 사람인지 아닌지 구별할 수 있거나 등록된 사람인지 식별할 수 있다..

FaceTech ® SDK 는 얼굴인식을 위해 다음과 같은 특징들을 가지고 있다.

a) 회전각 : 회전각 범위는 -15° 부터 15°
b) 고객 숙임 각 : 고개 숙임 각의 범위는 -15° 부터 15°
c) 기울기 각 : 기울기 각의 범위는 -10° 부터 10°
d) FaceTech ® SDK 는 모든 성별 및 연령대를 포함 한다.
e) 조도 : 강한 그림자를 드리우게 하는 것이 아니라면 조도의 변화는 성능에 관계 없다.
f) 안경 : 검은 선글라스를 제외한 안경착용을 허용한다.
g) 얼굴크기 : 얼굴인식을 위한 가장 작은 눈간의 거리는 30 pixels 이다.

2 시스템 요구조건

2.1 하드웨어 사양

CPU	Pentium Ⅲ 700MHz 이상
메모리	256MB 이상
Hard disk	40G 이상

2.2 소프트웨어 최소 요건

OS	Microsoft Windows 2000/XP/2003/7/10
개발환경	표준 C 인터페이스를 준수하는 개발환경

3 SDK

3.1 개발환경

FaceTech ® SDK 는 "C" Language 를 이용하여 개발되어졌다. 그러므로 표준 "C" 인터페이스방식을 통하여 이용할 수 있다.

3.2 상수정의

```
#define   WATCHLIST_OUT 5     // maximum candidate number of WatchList
```

메모리 DB에 Gallery(등록되어진 사용자들)들을 모두 로드 한 환경, 즉, 1:N 환경에서 1회 매칭을 수행하였을 때 순위를 나타 낼 수 있는 후보자 개수를 의미 하며 1:N 환경에서의 운영 프로그램이 아니라면 무시하여도 된다.

3.3 데이터 구조체 정의

- HS_RECT: rectangular

```
struct HS_RECT
{
    long    left;
    long    top;
    long    right;
    long    bottom;
};
```

본 구조체는 얼굴검출 시 사용되는 사각형 형태의 얼굴좌표를 담고 있는 데이터 구조체이다.

- HS_FACEMODE: face recognition mode

```
enum HS_FACEMODE
{
    HS_WatchList,          // watch list
    HS_DBScan,             // DBScan
    HS_Verify,             // Verification
    HS_1Vs1DBScan          // DBScan with 1:1 interface
};
```

본 열거형 변수는 얼굴인식 모드를 의미하며 일반적으로 HS_Verify 를 많이 사용한다.

- HS_FaceCandidate : result of face detection

```
struct HS_FaceCandidate        // 검출한 얼굴정보 모두를 저장
{
    int scale;                 // 사용하지 않음
    int trueleft;              // 후보얼굴의 좌측상단(시작) 좌표
    int truetop;
    int trueright;             // 후보얼굴의 우측하단(끝)좌표
    int truebottom;
    int left;
    int top;
    float confidence;          // 얼굴검출의 신뢰도 0.0  부터 100.0
};
```

본 구조체는 얼굴검출 결과에 대한 정보를 담고 있으며 1회 얼굴검출 시 검출할 수 있는 얼굴의 총 수는 100개이며 이 상한값은 이미지의 크기에 의해 좌우된다.

- HS_EyePoint: result of eye localization

```
struct HS_EyePoint                  // 눈 좌표검출
{
    int         xleft;              // 좌측 눈 좌표값 횡축
    int         yleft;              // 좌측 눈 좌표값 종축
    int         xright;             // 우측 눈 좌표값 횡축
    int         yright;             // 우측 눈 좌표값 종축
    float confidence;          // 눈 검출 신뢰도 50.0  부터 100.0
};
```

얼굴검출 후 눈을 검출하게 되는데 이때 사용되는 눈에 대한 좌표를 담고있는 데이터 구조체 이다. 좌측과 우측 눈에 대한 좌표를 의미하며 각각의 좌표는 X 축과 Y 축의 좌표로 표현된다.

- HS_CandWithNumID: face recognition candidate result

```
struct HS_CandWithNumID       // 얼굴인식결과의후보리스트
{
    long lUserID;                   // 얼굴에 할당된 ID
    float fDist;                    // 인증점수로 범위는 0.0 부터 1.0
    BYTE   bConfi;                  // 얼굴인식 결과에 대한 신뢰도
};
```

1:N 환경에서의 얼굴인식 수행 결과를 담고 있는 데이터 구조체이다.

3.4 함수 정의

3.4.1 SDK 시작과 종료

- HS_InitFaceIDSDK

  SDK 를 사용하기 위한 초기화 함수

int HS_InitFaceIDSDK (HS_FACEMODE enMode) ;

인 수	변수/값	입/출력	설 명
	HS_FACEMODE	입력	주어진 값에 의해 얼굴인식 방법이 설정된다.
반환값	Error Code (Table 1 참조)		
설 명	SDK 를 초기화 하는 함수이며 SDK 의 기능을 사용하기 위하여 반드시 한번은 호출해야 한다.		

- HS_UninitFaceIDSDK

  SDK 사용을 종료하기 위한 함수

int HS_UninitFaceIDSDK ();

인 수	없음
반환값	Error Code (Table 1 참조)
설 명	SDK 사용을 종료하기 위하여 반드시 한번은 호출해야 한다

3.4.2 특징점 크기 구하기

- HS_GetFaceFeatureSize

  얼굴특징점 크기를 반환한다.

int HS_GetFaceFeatureSize

인 수	없음
반환값	특징점 데이터 크기
설 명	얼굴 특징점 크기를 조회 하는 함수이며 이 값은 특징점 추출, 사용자 등록, 사용자 인증시에 주로 참조하게 된다.

3.4.3 얼굴 검출

- HS_FaceDetection

  Gray Scale 이미지에서 얼굴을 검출하는 함수

```
int HS_FaceDetection(const BYTE *pbyGrayImage, int nImageWidth,
    int nImageHeight, int nMinFaceSize, int nMaxFaceSize,
    HS_FaceCandidate *pOutFaceCand, int *nFaceNum);
```

인 수	변수/값	입/출력	설 명
	pbyGrayImage	in	Gray Scale 이미지
	nImageWidth	in	Gray Scale 이미지의 가로 크기
	nImageHeight	in	Gray Scale 이미지의 세로 크기
	nMinFaceSize	in	얼굴검출 되어질 얼굴의 크기 값으로 최소값은 20 이상의 값을 사용한다.
	nMaxFaceSize	in	검출되어질 얼굴의 크기. 원본 이미지의 가로,세로 크기중 작은쪽 크기를 사용한다.
	pOutFaceCand	out	이미지상에서 검출되어질 후보얼굴을 담을 수 있는 구조체이며 본 함수를 호출할 때 메모리를 할당하여야 하며 이때 할당되어질 수 있는 수는 100 을 초과 할 수 없다.
	nFaceNum	out	검출되어진 얼굴 개수
반환값	Error Code (Table 1 참조)		
설 명	원본 이미지에서 생성된 Gray Scale 이미지로부터 얼굴을 검출하여 그 결과값을 반환 하는 함수.		

3.4.4 눈 검출

- HS_ LocateEye

  Gray Scale 의 이미지에서 눈 좌표를 검출 한다.

```
int HS_LocateEye(const BYTE *pbyGrayImage, int nImageWidth,
    int nImageHeight, const HS_RECT *pFaceRect,
    HS_EyePoint *pEyePoint);
```

인 수	변수/값	입/출력	설 명
	pbyGrayImage	in	가로크기 * 세로크기의 Gray Scale 이미지
	nImageWidth	in	Gray Scale 이미지의 가로크기
	nImageHeight	in	Gray Scale 이미지의 세로크기
	pFaceRect	in	Gray Scale 이미지 내에서 검출된 얼굴좌표
	pEyePoint	out	얼굴영역 내에서의 눈 좌표
반환값	Error Code (Table 1 참조)		
설 명	주어진 Gray Scale 이미지에서 눈을 검출하고 그 좌표를 반환한다.		

3.4.5 얼굴 특징점 추출

- HS_ GetFaceFeatureSize

  얼굴 특징점의 크기를 Byte 단위로 조회한다.

```
int HS_GetFaceFeatureSize();
```

인 수	없음
반환값	정수형 얼굴특징점 크기
설 명	Byte 단위의 얼굴 특징점 크기(정수형)를 반환한다.

- HS_FaceExtractionByEyepos : face feature extraction

  눈좌표를 이용한 얼굴 특징점 추출

```
int HS_FaceExtractionByEyepos(const BYTE *pbyGrayImage,
    int nImageWidth, int nImageHeight,
    const HS_EyePoint *pEyePoint, BYTE *pbyFeature);
```

인 수	변수/값	입/출력	설 명
	pbyGrayImage	in	가로크기*세로크기의 Gray Scale 이미지
	nImageWidth	in	Gray Scale 이미지의 가로 크기
	nImageHeight	in	Gray Scale 이미지의 세로 크기
	pEyePoint	in	검출된 얼굴 내에서의 눈 중심 값
	pbyFeature	out	특징점 크기로 메모리 할당된 버퍼
반환값	Error Code (Table 1 참조)		
설 명	Gray Scale 의 이미지와 눈 좌표를 입력하여 얼굴특징점 데이터를 추출한다.		

- HS_MergeMultiFeatures

  한 사용자의 여러 개의 특징점 데이터를 하나의 특징점 데이터로 병합한다.

```
int HS_MergeMultiFeatures(const unsigned char *pbyFeatures,
    int nFeatNum, unsigned char *pbyMergedFeature);
```

인 수	변수/값	입/출력	설 명
	pbyFeatures	in	다수개의 특징점 데이터
	nFeatNum	in	입력되는 특징점 개수
	pbyMergedFeature	out	특징점 크기로 할당된 메모리 버퍼
반환값	Error Code (Table 1 참조)		
설 명	한 사용자의 여러 개의 특징점 데이터들을 하나의 특징점 데이터로 병합한다.		

3.4.6 특징점 데이터 메모리 로딩 및 제거

- HS_LoadNumIDPopulation

  정수형 ID 를 가진 특징점 데이터를 메모리에 로드 한다.

```
int HS_LoadNumIDPopulation(long lUserID, const BYTE *pbyFeature);
```

인 수	변수/값	입/출력	설 명
	lUserID	in	정수형 개별 아이디
	pbyFeature	in	개별 얼굴 특징점 데이터
반환값	Error Code (Table 1 참조)		
설 명	정수형 ID 를 가진 특징점 데이터를 메모리에 로드하며 특징점 데이터와 정수형 ID 를 하나씩 메모리에 로드 할 수 있다		

- HS_RemoveNumIDPopulation

  주어진 ID 에 해당하는 생체 특징점 데이터를 메모리에서 삭제 한다.

```
int HS_RemoveNumIDPopulation(long lUserID);
```

인 수	변수/값	입/출력	설 명
	lUserID	in	메모리에서 삭제하고자 하는 고유 ID
반환값	Error Code (Table 1 참조)		
설 명	인수로 전달된 ID 에 해당하는 얼굴 생체 특징점 데이터를 메모리에서 삭제한다.		

3.4.7 인증

- HS_ VerifyWithNumID

  주어진 ID 와 얼굴 특징점 데이터를 이용하여 인증을 수행한다.

```
int HS_VerifyWithNumID(const BYTE *pbyFeatures,
    int nProbeNum, long lUserID, float *fScore);
```

인 수	변수/값	입/출력	설 명
	pbyFeatures	in	하나 또는 이상의 얼굴 특징점 데이터
	nProbeNum	in	얼굴 특징점 데이터 개수
	lUserID	in	인증해야 할 사용자 ID
	fScore	out	인증 결과 점수
반환값	Error Code (Table 1 참초)		
설 명	주어진 ID 와 얼굴 특징점 데이터를 이용하여 인증을 수행한다. 본 함수에서는 여러 개의 얼굴 특징점 데이터를 이용하여 등록된 얼굴 특징점 데이터와 비교 인증한다.		

- HS_VerifyTwoTemplate

  두개의 얼굴 특징점 데이터(Gallery, Probe)를 이용하여 얼굴인증을 수행 한다.

```
int HS_VerifyTwoTemplate(const unsigned char *pbyProbeFeat,
    const unsigned char *pbyGalleryFeat, float *fScore);
```

인 수	변수/값	입/출력	설 명
	pbyProbeFeat	in	Probe 얼굴 특징점 데이터
	pbyGalleryFeat	in	Gallery 얼굴 특징점 데이터
	fScore	out	인증 점수
반환값	Error Code (Table 1 참조)		
설 명	입력한 얼굴 특징점 데이터 (Probe, Gallery)를 이용하여 서로의 유사도를 검사하며 그 결과 값으로 인증 점수를 출력한다.		

※ Probe : 얼굴인증을 수행하고자 하는 현재의 입력 데이터

※ Gallery : 기 등록된 사용자 얼굴 특징점 데이터

3.4.8 보조 함수

- HS_GetGalleryNum

  메모리에 로드 되어진 등록된 얼굴 특징점 데이터들의 갯수를 조회한다.

```
int HS_GetGalleryNum();
```

인 수	Void
반환값	등록되어진 사용자 갯수
설 명	사용자 등록 데이터를 메모리에 로드 하여 인증을 수행할 경우에 사용되어지며 현재 메모리에 로드 된 사용자 등록자 수를 반환한다.

3.4.9 이미지 처리 함수

- HS_GetImageSize

  이미지의 가로 세로 크기를 구하는 함수

```
int HS_GetImageSize(const char *strImagePathName,
    int *nImageWidth, int *nImageHeight);
```

인 수	변수/값	입/출력	설 명
	strImagePathName	in	다음 이미지 포맷에 대한 이미지를 지원한다. BMP, GIF, ICO, CUR, JBG, JPG, JPC, JP2, PCX, PGX, PNG, PNM, RAS, TGA, TIF, WBMP, WMF
	nImageWidth	out	이미지 가로 크기
	nImageHeight	out	이미지 세로 크기
반환값	Error Code (Table 1 참조)		
설 명	인수로 입력된 이미지 파일에 대한 이미지의 가로와 세로크기를 조회 한다.		

- HS_GetGrayImageFromFile

  Gray Scale 이미지를 구하는 함수

```
int HS_GetGrayImageFromFile(const char *strImagePathName,
    BYTE *pbyOutGrayImage);
```

인 수	변수/값	입/출력	설 명
	strImagePathName	in	컬러이미지 파일 절대경로명이 포함된 파일명 지원 파일 포맷 : BMP, GIF, ICO, CUR, JBG, JPG, JPC, JP2, PCX, PGX, PNG, PNM, RAS, TGA, TIF, WBMP, WMF
	pbyOutGrayImage	out	Gray Image 를 반환 받을 버퍼로 이미지의 가로,세로 크기를 이용하여 메모리 할당 및 초기화된 버퍼이다.
반환값	Error Code (Table 1 참조)		
설 명	입력된 컬러 이미지파일을 이용하여 Gray Scale 의 이미지를 구한다.		

- HS_GetImageDIBSize

  이미지 파일에 해당하는 DIB 이미지 사이즈를 구한다.

```
int HS_GetImageDIBSize(const char *strImagePathName, int *nSize);
```

인 수	변수/값	입/출력	설 명
	strImagePathName	in	DIB 이미지 크기를 구할 이미지 파일명 지원하는 이미지 타입은 BMP, GIF, ICO, CUR, JBG, JPG, JPC, JP2, PCX, PGX, PNG, PNM, RAS, TGA, TIF, WBMP, WMF
	nSize	out	Byte 단위의 이미지 크기
반환값	Error Code (Table 1 참조)		
설 명	주어진 이미지 파일의 DIB 형태의 이미지 크기를 구한다.		

- HS_GetDIBImageFromFile

  이미지 파일로부터 DIB 이미지를 구한다.

```
int HS_GetDIBImageFromFile(const char *strImagePathName,
    BYTE *pbyOutDib);
```

인 수	변수/값	입/출력	설 명
	strImagePathName	in	DIB 형태의 이미지를 구할 이미지 파일 명. 지원하는 이미지 타입은 BMP, GIF, ICO, CUR, JBG, JPG, JPC, JP2, PCX, PGX, PNG, PNM, RAS, TGA, TIF, WBMP, WMF
	pbyOutDib	out	반환되는 DIB 이미지를 저장할 이미지 버퍼로 호출할 때 해당 DIB 이미지 사이즈로 메모리 할당을 반드시 하여야 한다.
반환값	Error Code (Table 1 참조)		

설 명	입력된 이미지 파일로부터 DIB 이미지를 구한다.

- HS_DibImageToGrayImage

  DIB 이미지를 Gray Scale 이미지로 변환 한다.

```
int HS_DibImageToGrayImage(const BYTE *pbyInDibImage, BYTE *pbyOutGrayImage);
```

인 수	변수/값	입/출력	설 명
	pbyInDibImage	in	DIB 형태의 이미지
	pbyOutGrayImage	out	변환된 Gray Scale 이미지를 저장할 버퍼로 호출함수로 부터 메모리 할당을 한 후 사용한다.
반환값	Error Code (Table 1 참조)		
설 명	DIB 이미지를 Gray Scale 이미지로 변환한다.		

- HS_GetImageSizeFromBuffer

  메모리에 할당된 이미지로부터 사이즈를 구한다.

```
int HS_GetImageSizeFromBuffer
(const unsigned char *pbyBuffer, int nBufferLen, int *nImageWidth, int *nImageHeight);
```

인 수	변수/값	입/출력	설 명
	pbyBuffer	in	Jpeg, Bmp, Tif 형태의 이미지로 메모리에 저장된 이미지 버퍼
	nBufferLen	in	메모리에 할당된 이미지 크기
	nImageWidth	out	이미지의 가로 크기
	nImageHeight	out	이미지의 세로 크기
반환값	Error Code (Table 1 참조)		
설 명	메모리에 저장된 이미지로부터 이미지의 가로 및 세로크기를 구한다.		

- HS_GetGrayImageFromBuffer

  메모리에 저장된 이미지를 Gray Scale 로 변환 한다.

int HS_GetGrayImageFromBuffer
(const unsigned char *pbyBuffer, int nBufferLen, BYTE *pbyOutGrayImage);

인 수	변수/값	입/출력	설 명
	pbyBuffer	in	Jpeg, BMP, TIF 형태의 이미지 버퍼
	nBufferLen	in	메모리 이미지의 크기
	pbyOutGrayImage	out	변환될 Gray Scale 의 이미지를 저장할 버퍼
반환값	Error Code (Table 1 참조)		
설 명	메모리버퍼에 저장되어 있는 이미지를 Gray Scale 형태의 이미지로 변환 한다.		

- HS_GetImageDIBSizeFromBuffer

  메모리 버퍼의 이미지로 부터 DIB 형태의 이미지 사이즈를 구한다.

int HS_GetImageDIBSizeFromBuffer(const unsigned char *pbyBuffer, int nBufferLen, int *nSize);

인 수	변수/값	입/출력	설 명
	pbyBuffer	in	Jpeg, bmp, tif 형태의 이미지 메모리 버퍼
	nBufferLen	in	메모리 버퍼에 저장되어 있는 이미지 크기
	nSize	out	DIB 형태의 이미지 크기
반환값	Error Code (Table 1 참조)		
설 명	메모리 버퍼에 저장된 DIB 이미지의 크기를 구한다.		

- HS_GetDIBImageFromBuffer

  메모리 버퍼에 저장된 이미지를 이용하여 DIB 이미지를 구한다.

int HS_GetDIBImageFromBuffer(const unsigned char *pbyBuffer, int nBufferLen, BYTE *pbyOutDib);

인 수	변수/값	입/출력	설 명
	pbyBuffer	in	Jpeg, bmp, tif 형태의 이미지 버퍼
	nBufferLen	in	이미지 버퍼의 이미지 크기
	pbyOutDib	out	DIB 형태의 이미지 출력 버퍼, 호출자에서 메모리 할당 해야 함.
반환값	Error Code (Table 1 참조)		
설 명	메모리 버퍼에 있는 이미지를 DIB 형태의 이미지로 변환 한다.		

- HS_WriteDIBImage
  주어진 DIB 형태의 이미지를 BMP 형태의 이미지로 File Write 한다.

int HS_WriteDIBImage(const unsigned char *pbyDib, const char *szFileName);

인 수	변수/값	입/출력	설 명
	pbyDib	in	DIB 형태의 이미지
	szFileName	out	변환되어 저장되는 이미지 파일 명
반환값	Error Code (Table 1 참조)		
셜 명	DIB 형태의 이미지를 BMP 형태의 이미지 파일로 저장 한 다음 BMP 파일 명을 반환한다.		

- HS_CropFromDIBImage
  DIB 이미지로부터 주어진 영역만큼 이미지를 도려 낸다.

int HS_CropFromDIBImage(const unsigned char *pbyInDib, const HS_RECT *pCropRect, unsigned char *pbyCropDib);

인 수	변수/값	입/출력	설 명
	pbyInDib	in	DIB 형태의 이미지
	pCropRect	in	도려낼 영역
	pbyCropDib	out	도려내진 이미지를 반환할 DIB 형태의 메모리 버퍼
반환값	Error Code (Table 1 참조)		
설 명	DIB 형태 이미지에 주어진 영역만큼의 임지를 도려 낸다.		

- HS_EncodeDIBToJPG
  DIB 이미지 형태를 JPEG 이미지로 변환한다.

int HS_EncodeDIBToJPG(const unsigned char *pbyInDib, unsigned char *pbyOutBuffer, int *nEncodeSize);

인 수	변수/값	입/출력	설 명
	pbyInDib	in	DIB 형태의 이미지
	pbyOutBuffer	out	JPEG 형태의 이미지 버퍼로 호출자가 메모리 버퍼를 할당하여야 하며 메모리 할당 크기는 DIB 형태의 이미지 사이즈 이다.
	nEncodeSize	out	JPEG 이미지 사이즈
반환값	Error Code (Table 1 참조)		
설 명	DIB 형태의 이미지를 JPEG 이미지로 변환 한다.		

- HS_GetJPGBufferDIBSize

  JPEG 메모리 이미지에서 DIB 형태의 이미지 크기를 구한다.

int HS_GetJPGBufferDIBSize(const unsigned char *pbyJPGBuffer, int nJPGSize, int *nDIBSize);

인 수	변수/값	입/출력	설 명
	pbyJPGBuffer	in	Jpeg 형태의 메모리
	nJPGSize	in	Jpeg 형태의 이미지 크기
	nDIBSize	out	DIB 이미지 크기
반환값	Error Code (Table 1 참조)		
설 명	Jpeg 이미지 메모리 버퍼로부터 DIB 이미지 형태의 이미지 크기를 구한다.		

- HS_DecodeJPGToDIB

  Jpeg 형태의 이미지를 DIB 형태의 이미지로 변환

int HS_DecodeJPGToDIB(const unsigned char *pbyJPGBuffer, int nJPGSize, unsigned char *pbyDib);

인 수	변환/값	입/출력	설 명
	pbyJPGBuffer	in	Jpeg 형태의 메모리 버퍼
	nJPGSize	in	Jpeg 형태 이미지의 크기
	pbyDib	out	변환된 DIB 이미지 메모리 버퍼
반환값	Error Code (Table 1 참조)		
설 명	Jpeg 형태의 이미지 버퍼를 DIB 형태의 이미지로 변환 한다.		

3.5 Error Code

아래 테이블 1 은 FaceTech SDK 를 사용하는 동안 발생할 수 있는 ERROR 유형들이다.

Table 1 Table of Error Code

Error definition	Error code	Error description
FT_ERR_NONE	0	No error
FT_ERR_GENERIC	1	Genetic error, no specific meaning
FT_ERR_NODOG	-1	Watch dog not found
FT_ERR_DOGERROR	-2	Error open watch dog
FT_ERR_READDOG	-3	Error read watch dog
FT_ERR_INVALIDDOG	-4	Invalid watch dog
FT_ERR_DOGUSERERROR	-5	Invalid reading user of watch dog
FT_ERR_INVALIDUSER	-6	Invalid user of watch dog
FT_ERR_MOUDLEERROR	-7	Error read module authority
FT_ERR_INVALIDMOUDLE	-8	Module not authority
FT_ERR_DATABASEFULL	-9	Database full, can't enroll any more
FT_ERR_DOGTIMEOUT	-10	Authority timeout
FT_ERR_INVALIDCALL	-99	Invalid input argument
FT_ERR_EXCEPTION	-100	Exception
FT_ERR_CANCELENROLL	9	Cancel the enrollment
FT_ERR_MEMORYALLOC	2	Memory allocation error
FT_ERR_DATACRC	3	Library file corrupted
FT_ERR_GETMODULEPATH	4	Can't get the path of the SDK
FT_ERR_FILEIO	5	File I/O error
FT_ERR_MODENOTMATCH	6	Initialize the SDK with one mode, however, use the function with another mode.

4 FaceTech SDK 사용 방법

4.1 폴더 구성

FaceTech SDK 의 폴더 구성은 다음과 같다.

폴더명	구 성 내 용
include	FaceTech ® SDK 의 Header File 로 구성 됨.
lib	FaceTech ® SDK 의 Library File 들로 구성 됨
doc	SDK 와 관련된 Document
samples	FaceTech ® SDK 의 Sample Code 등으로 구성 됨

4.2 사용방법

Microsoft Visual C++용으로 설명을 하면, Microsoft Visual C++ IDE (통합개발 환경)의 해당 프로젝트의 속성내 상기 "include" Path 와 "lib" Path 를 추가 하여 준다. 그리고 "프로젝트 속성 ➔ 구성속성 ➔ 링커 ➔ 입력 ➔ 추가종속성" 에 해당 Library 명을 설정한다.

4.3 SDK 제약

본 SDK 에서 제시하는 범위 이외의 값을 이용하거나 기타 기능을 수행할 경우 그 정확성은 보장 할 수 없다.

4.4 Sample codes

4.4.1 Sample face detection

```
#include "FaceTech_FaceID.h"
int SampleFaceDetection()
{
    HS_FACEMODE   enMode;              // work mode of face recognition SDK
    int nRetCode;                      // Error Code
```

```
    HS_FaceCandidate   *pFaceCand;          // Candidate of face detection
    int nFaceNum;                           // Number of faces detected
    const char *strImagePath = "d:₩₩sampleimage.jpg";     // image file name
    int nImageWidth, nImageHeight;                        // image width & height
    BYTE *pbyGrayImage = NULL;                        // gray image buffer

    // initialize SDK
    enMode = HS_DBScan;
    nRetCode = HS_InitFaceIDSDK(enMode);
    if (nRetCode != FT_ERR_NONE)
    {
        cout << "Can't initialize FaceTech SDK!" << endl;
        return nRetCode;
    }

    // read image from file, convert it to gray image
    nRetCode = HS_GetImageSize(strImagePath, &nImageWidth, &nImageHeight);
    pbyGrayImage = new BYTE[nImageWidth * nImageHeight];
    nRetCode = HS_GetGrayImageFromFile(strImagePath, pbyGrayImage);

    pFaceCand = new HS_FaceCandidate[100];          // allocate memory for face detection result

    // face detection
    nRetCode = HS_FaceDetection(pbyGrayImage, nImageWidth, nImageHeight, 20,
        min(nImageWidth, nImageHeight), pFaceCand, &nFaceNum);

    delete[] pbyGrayImage;
    delete[] pFaceCand;

    // terminate the SDK
    nRetCode = HS_UninitFaceIDSDK();

    return nRetCode;
}
```

4.4.2 Sample eye localization

```
#include "FaceTech_FaceID.h"
int SampleEyeLocation()
{
    HS_FACEMODE   enMode;           // work mode of face recognition SDK
    int nRetCode;                   // Error Code
```

```
HS_FaceCandidate   *pFaceCand;          // Candidate of face detection
int nFaceNum;                           // number of faces detected
const char *strImagePath = "d:₩₩sampleimage.jpg";    // face image file name
int nImageWidth, nImageHeight;                       // image width & height
BYTE *pbyGrayImage = NULL;                      // gray image buffer
HS_EyePoint eyePoint;                           // Eye position

// Initialize SDK
enMode = HS_DBScan;
nRetCode = HS_InitFaceIDSDK(enMode);
if (nRetCode != FT_ERR_NONE)
{
    cout << "Can't initialize FaceTech SDK!" << endl;
    return nRetCode;
}

// read image from file, convert image to gray image
nRetCode = HS_GetImageSize(strImagePath, &nImageWidth, &nImageHeight);
pbyGrayImage = new BYTE[nImageWidth * nImageHeight];
nRetCode = HS_GetGrayImageFromFile(strImagePath, pbyGrayImage);

pFaceCand = new HS_FaceCandidate[100];          // allocate memory for face
detection

// face detection
nRetCode = HS_FaceDetection(pbyGrayImage, nImageWidth, nImageHeight, 20,
    min(nImageWidth, nImageHeight), pFaceCand, &nFaceNum);

// eye localization for each detected face
for (int i=0; i<nFaceNum; ++i)
{
    HS_RECT rcFace;
    rcFace.left = pFaceCand[i].trueleft;
    rcFace.top = pFaceCand[i].truetop;
    rcFace.right = pFaceCand[i].trueright;
    rcFace.bottom = pFaceCand[i].truebottom;
    nRetCode = HS_LocateEye(pbyGrayImage, nImageWidth, nImageHeight,
        &rcFace, &eyePoint);
}
```

```
    delete[] pbyGrayImage;
    delete[] pFaceCand;

    // terminate the SDK
    nRetCode = HS_UninitFaceIDSDK();

    return nRetCode;
}
```

4.4.3 Sample feature extraction

```
#include "FaceTech_FaceID.h"
int SampleFeatExtract()
{
    HS_FACEMODE    enMode;              // work mode of face recognition SDK
    int nRetCode;                       // Error code
    HS_FaceCandidate    *pFaceCand;          // candidate of face
    int nFaceNum;                       // number of faces detected
    const char *strImagePath = "d:₩₩sampleimage.jpg";    // face image file name
    int nImageWidth, nImageHeight;                       // image width & height
    BYTE *pbyGrayImage = NULL;                  // buffer for gray image
    HS_EyePoint eyePoint;                       // eye position
    int nFeatSize;                      // face features size in bytes
    BYTE *pbyFeatures = NULL;      // feature buffer

    // initialize SDK
    enMode = HS_DBScan;
    nRetCode = HS_InitFaceIDSDK(enMode);
    if (nRetCode != FT_ERR_NONE)
    {
        cout << "Can't initialize FaceTech SDK!" << endl;
        return nRetCode;
    }

    // Get face features size in bytes
    nFeatSize = HS_GetFaceFeatureSize();

    // read image from file, convert image to gray image
    nRetCode = HS_GetImageSize(strImagePath, &nImageWidth, &nImageHeight);
```

```
    pbyGrayImage = new BYTE[nImageWidth * nImageHeight];
    nRetCode = HS_GetGrayImageFromFile(strImagePath, pbyGrayImage);

    pFaceCand = new HS_FaceCandidate[100];          // allocate memory for face
detection

    // face detection
    nRetCode = HS_FaceDetection(pbyGrayImage, nImageWidth, nImageHeight, 20,
        min(nImageWidth, nImageHeight), pFaceCand, &nFaceNum);

    // eye localization for each face detected
    for (int i=0; i<nFaceNum; ++i)
    {
        HS_RECT rcFace;
        rcFace.left = pFaceCand[i].trueleft;
        rcFace.top = pFaceCand[i].truetop;
        rcFace.right = pFaceCand[i].trueright;
        rcFace.bottom = pFaceCand[i].truebottom;
        nRetCode = HS_LocateEye(pbyGrayImage, nImageWidth, nImageHeight,
            &rcFace, &eyePoint);

        // feature extraction
        pbyFeatures = new BYTE[nFeatSize];
        nRetCode = HS_FaceExtractionByEyepos(pbyGrayImage, nImageWidth,
            nImageHeight, &eyePoint, pbyFeatures);

        delete[] pbyFeatures;
    }

    delete[] pbyGrayImage;
    delete[] pFaceCand;

    // terminate the SDK
    nRetCode = HS_UninitFaceIDSDK();

    return nRetCode;
}
```

4.4.4 Sample verification

```
#include "FaceTech_FaceID.h"
int SampleVerify()
{
    HS_FACEMODE   enMode;          // work mode of face recognition SDK
    int nRetCode;                  // error code
    HS_FaceCandidate   *pFaceCand;     // candidate of face detection
    int nFaceNum;                  // face number detected
    const char *strImagePath = "d:₩₩sampleimage.jpg"; // gallery file name
    const char *strTestImagePath = "d:₩₩testimage.jpg";     // probe file name
    int nImageWidth, nImageHeight;                  // image width & height
    BYTE *pbyGrayImage = NULL;                  // gray image buffer
    HS_EyePoint eyePoint;                  // eye position
    int nFeatSize;                 // face feature size in bytes
    BYTE *pbyFeatures = NULL;      // feature buffer
    int nGalleryNum = 0;           // number of gallery
    long lUserID = 0;              // user ID

    // initialize SDK
    enMode = HS_Verify;
    nRetCode = HS_InitFaceIDSDK(enMode);
    if (nRetCode != FT_ERR_NONE)
    {
        cout << "Can't initialize FaceTech SDK!" << endl;
        return nRetCode;
    }

    // get face feature size
    nFeatSize = HS_GetFaceFeatureSize();

    // read image from file, convert it to gray image
    nRetCode = HS_GetImageSize(strImagePath, &nImageWidth, &nImageHeight);
    pbyGrayImage = new BYTE[nImageWidth * nImageHeight];
    nRetCode = HS_GetGrayImageFromFile(strImagePath, pbyGrayImage);

    pFaceCand = new HS_FaceCandidate[100];      // allocate memory
```

```
// face detection
nRetCode = HS_FaceDetection(pbyGrayImage, nImageWidth, nImageHeight, 20,
      min(nImageWidth, nImageHeight), pFaceCand, &nFaceNum);

// eye localization
if (nFaceNum > 0)
{
      HS_RECT rcFace;
      rcFace.left = pFaceCand[0].trueleft;
      rcFace.top = pFaceCand[0].truetop;
      rcFace.right = pFaceCand[0].trueright;
      rcFace.bottom = pFaceCand[0].truebottom;
      nRetCode = HS_LocateEye(pbyGrayImage, nImageWidth, nImageHeight,
            &rcFace, &eyePoint);

      // feature extraction
      pbyFeatures = new BYTE[nFeatSize];
      nRetCode = HS_FaceExtractionByEyepos(pbyGrayImage, nImageWidth,
            nImageHeight, &eyePoint, pbyFeatures);

      // enroll gallery into memory
      nRetCode = HS_LoadNumIDPopulation(lUserID++, pbyFeatures);
      ++nGalleryNum;

      delete[] pbyFeatures;
}

delete[] pbyGrayImage;

// Verification
float fScore;            // match score
long lClaimedID = 0;  // claimed ID
// read image from file, convert it to gray image
nRetCode = HS_GetImageSize(strTestImagePath, &nImageWidth, &nImageHeight);
pbyGrayImage = new BYTE[nImageWidth * nImageHeight];
nRetCode = HS_GetGrayImageFromFile(strTestImagePath, pbyGrayImage);

// face detection
```

```
    nRetCode = HS_FaceDetection(pbyGrayImage, nImageWidth, nImageHeight, 20,
        min(nImageWidth, nImageHeight), pFaceCand, &nFaceNum);

    if (nFaceNum > 0)
    {
        HS_RECT rcFace;
        rcFace.left = pFaceCand[0].trueleft;
        rcFace.top = pFaceCand[0].truetop;
        rcFace.right = pFaceCand[0].trueright;
        rcFace.bottom = pFaceCand[0].truebottom;

        // eye localization
        nRetCode = HS_LocateEye(pbyGrayImage, nImageWidth, nImageHeight,
            &rcFace, &eyePoint);

        // feature extraction
        pbyFeatures = new BYTE[nFeatSize];
        nRetCode = HS_FaceExtractionByEyepos(pbyGrayImage, nImageWidth,
            nImageHeight, &eyePoint, pbyFeatures);

        // face verification
        nRetCode = HS_VerifyWithNumID ( pbyFeatures, 1, lClaimedID, &fScore);

        delete[] pbyFeatures;
    }

    delete[] pbyGrayImage;
    delete[] pFaceCand;

    // terminate SDK
    nRetCode = HS_UninitFaceIDSDK();

    return nRetCode;
}
```

- End Of Document –