Confidential

OMRON

OKAO Vision™ SOFTWARE LIBRARY

SOFTWARE SPECIFICATIONS(V4.6: for Evaluation)

OMRON Corporation

OKAO, OKAO Vision are registered trademarks of OMRON Corporation in Japan.

Microsoft product names such as Microsoft and Windows in this specification are the trademarks or registered trademarks of Microsoft Corporation. Other company names and product names are likewise trademarks or product names of their respective companies.

Functions name and Parameters

● Face detection

Function name	:	OKAO_Detection()	
Prefix	:	DT	
Handle name	:	HDETECTION	Face detection handle
Result	:	HDTRESULT	Result of face detection handle

● Facial parts detection

Function name	:	OKAO_Pointer()	
Prefix	:	PT	
Handle name	:	HPOINTER	Facial parts detection handle
Result	:	HPTRESULT	Result of facial parts detection handle

● Face recognition

Function name	:	OKAO_FrIdentify()	
Prefix	:	FR	
Handle name	:	HFACERECOG	Face recognition handle
Result	:	HFRRESULT	Result of face recognition handle
Prefix :	:	DB	
Handle name	:	HALBUMDB	Database handle

● Gender estimation

Function name	:	OKAO_GenderEstimate()	
Prefix	:	GE	
Handle name	:	HGENDER	Gender estimation handle
Result	:	HGENRESULT	Result of gender estimation handle

● Age group estimation

Function name	:	OKAO_AgeEstimate()	
Prefix	:	AGE	
Handle name	:	HAGE	Age group estimation handle
Result	:	HAGERESULT	Result of age group estimation handle

● Age estimation

Function name	:	OKAO_YearEstimate()	
Prefix	:	YE	
Handle name	:	HYEAR	Age estimation handle
Result	:	HYEARRESULT	Result of age estimation handle

● Smile degree estimation

Function name	:	OKAO_SmileCheck()	
Prefix	:	SMILE	
Handle name	:	HSMILE	Smile degree estimation handle

● Facial feature outline detection

Function name	:	OKAO_Contour()	
Prefix	:	CT	
Handle name	:	HCONTOUR	Facial feature outline detection handle
Result	:	HCTRESULT	Result of facial feature outline detection handle

CONTENTS

1. Overview of the library

1.1 Overview

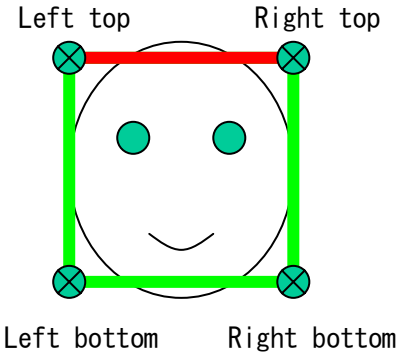
OKAO Vision™ Software Library (The Library) has implemented a number of features on image data that was input from the application side. Those features include face detection, facial parts detection, face recognition, estimation of gender, age group, or age, smile degree estimation, and detection of facial-feature outline. The Library runs on the Windows platform of PC/AT-compatible machines.

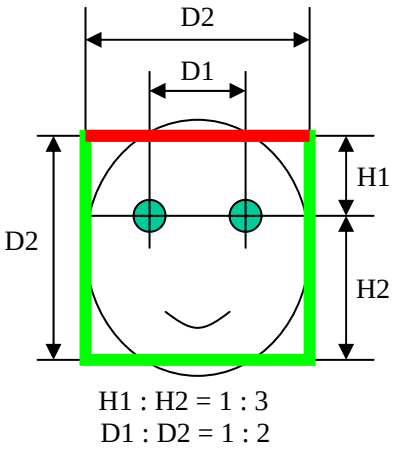
1.2 Function

The Library offers the following features.

1.2.1 Face detection

After having the image data as input, you can detect the position of the face from it. Then you can output the number of faces detected and individual rectangular coordinates (coordinates of endpoints for the face rectangle). As seen in the lower left diagram, for the facial rectangular coordinates, you can output the four square coordinates on the left top, right top, left bottom, and right bottom of the face. The ratios to the eye position and eye width (D1) are given in the right bottom diagram. For details, refer to the explanation on functions in OKAO_GetDtCorner() (the ratio values should be regarded as rules of thumb).





The Library has the following features for face detection:

The size of the face ranging from 20 pixels① to the short side of the image can be detected. Faces with rotating angles up to 60°horizontally and up to 30°vertically to the frontal view② can be detected

Multiple faces in the same image can be detected.

Faces with all angles from 0°to 360°can be detected.

The processing time can be reduced by using the peripheral mask and color masks.

Even under the above conditions, the face images may sometimes not be detectable.

① As the size decreases below 20 pixels, the detection performance will decline.

② As the angle to the front view increases, the detection performance will decline.

1.2.2 Facial parts detection (including facial pose estimation, gaze estimation, open-close estimation)

Using the information on the detected rectangular area of the face and other information from face detection, one may detect the position of specified facial features. The Library has the following features for facial parts detection:

Note: On this document, “Left eye” means the eye which X position is lower than another (based on straight forward position). So for person, this is right eye.

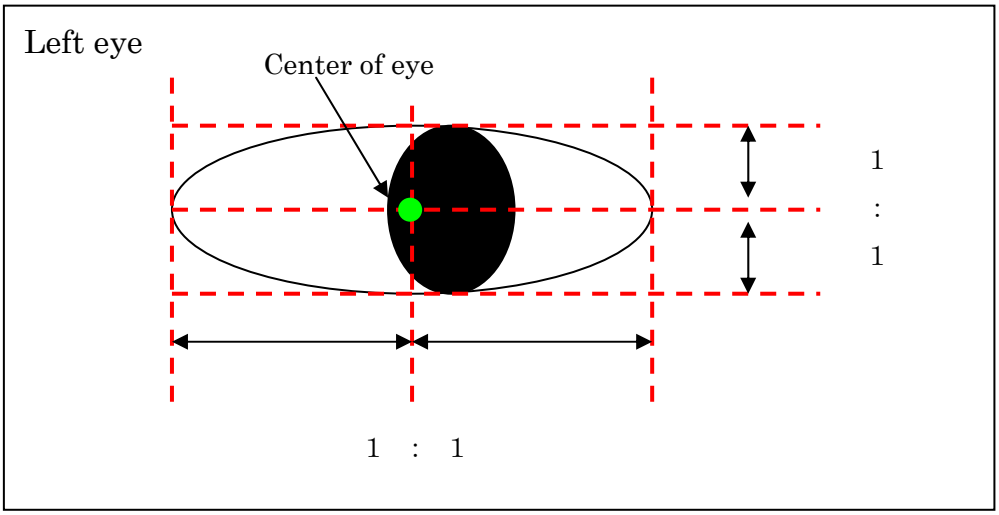
A) Detection of the Facial Parts Central Points

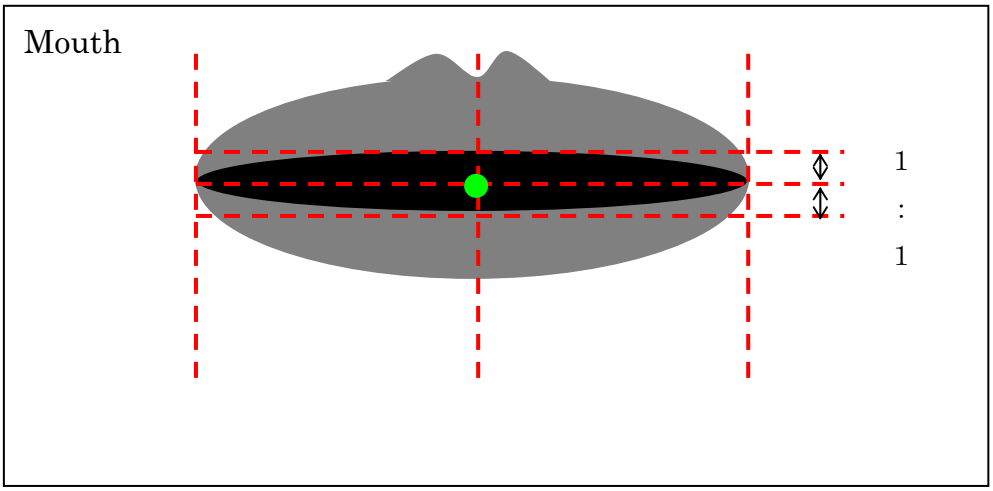
Using the information on the detected rectangular area of the face and other information from face detection, the three central points near the position of the eyes and mouth of the specified face can be detected.

The center of the eye refers to the point connecting the midpoint between the outer and inner corners of the eye and the midpoint between the upper edge and lower edge of the eye (not the center of the pupil); the midpoint of the mouth refers to the point connecting the center of the lips and the midpoint between the lower edge of the upper lip and upper edge of the lower lip.

Facial features up to 40 pixels in one side of the detected image can be detected. Facial features with rotating angles up to 60°horizontally and up to 30°vertically to the frontal view can be detected.③

Even if these points can't be seen, the positions will be estimated from other information.





③ With 40 pixels or smaller, the detection performance will decline. Also, facial parts detection may not be performed correctly due to the shooting condition of facial images.

B) Detection of Facial Parts Endpoints

Using the information on the detected rectangular area of the face and other information from face detection, the six central points near the positions of the outer and inner corners of the eyes and mouth of the specified face can be detected.

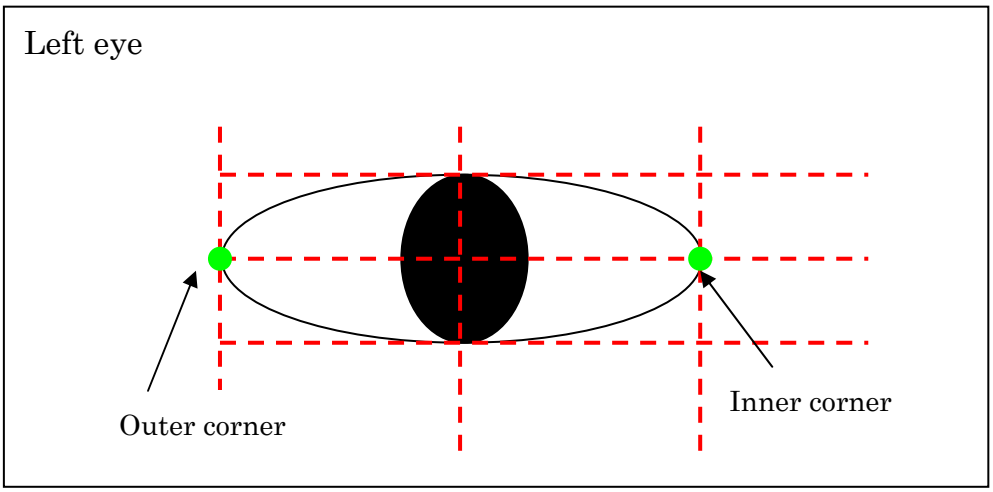
The outer corner of the eye refers to the edge of either eye closer to the ear; the point intersecting the upper side and lower side of the boundary between the whites of the eyes and the skin.

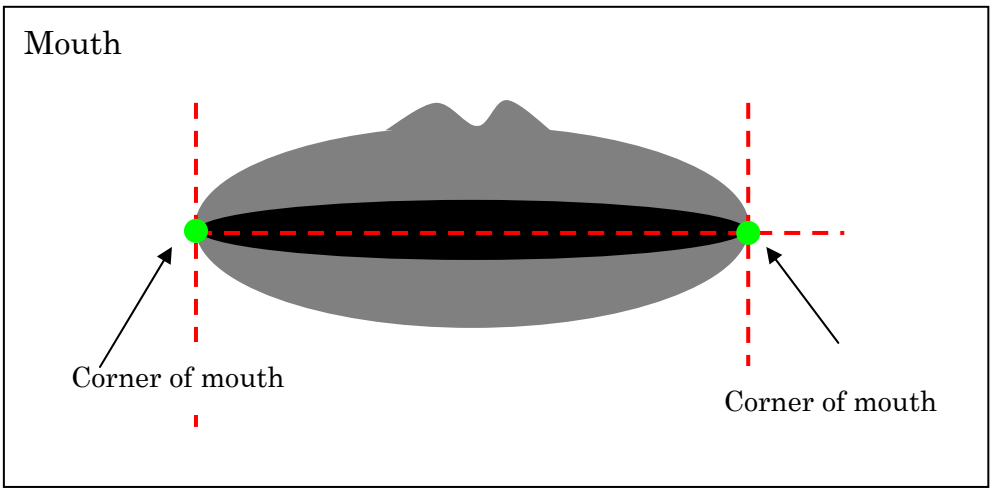
The inner corner (canthus) of the eye refers to the point intersecting the upper and lower eyelids.

The mouth refers to the point intersecting the upper side of the upper lip and the lower side of the lower lip.

Facial features up to 40 pixels in one side of the detected image can be detected. Facial features with rotating angles up to 60°horizontally and up to 30°vertically to the frontal view can be detected.④

Even if these points can't be seen, the positions will be estimated from other information.





④ With 40 pixels or smaller, the detection performance will decline. Also, facial parts detection may not be performed correctly due to the shooting condition of facial images.

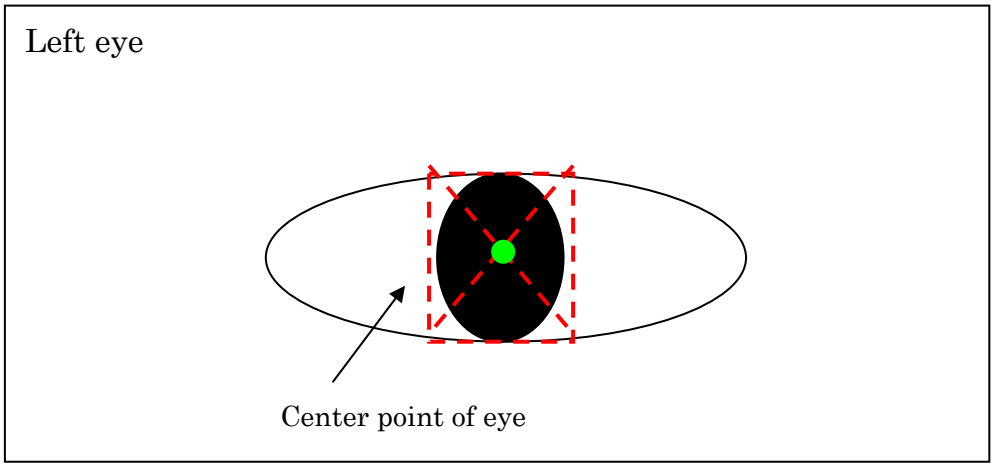
C) Detection of the Pupil Center

Using the information on the detected rectangular area of the face and other information from face detection, one may detect the area near the center of the pupil of a specified face.

The center of the pupil refers to the central point when visualizing a square surrounding the pupilary area (the pupil and the iris).

Facial features up to 80 pixels in one side of the detected image can be detected. Facial features with rotating angles up to 30°horizontally and up to 15°vertically to the frontal view can be detected.⑤

Even if these points can't be seen, the positions will be estimated from other information.



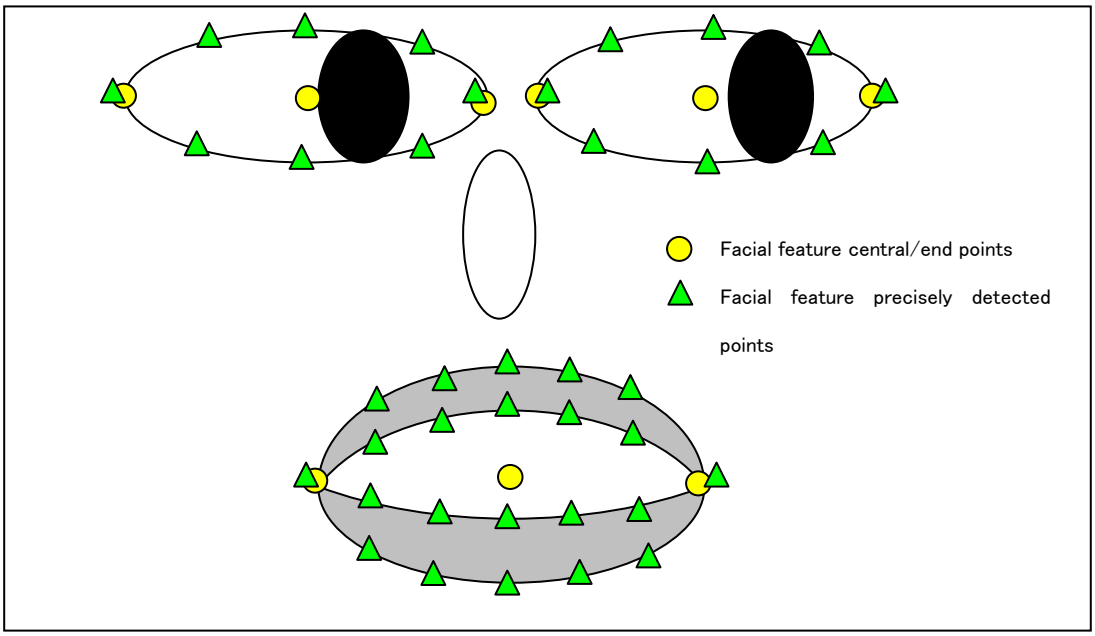
⑤ With 80 pixels or smaller, the detection performance will decline. Also, facial parts detection may be performed incorrectly due to the shooting condition of facial images.

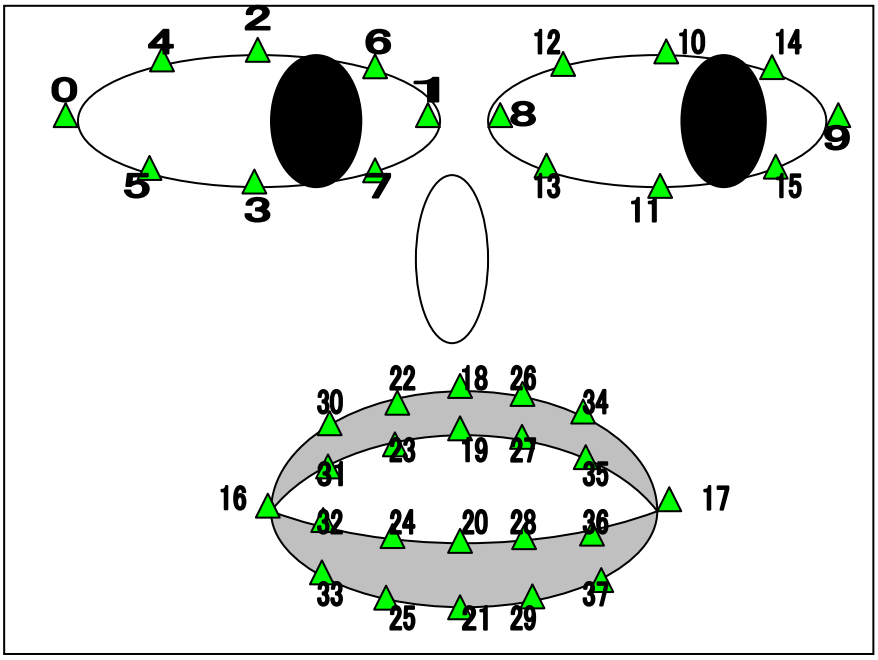
D) Facial Parts Precisely Detection

Using the information on the detected rectangular area of the face and other information from face detection, many positions of the specified face can be detected.

Even if these points can't be seen, the positions will be estimated from other information.

Facial features up to 40 pixels in one side of the detected image can be detected. Facial features with rotating angles up to 60°horizontally and up to 30°vertically to the frontal view can be detected.⑥





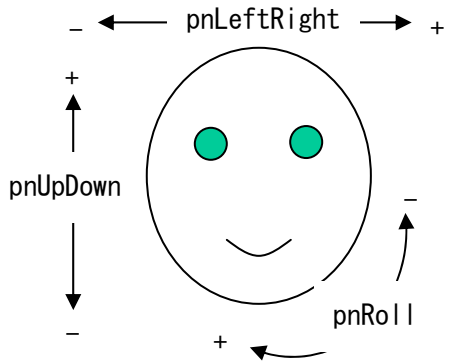
⑥ With 40 pixels or smaller, the detection performance will decline. Also, facial parts detection may not be performed correctly due to the shooting condition of facial images.

E) Facial Pose Estimation

Using the information on the detected rectangular area of the face and other information from face detection, direction of face can be estimated.

Result will be composed of three degrees (up-down, left-right, roll).

Face directions up to 40 pixels in one side of the detected image can be detected. Face directions with rotating angles up to 60°horizontally and up to 30°vertically to the frontal view can be detected.⑦



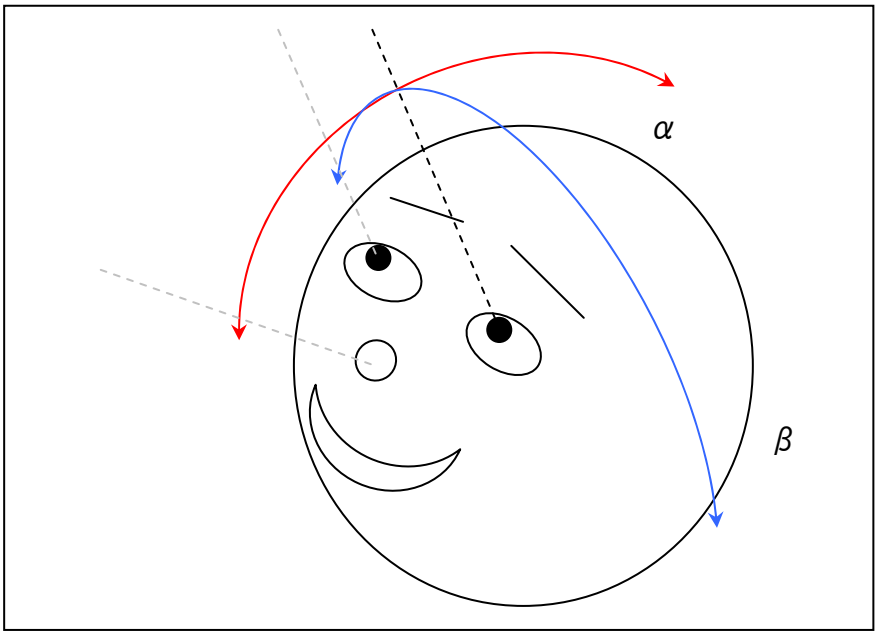
⑦ With 40 pixels or smaller, the detection performance will decline. Also, facial parts detection may not be performed correctly due to the shooting condition of facial images.

F)Gaze estimation

Using the information on the detected rectangular area of the face and other information from face detection, angle of the gaze can be detected. Angle is including face direction.

Output is the angle of both eyes' focus.

Gaze up to 80 pixels in one side of the detected image can be estimated. Gaze and face with rotating angles up to 30°horizontally and up to 15°vertically to the frontal view can be estimated.⑧



⑧ With 80 pixels or smaller, the detection performance will decline. Also, facial parts detection may be performed incorrectly due to the shooting condition of facial images.

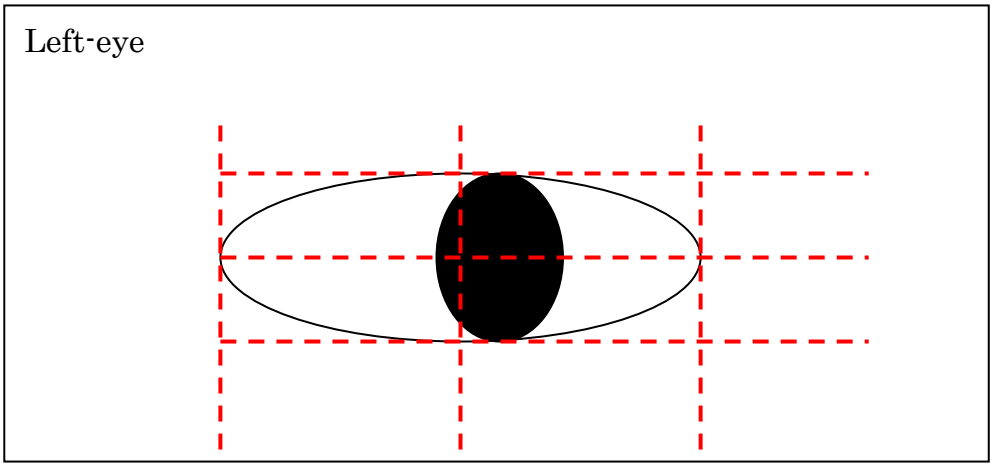
G)Open-close of the Facial Parts Estimation

Open-close level and confidence of the left eye/right eye/mouth are output.

Open-close level is shown as 0 to 1000 integer, based on height divided by width.

1000 is the maximum value of extreme case, so you can find “open” if open-close grade is over 250.

Open-close of the faces up to 80 pixels in one side of the detected image can be estimated. Open-close of the faces with rotating angles up to 30°horizontally and up to 15°vertically to the frontal view can be estimated.⑨



⑨ With 80 pixels or smaller, the detection performance will decline. Also, facial parts detection may be performed incorrectly due to the shooting condition of facial images.

1.2.3 Face recognition

By extracting individual facial characteristics from the position of facial features and other information detected by facial parts detection and then comparing it with previously registered data, it is possible to distinguish (recognize from unspecified persons).
Faces with angles up to 30° vertically and 45° horizontally to the frontal view can be recognized.⑩
However, as a rule, the Library does not perform recognition on small-sized faces.

If version of The Library changed, registered data of previous version may not be used. So we recommend saving original images to make data at new version.

⑩ Recognition may be performed incorrectly due to the shooting condition of facial images.

1.2.4 Gender estimation

By extracting facial gender characteristics from the positions of facial features and other information detected by facial parts detection, the gender of that person can be estimated.

The gender of faces with rotating angles up to 30° horizontally and 20° vertically to the frontal view can be estimated.

*Now this is only for Japanese.

*When his/her age is very low, estimation may be performed incorrectly.

1.2.5 Age group estimation

By extracting facial age group characteristcs from the positions of facial features and other information detected by facial parts detection, the age group of that person can be roughly estimated.

The age group of faces with rotating angles up to 30° horizontally and 20° vertically to the frontal view can be estimated.

*Now this is only for Japanese.

1.2.6 Age estimation

By extracting facial age characteristcs from the positions of facial features and other information detected by facial parts detection, the age of that person can be roughly estimated.

The age of faces with rotating angles up to 30° horizontally and 20° vertically to the frontal view can be estimated.

*Now this is only for Japanese.

1.2.7 Smile degree estimation

By extracting facial characteristics from the positions of facial features and other information detected by facial parts detection, the smile rate can be measured.

The smile rate of the faces up to 60 pixels in one side of the detected image can be measured.The smile rate of faces with rotating angles up to 30° horizontally and 15° vertically to the frontal view can be measured.⑪

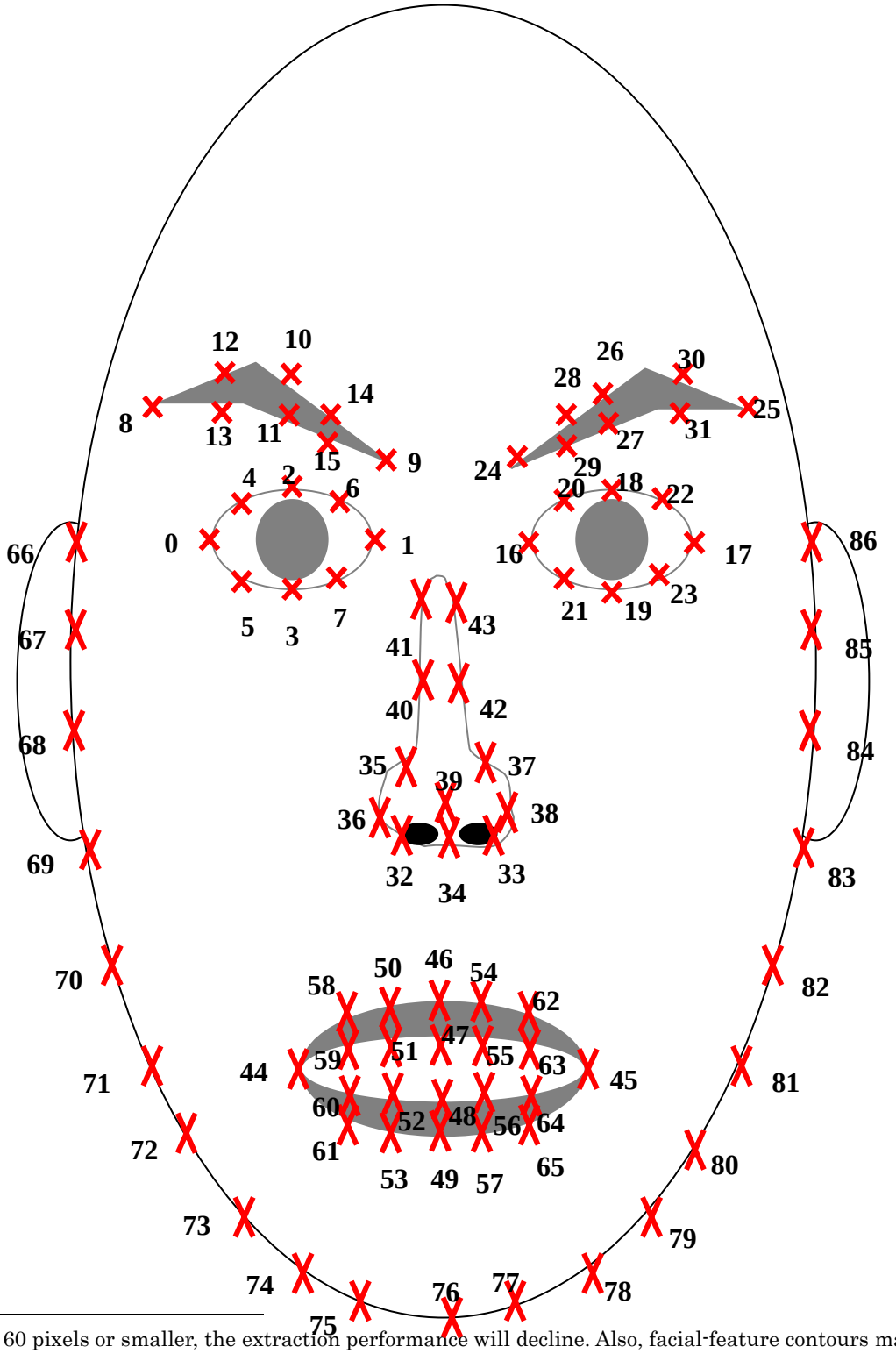
⑪ As the size decreases, the estimation performance will decline. Estimation may be performed incorrectly due to the shooting condition of facial images.

1.2.8 Facial feature outline detection

From the facia parts position obtained by facial parts detection, one may extract the outline position of facial features (eye, eyebrow, nose, mouth, and facial shape) of that face. A facial-feature outline with a space between the eyes up to 60 pixels can be detected as a result of facial-feature position detection.

The facial-feature contour can also be extracted for faces with rotating angles up to 5°both horizontally and vertically to the frontal view.⑫

The positions of facial-feature contours extractable in this version are as follows.



⑫ With 60 pixels or smaller, the extraction performance will decline. Also, facial-feature contours may be detected incorrectly due to the shooting condition of facial images.

2. System Requirements

2.1 Computer Requirements

OS	It has been verified under Microsoft Windows 2000/XP(Japanese/English).
CPU	PentiumⅢ 700MHz or faster[⑬] are recommended.
Memory	128MB or greater are recommended.[⑭]
Development Environment	It has been verified with Japanese Microsoft Visual C++6.0 SP5.

2.2 Input Image Requirements

①Input Image Size	<Range of Compatibility> "Still Image" Vertical: 30～1024 pixels, Horizontal: 30～1280 pixels "Movie Image" Vertical: 30～1024 pixels, Horizontal: 30～1280 pixels
②Input Face Size	<Range of Compatibility> Minimum: face size of 20 pixels[⑮] Maximum: less than the short side (pixels) of the face-width input image
③Format	8-bit grayscale image and 24-bit RGB image in the RAW image format (the sequence of 24-bit RGB image data is in BGRBGR order)

⑬ It varies depending on the image size to input and the application requirements as the source of invocation.
⑭ It varies depending on the image size to input and the application requirements as the source of invocation.
⑮ With 20 pixels or smaller, the detection performance will decline.

3. Software

3.1 Development Environment

The Library is developed by the C Language and premised to be called up from C/C++.

3.2 Function Specifications

3.2.1 Common functions

● **Version acquisition**

int OKAO_GetVersion(BYTE *pbyMajor, BYTE *pbyMinor)

Argument	Output : pbyMajor Major Version pbyMinor Minor Version
Return value	Error Code (Refer to Table 1)
Description	Acquisition of the Library Version

Version	pbyMajor	pbyMinor
V1.0	1	0
V1.1	1	1
V2.0	2	0
V2.1	2	1
V3.0	3	0
V3.2	3	2
V3.3	3	3
V3.4	3	4
V4.0	4	0
V4.1	4	1
V4.2	4	2
V4.3	4	3
V4.5	4	5
V4.6	4	6

int OKAO_GetDetailVersion(DWORD dwMode, BYTE *pVersionString)

Argument	Input : dwMode Selection of the function to get a version Output : pVersionString Version String
Return value	Error Code (Refer to Table 1)
Description	Acquisition of the Version of each function. function OKAO_COMMON Common OKAO_DETECTION Face detection OKAO_POINTER Facial parts detection OKAO_RECOGNITION Face recognition OKAO_GENDER Gender estimation OKAO_AGE Age group estimation OKAO_YEAR Age estimation As for pVersionString, it is necessary to secure the buffer for 50 characters before calling it up.

● Library initialization/Termination

`int OKAO_Initialize(DWORD dwMode)`

Argument	Input：dwMode　Designation of the Library to Initialize
Return value	Error Code (Refer to Table 1)
Description	The library to be used can be initialized. mode OKAO_DETECTION　Face detection OKAO_POINTER　Facial parts detection OKAO_RECOGNITION　Face recognition OKAO_GENDER　Gender estimation OKAO_AGE　Age group estimation OKAO_YEAR　Age estimation OKAO_SMILE　Smile degree estimation OKAO_CONTOUR　Facial feature outline detection OKAO_ALL　select all function For multiple designations, specify with OR（|）. For all designation, you can specify OKAO_ALL. For the initialization function, you can execute the securing of data area used for internal processing, data read, and calculations that can be preprocessed. By specifying only the libraries you use, you can save memory and initialization time to a certain degree.
Restriction	Please call this function only once at same mode within a process (even if you use it by the multithread). When it calls this two times or more at same mode, it does not operate normally. We don't assure if you set dwMode with other numbers which does not OR of OKAO_*.

`int OKAO_Terminate(DWORD dwMode)`

Argument	Input：dwMode　Designation of the Library to Terminate
Return value	Error Code (Refer to Table 1)
Description	Termination process of the specified library can be performed. The mode is the same as OKAO_Initialize(). ※Before calling up OKAO_Terminate(), delete the various handles created in The Library. Not doing so may cause a memory leak.
Restriction	Please call this function only once at same mode within a process (even if you use it by the multithread). When it calls this two times or more at same mode, it does not operate normally. We don't assure if you set dwMode with other numbers which does not OR of OKAO_*.

● Setting/Acquisition of Result handle data

`int OKAO_GetResultSize(void *hResult, int *plSize)`

Argument	Input：hResult Result handle Outpu：plSize Size of Result handle data
Return value	Error Code (Refer to Table 1)
Description	It is possible to obtain the result handle data size obtainable by OKAO_GetResultData(). Result handle HDTRESULT Result of face detection handle HPTRESULT Result of facial parts detection handle If result handle has no data, zero will be returned.
Restriction	

`int OKAO_GetResultData(void *hResult, BYTE *pbyBuffer)`

Argument	Input：hResult Result handle Output：pbyBuffer Result data buffer
Return value	Error Code (Refer to Table 1)
Description	Data necessary to save result handle data can be obtained. It is necessary to secure a result data buffer for OKAO_GetResultSize() before calling it up. Result handle HDTRESULT Result of face detection handle HPTRESULT Result of facial parts detection handle
Restriction	

`int OKAO_SetResultData(void *hResult, BYTE *pbyBuffer)`

Argument	Output：hResult Result handle Input：pbyBuffer Result data buffer
Return value	Error Code (Refer to Table 1)
Description	It is possible to configure the binary data to be obtained by OKAO_GetResultData() into Result handle.
Restriction	It may not be possible to configure binary data that have been obtained in different versions.

3.2.2 Face detection

● Create/Delete Face detection handle

```
HDETECTION OKAO_CreateDetection(enum DT_MODE mode)
```

Argument	Input : mode	Face detection mode
Return value	! NULL : Face detection handle , NULL : false	
Description	Handles for the face detection module can be created. The face detection modes you can set are as follows: DT_MODE_DEFAULT （default） ※After using the handles, call up OKAO_DeleteDetection() and delete them.	

```
int OKAO_DeleteDetection(HDETECTION hDT)
```

Argument	Input : hDT	Face detection handle to delete
Return value	Error Code (Refer to Table 1)	
Description	The handles created by OKAO_CreateDetection() can be deleted.	

```
HDTRESULT OKAO_CreateDtResult()
```

Argument	None
Return value	! NULL : Face detection result handle , NULL : false
Description	Handles for face detection result storage can be created. ※After using the handles, call up OKAO_DeleteDtResult() and delete them.

```
int OKAO_DeleteDtResult(HDTRESULT hResult)
```

Argument	Input : hResult	Face detection result handle to delete
Return value	Error Code (Refer to Table 1)	
Description	Handles created by OKAO_CreateDtResult() can be deleted.	

● Face detection

```
int OKAO_Detection(HDETECTION hDT, RAWIMAGE *pImage, int nWidth,
        int nHeight, int nDepth, HDTRESULT hResult)
```

Argument	Input : hDT pImage nWidth nHeight nDepth Output : hResult	Face detection handle Input Image data Image width（pixel） Image height（pixel） Image depth（8bit gray scale:1 , 24bit RGB:3） Face detection result handle
Return value	Error Code (Refer to Table 1)	
Description	Face detection can be conducted from the input image and the result stored in hResult. Face detection can be conducted on a RAW image (8-bit grayscale or 24-bit RGB). The case of 24-bit RGB is compatible only with the point-sequence format in BGR order.	
Restriction	The processible input image format has integers 30～1280 for its width, 30～1024 for its height, and only 1 or 3 for its depth.	

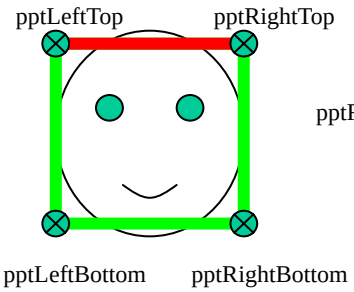
● Result of face detection

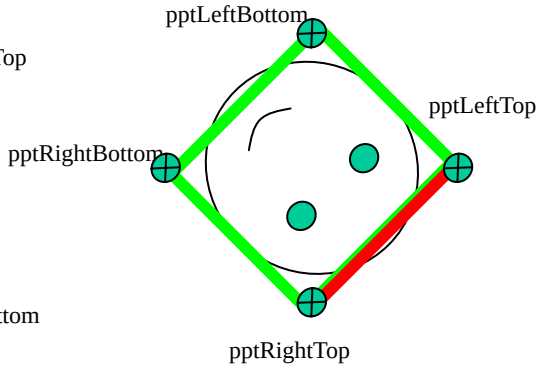
`int OKAO_GetDtFaceCount(HDTRESULT hResult, int *pnCount)`

Argument	Input : hResult	Face detection result handle
	Output : pnCount	Face detection count
Return value	Error Code (Refer to Table 1)	
Description	The number of faces detected by OKAO_Detection() can be obtained.	

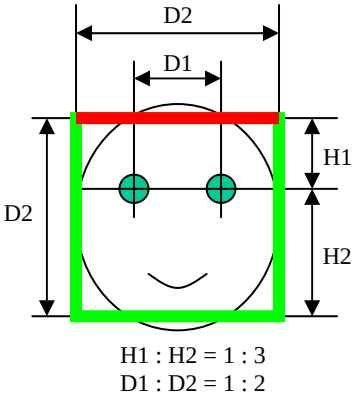
`int OKAO_GetDtCorner(HDTRESULT hResult, int nIndex, POINT *pptLeftTop, POINT *pptRightTop, POINT *pptLeftBottom, POINT *pptRightBottom, int *pnConfidence)`

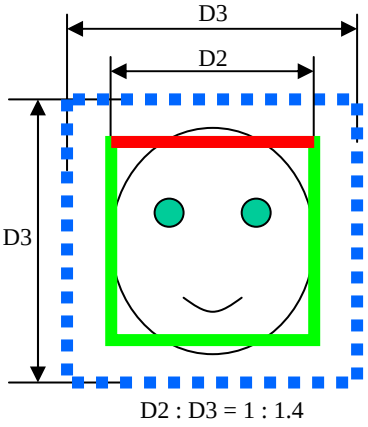
Argument	Input : hResult	Face detection result handle
	nIndex	Index of Faces to be obtained
	Output : pptLeftTop	Left top coordinate of the face
	pptRightTop	Right top coordinate of the face
	pptLeftBottom	Left bottom coordinate of the face
	pptRightBottom	Right bottom coordinate of the face
	pnConfidence	Reliability of the face detected (integers 0 – 1000)
Return value	Error Code (Refer to Table 1)	
Description	The four coordinates of the frame surrounding the detected face can be obtained. Depending on the inclination of the face, the frame will also have four inclined coordinates.	





The face rectangle (D2xD2) has ratios to the eye position and eye width (D1) as given in the right bottom diagram. The image area necessary for face detection will have a broken-line size (D3xD3) in relation to the face rectangle (D2xD2). If this area, even part of it, is located outside the image, one may have trouble detecting the face (the ratio values should be regarded as rules of thumb).





As for obtaining the face rectangle, the smaller nIndex is, the larger the rectangle will be (it has been sorted based on the distance between pptLeftTop and pptRightTop). The coordinate value may be located outside the image depending on the detected position.

	Example) If the face is detected near the left edge of the image, the X-coordinate value on the left of the face rectangle can be negative. If the face is detected near the right edge of the image, the X-coordinate value of the face rectangle can be larger than the width of the image. Likewise, the Y-coordinate can have a negative value near the upper edge of the image while it can have a larger value than the height of the image near the lower edge of the image.
Restriction	nIndex is 0 ～ integers less than the number of detections.

`int OKAO_GetDtFacePose(HDTRESULT hResult, int nIndex, int *pnPose)`

Argument	Input：hResult　Face detection result handle nIndex　Index of detected faces Output：pnPose　Information of Detected Face pose
Return value	Error Code (Refer to Table 1)
Description	A certain degree of face angle information can be returned on the detected face. In this version, the following numerical values will be returned. -45: Left face pose 45 0: Frontal face（left-right 30） 45: Right face pose 45
Restriction	If the face detection face angle is set as DETECT_PROFILE by OKAO_SetPose(), all values will be returned, DETECT_HALF_PROFILE by OKAO_SetDtPose(), -45 or 0 or 45 will be returned. In the case of DETECT_FRONT, only 0 will be returned.

● Setting/Acquisition of the Maximum Number of Face Detections

`int OKAO_SetDtMaxFaceNumber(HDETECTION hDT, int nMax)`

Argument	Input：hDT nMax	Face detection handle Maximum Number to Configure
Return value	Error Code (Refer to Table 1)	
Description	The maximum number of faces to be detected by OKAO_Detection() can be set. OKAO_Detection() does not input face detection results that are more than the set values.	
Restriction	Values that can be set are integers of 1～140.	

`int OKAO_GetDtMaxFaceNumber(HDETECTION hDT, int *pnMax)`

Argument	Input：hDT Output：pnMax	Face detection handle The maximum number of face detections configured
Return value	Error Code (Refer to Table 1)	
Description	The maximum number of faces to be detected by OKAO_Detection() can be obtained.	

● Setting/Acquisition of Face detection Size mode

int OKAO_SetDtMinMaxFaceSizeMode(HDETECTION hDT, enum DETECT_FACE_SIZE_MODE mode)

Argument	Input：hDT　Face detection handle mode　Size mode of face detection
Return value	Error Code (Refer to Table 1)
Description	The mode for minimum and maximum face size by face detection can be set. ◎DETECT_FACE_RATIO: Percentage mode This is the mode configured by the percentage (%) of the minimum and maximum face-detection sizes to the length of the short side of the input image. The following settings/"get" function groups will be effective. OKAO_SetDtFaceSizeRatioRange(), OKAO_GetDtFaceSizeRatioRange() ◎DETECT_FACE_PIXEL: Pixel mode This is the mode by which the minimum and maximum face-detection sizes are configured by pixel values. The following settings/"get" function groups will be effective. OKAO_SetDtFaceSizeRange(), OKAO_GetDtFaceSizeRange() The minimum and maximum face-detection sizes that are already configured cannot be initialized even after changing the mode (the values will be preserved).
Restriction	If the setting for the minimum and maximum face-detection sizes/get function is called up other than the effective mode, an error will occur.

int OKAO_GetDtMinMaxFaceSizeMode(HDETECTION hDT, enum DETECT_FACE_SIZE_MODE *pMode)

Argument	Input：hDT　Face detection handle Output：pMode　Size mode of face detection
Return value	Error Code (Refer to Table 1)
Description	The mode for minimum and maximum face-detection sizes can be obtained.

● Setting/Acquisition of Face Size

`int OKAO_SetDtFaceSizeRange(HDETECTION hDT, int nMinSize, int nMaxSize)`

Argument	Input：hDT	Face detection handle
	nMinSize	Minimum face size（pixel）
	nMaxSize	Maxmum face size（pixel）
Return value	Error Code (Refer to Table 1)	
Description	The minimum and maximum face sizes to be detected can be configured by OKAO_Detection(). With nMinSize set at 100, a face with a size of 100 x 100 pixels or larger will be subject to detection. With nMaxSize set at 200, a face with a size of 200 x 200 pixels or smaller will be subject to detection. However, the configured size is merely a rule of thumb. If the minimum and maximum ranges are increased, the processing speed will be longer, but a large number of faces can be detected.	
Restriction	The setting value is the integer between 20 and 1024, which satisfies the condition of Min <= Max. ※If the setting is called up in a non-pixel mode, an error will occur. In that case, it must be set to the pixel mode by OKAO_SetDtMinMaxFaceSizeMode().	

`int OKAO_GetDtFaceSizeRange(HDETECTION hDT, int *pnMinSize, int *pnMaxSize)`

Argument	Input：hDT	Face detection handle
	Output：pnMinSize	Minimum face size (pixel)
	pnMaxSize	Maximum face size（pixel）
Return value	Error Code (Refer to Table 1)	
Description	The range of the face size to be detected can be obtained by OKAO_Detection().	
Restriction	If the setting is called up in a non-pixel mode, an error will occur. In that case, it must be set to the pixel mode by OKAO_SetDtMinMaxFaceSizeMode()	

int OKAO_SetDtFaceSizeRatioRange(HDETECTION hDT, int nMinSizeRatio, int nMaxSizeRatio)

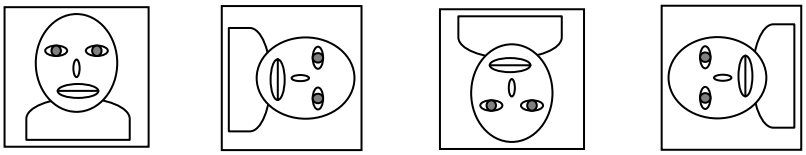
Argument	Input : hDT	Face detection handle
	nMinSizeRatio	Percentage of minimum face size (%)
	nMaxSizeRatio	Percentage of maximum face size (%)
Return value	Error Code (Refer to Table 1)	
Description	The range of the face size to be detected can be set by OKAO_Detection() using its percentage to the short-side length of the input image. If the short side is 500, with nMinSizeRatio set at 10, a face size of 50 x 50 pixels or larger will be subject to detection. With nMaxSizeRatio set at 100, a face size of 500 x 500 pixels or smaller will be subject to detection. However, the configured size is merely a rule of thumb. If the ranges of Min and Max values increase, the processing speed will be longer, but a large number of faces can be detected.	
Restriction	The setting value is the integer between 5 and 100, which satisfies the condition of Min <= Max. However, even within the above range, faces smaller than 20 pixels may not be detected. ※If the setting is called up in a non-percentage mode, an error will occur. In that case, it must be set to the percentage mode by OKAO_SetDtMinMaxFaceSizeMode().	

int OKAO_GetDtFaceSizeRatioRange(HDETECTION hDT, int *pnMinSizeRatio, int *pnMaxSizeRatio)

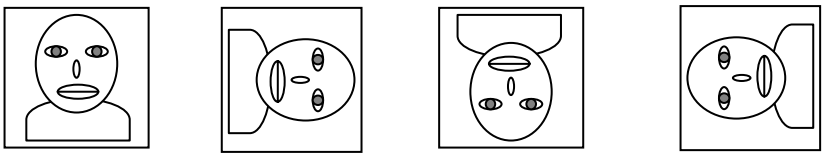
Argument	Input : hDT	Face detection handle
	Output : pnMinSizeRatio	Percentage of minimum face size (%)
	pnMaxSizeRatio	Percentage of maximum face size (%)
Return value	Error Code (Refer to Table 1)	
Description	The percentage (%) range of the face size to be detected by OKAO_Detection() can be obtained.	
Restriction	If the setting is called up in a pixel mode, an error will occur. In that case, it must be set to the pixel mode by OKAO_SetDtMinMaxFaceSizeMode()	

● Setting/Acquisition of search direction for Face detection

`int OKAO_SetDtDetectDirection(HDETECTION hDT, int nDirection)`

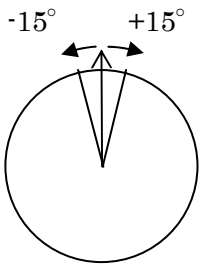
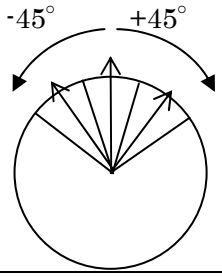
Argument	Input：hDT　Face detection handle nDirection　Search Direction
Return value	Error Code (Refer to Table 1)
Description	The rotation direction of the face image that should be detected can be specified. DETECT_UP　up direction image DETECT_RIGHT　right direction image DETECT_LEFT　left direction image DETECT_DOWN　down direction image  DETECT_UP　DETECT_RIGHT　DETECT_DOWN　DETECT_LEFT OKAO_Detection() performs face detection processing only in the direction configured. If specifying multiple directions, do it with OR（|）. For example, in the case of an upward- or right-angled image, it is necessary to specify DETECT_UP | DETECT_RIGHT.

`int OKAO_GetDtDetectDirection(HDETECTION hDT, int *pnDirection)`

Argument	Input：hDT　Face detection handle Output：pnDirection　Search Direction
Return value	Error Code (Refer to Table 1)
Description	The rotation direction of the face image to which the detection will be performed can be obtained under the existing setting.  DETECT_UP　DETECT_RIGHT　DETECT_DOWN　DETECT_LEFT

● Setting/Acquisition of the Angle Range of Face Detection

`int OKAO_SetDtDetectAngle(HDETECTION hDT, enum DETECT_ANGLE angle)`

Argument	Input：hDT angle	Face detection handle Angle Range
Return value	Error Code (Refer to Table 1)	
Description	The range of rotation to perform face detection can be set. Face detection is compliant with an angle of up to 15° in one direction. If search directions are increased, it is possible to detect more angled faces. But the processing time will be longer. DETECT_1ANGLE 1 direction（0°） DETECT_3ANGLE 3 direction（0°, 30°, -30°）	
	Angle Range DETECT_1ANGLE 	Angle Range DETECT_3ANGLE 

`int OKAO_GetDtDetectAngle(HDETECTION hDT, enum DETECT_ANGLE *pAngle)`

Argument	Input：hDT Output：pAngle	Face detection handle Range of face angle
Return value	Error Code (Refer to Table 1)	
Description	The range of rotation to perform face detection can be obtained under the current setting.	

● Setting/Acquisition of the Face Detection Search Steps

`int OKAO_SetDtStep(HDETECTION hDT, int nStep)`

Argument	Input : hDT nStep	Face detection handle Search step
Return value	Error Code (Refer to Table 1)	
Description	The number of search step for OKAO_Detection() can be set. If nStep is reduced, more detailed face detection can be performed, but the processing time will be longer. If nStep is increased, face detection can be performed at a higher speed.	
Restriction	The setting value is the integer between 10 and 40.	

`int OKAO_GetDtStep(HDETECTION hDT, int *pnStep)`

Argument	Input : hDT Output : pnStep	Face detection handle Search Steps
Return value	Error Code (Refer to Table 1)	
Description	The number of search steps for OKAO_Detection() can be obtained.	

● Setting/Acquisition of the Face Detection Color Mask

`int OKAO_SetDtColorMask(HDETECTION hDT, BOOL bFlag)`

Argument	Input : hDT bFlag	Face detection handle TRUE : valid , FALSE : invalid
Return value	Error Code (Refer to Table 1)	
Description	This is the dummy function for upper-compatible. No functional.	

`int OKAO_GetDtColorMask(HDETECTION hDT, BOOL *pbFlag)`

Argument	Input : hDT Output : pbFlag	Face detection handle TRUE : valid , FALSE : invalid
Return value	Error Code (Refer to Table 1)	
Description	This is the dummy function for upper-compatible. No functional.	

● Setting/Acquisition of the Face Detection Peripheral Mask

`int OKAO_SetDtRectangleMask(HDETECTION hDT, RECT rcArea)`

Argument	Input : hDT	Face detection handle
	rcArea	Rectangular Area of the Peripheral Mask to Configure
Return value	Error Code (Refer to Table 1)	
Description	The peripheral mask for OKAO_Detection () can be configured. The mask can be configured as an area to perform face detection in the rectangle specified by rcArea. However, the configured area is merely a rule of thumb. If –1 is configured for all elements of rcArea, the peripheral mask will be invalid. With the use of the peripheral mask, the detection rate of the face near the mask boundary may decline.	
Restriction	The setting range is –1 for all elements of rcArea or 0〜1023 for individual elements. Also, it is necessary to set the values so that top⇔bottom, left⇔right will not be reversed.	

`int OKAO_GetDtRectangleMask(HDETECTION hDT, RECT *prcArea)`

Argument	Input : hDT	Face detection handle
	Output : prcArea	Rectangular Area of the Configured Peripheral Mask
Return value	Error Code (Refer to Table 1)	
Description	The peripheral mask set by OKAO_Detection() can be obtained.	

● Setting/Acquisition of the Face Detection Threshold

`int OKAO_SetDtThreshold(HDETECTION hDT, int nThreshold)`

Argument	Input : hDT	Face detection handle
	nThreshold	Threshold
Return value	Error Code (Refer to Table 1)	
Description	The threshold for OKAO_Detection() can be configured. The face with a reliability less than this threshold is not detected.	
Restriction	The setting value is an integer between 0 and 1000.	

`int OKAO_GetDtThreshold(HDETECTION hDT, int *pnThreshold)`

Argument	Input : hDT	Face detection handle
	Output : pnThreshold	Threshold Value Configured
Return value	Error Code (Refer to Table 1)	
Description	The threshold for OKAO_Detection() can be obtained.	

● Setting/Acquisition of Face Detection Face Pose

`int OKAO_SetDtPose(HDETECTION hDT, int nPose)`

Argument	Input：hDT Face detection handle nPose Face pose
Return value	Error Code (Refer to Table 1)
Description	The face pose (pan) can be set to perform face detection. For the frontal view, faces with up to 30°right to left can be detected; for the diagonal view, faces with up to 60°right to left can be detected. If DETECT_HALF_PROFILE or DETECT_PROFILE is configured, a wider range of face angles can be detected, but the processing speed will decline. DETECT_FRONT Frontanl Face（±30°） DETECT_HALF_PROFILE Halfprofile Face（±60°）

`int OKAO_GetDtPose(HDETECTION hDT, int *pnPose)`

Argument	Input：hDT Face detection handle Output：pnPose Face pose Configured
Return value	Error Code (Refer to Table 1)
Description	The face pose(pan) can be obtained at the time of detection. DETECT_FRONT Frontal Face（±30°） DETECT_HALF_PROFILE Halfprofile Face（±60°）

3.2.3 Facial parts detection

● Creation/Deletion of Facial Parts Detection Handles

HPOINTER OKAO_CreatePointer(enum PT_MODE mode)

Argument	Input：mode Facial parts detection mode
Return value	！NULL : Facial parts detection handle ， NULL : false
Description	A handle can be created for the facial parts detection module. The setting modes for facial parts detection are as follows.. PT_MODE_DEFAULT（default mode） ※After using the handles, it is necessary to call up OKAO_DeletePointer() and delete them.

int OKAO_DeletePointer(HPOINTER hPT)

Argument	Input：hPT Facial parts detection handles to delete
Return value	Error Code (Refer to Table 1)
Description	The handles created by OKAO_CreatePointer() can be deleted.

HPTRESULT OKAO_CreatePtResult()

Argument	-
Return value	！NULL : Facial parts detection result handle ， NULL : false
Description	Handles can be created for the facial parts detection result storage. ※After using the handles, it is necessary to call up OKAO_DeletePtResult() and delete them.

int OKAO_DeletePtResult(HPTRESULT hResult)

Argument	Input：hResult Facial parts detection result handle to delete
Return value	Error Code (Refer to Table 1)
Description	The handles created by OKAO_CreatePtResult() must be deleted.

● Setting face position

int OKAO_SetPtPosition(HPOINTER hPT, HDTRESULT hResult, int nIndex)

Argument	Input：hPT Facial parts detection handle hResult Face detection result handle nIndex Face Index
Return value	Error Code (Refer to Table 1)
Description	The face position can be specified from the face detection result storage handle to the facial parts detection handle. nIndex enables specifying the index of face positions that are subject to adjustment.
Restriction	nIndex varies in the range of the integers between 0 and less than face detection［OKAO_GetDtFaceCount()-1］.

● Facial parts detection

```
int OKAO_Pointer(HPOINTER hPT, RAWIMAGE *pImage, int nWidth,
        int nHeight, int nDepth, HPTRESULT hResult, int *pnConfidence)
```

Argument	Input : hPT	Facial parts detection handle
	pImage	Input image
	nWidth	Image width
	nHeight	Image height
	nDepth	Image depth
	Output : hResult	Facial parts detection result handle
	pnConfidence	Confidence of facial parts detection (value from 0 to 1000)
Return value	Error Code (Refer to Table 1)	
Description	The specified positions can be detected from the face. This function does not cover detection of the pupil center. If you use gaze estimation, open-close estimation, or detection of the pupil center, you have to use OKAO_PointerAndGaze().	

```
int OKAO_PointerAndGaze(HPOINTER hPT, RAWIMAGE *pImage, int nWidth,
        int nHeight, int nDepth, HPTRESULT hResult, int *pnConfidence)
```

Argument	Input : hPT	Facial parts detection handle
	pImage	Input image
	nWidth	Image width
	nHeight	Image height
	nDepth	Image depth
	Output : hResult	Facial parts detection result handle
	pnConfidence	Confidence of facial parts detection (value from 0 to 1000)
Return value	Error Code (Refer to Table 1)	
Description	The specified positions and gaze can be detected from the face. If you use gaze estimation, open-close estimation, or detection of the pupil center, you have to use this function.	

● Points in Facial Parts Detection

`int OKAO_GetPtPointNumber(HPTRESULT hResult, int *pnCount)`

Argument	Input : hResult Output : pnCount	Facial parts detection result handle Number of detected facial feature points
Return value	Error Code (Refer to Table 1)	
Description	The number of facial feature basic points can be obtained by OKAO_Pointer() or OKAO_PointerAndGaze().	
Restriction	Call after OKAO_Pointer() or OKAO_PointerAndGaze().	

`int OKAO_GetPtPoint(HPTRESULT hResult, POINT aptPoint[], int aConf[])`

Argument	Input : hResult Output : aptPoint aConf	Facial parts detection result handle Coordinates of Detection Points (Note 2) Confidence of facial parts detection（value from 0 to 1000）
Return value	Error Code (Refer to Table 1)	
Description	The detection-point coordinate found by OKAO_Pointer() or OKAO_PointerAndGaze() can be obtained. When the returned coordinate is FEATURE_NO_POINT, it shows that the point could not be detected. For aptPoint and aConf, it is necessary to secure the buffer for OKAO_GetPtPointNumber() before calling it up.	
Restriction	Call after OKAO_Pointer() or OKAO_PointerAndGaze().	

`int OKAO_GetPtDetailPointNumber(HPTRESULT hResult, int *pnCount)`

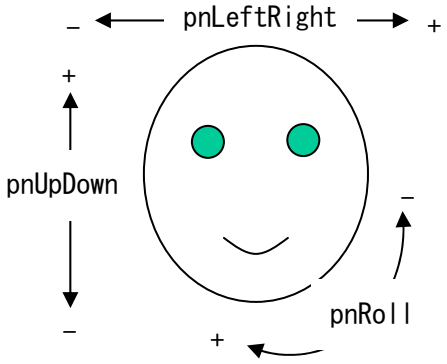
Argument	Input : hResult Output : pnCount	Facial parts detection result handle Number of detected facial feature points
Return value	Error Code (Refer to Table 1)	
Description	The number of facial feature precise points can be obtained by OKAO_PointerAndGaze().	
Restriction	Call after OKAO_PointerAndGaze().	

`int OKAO_GetPtDetailPoint(HPTRESULT hResult, POINT aptFeatureDetail[], int aConfDetail[])`

Argument	Input : hResult Output : aptFeatureDetail aConfDetail	Facial parts detection result handle Coordinates of Detection Points (Note 4) Confidence of facial parts detection（value from 0 to1000）
Return value	Error Code (Refer to Table 1)	
Description	The detection-point coordinate found by OKAO_PointerAndGaze() can be obtained. When the returned coordinate is FEATURE_NO_POINT, it shows that the point could not be detected. For aptFeatureDetail and aConfDetail, it is necessary to secure the buffer for OKAO_GetPtDetailPointNumber() before calling it up.	
Restriction	This function does not output confidence at this version(Reserved). Call after OKAO_PointerAndGaze().	

● Acquisition of Estimated Result of Face Direction

```
int OKAO_GetPtDirection(HPTRESULT hResult, int *pnUpDown, int *pnLeftRight,
                        int *pnRoll, int *pnConfidence)
```

Argument	Input : hResult Output : pnUpDown pnLeftRight pnRoll pnConfidence	Facial parts detection result handle The angle of vertical direction of face (degree) The angle of horizontal direction of face (degree) The angle of rotation direction of face (degree) Confidence of facial parts detection（value from 0 to 1000）
Return value	Error Code (Refer to Table 1)	
Description	The angle (degree) of face direction detected by OKAO_Pointer() can be obtained. When the value of “pnUpDown” is positive, it means that the direction of face is upward. When the value of ” pnLeftRight” is positive, it means that the direction of face is rightward from the observer’s viewpoint. (This is from person whose face is printed.) When the value of “pnRoll” is positive, it means that the rotation direction of face is clockwise from the observer’s viewpoint. (This is from observer[image file].) (This is based on vertical/horizontal direction.) The value of “pnConfidence” is same as the output of OKAO_Pointer().  	

● Gaze Estimation Result

```
int OKAO_GetPtGazePoint (HPTRESULT hResult, int *pnGazeLR, int *pnGazeUD,
                         int *pnConfidenceGaze)
```

Argument	Input : hResult Output : pnGazeLR pnGazeUD pnConfidenceGaze	Facial parts detection result handle The angle of horizontal gaze (degree) The angle of vertical gaze (degree) Confidence of gaze estimation (value from 0 to 1000)
Return value	Error Code (Refer to Table 1)	
Description	The angle (degree) of gaze detected by OKAO_PointerAndGaze() can be obtained. When the value of "pnGazeUD" is positive, it means that the gaze is upward. When the value of " pnGazeLR" is positive, it means that the gaze is rightward from the observer's viewpoint. The angle of gaze is including face direction. So if the person sees straight, but face direction is 30 degrees, output is 30. (This is from observer[image file], not from person whose face is printed.) The confidence will be low if eye is closing or observer can't see eye perfectly because of face direction.	

● Open-Close of the Facial Parts Estimation Result

```
int OKAO_GetPtOpenLevel (HPTRESULT hResult, int *pnLeftEyeOpenLevel,
        int *pnRightEyeOpenLevel, int *pnMouthOpenLevel,
        int *pnLeftEyeOpenLevelConfidence, int *pnRightEyeOpenLevelConfidence,
        int *pnMouthOpenLevelConfidence)
```

Argument	Input : hResult Output : pnLeftEyeOpenLevel pnRightEyeOpenLevel pnMouthOpenLevel pnLeftEyeOpenLevelConfidence pnRightEyeOpenLevelConfidence pnMouthOpenLevelConfidence	Facial parts detection result handle The level of left eye open (value from 0 to 1000) The level of right eye open (value from 0 to 1000) The level of mouth open (value from 0 to 1000) The confidence of left eye open level (value from 0 to 1000) The confidence of right eye open level (value from 0 to 1000) The confidence of mouth open level (value from 0 to 1000)
Return value	Error Code (Refer to Table 1)	
Description	Open-close estimation results detected by OKAO_PointerAndGaze() can be obtained. Level will be shown as integer between 0 to 1000. The bigger the level is, more nearly open the facial feature is. The confidence will be low if observer can't see eye perfectly because of face direction.	

3.2.4 Face recognition

We assume these definitions.

```
typedef    FR_USER_ID    BYTE;          /* Personal identification ID */
#define    USER_ID_LENGTH    16         /* Length of personal identification ID */
```

* Personal identification ID is 16 bytes data (not strings). '¥0' is not end. We will check equals or not with memcmp().

● Setting/Acquisition of Face-Recognition Data Handles

`HFACERECOG OKAO_CreateFaceRecognition(enum FR_MODE mode)`

Argument	Input : mode　　Face recognition mode
Return value	！NULL : Face recognition data handle ，　NULL : false
Description	Face-recognition data handles can be created. The setting modes for creating face-recognition data are as follows. FR_MODE_DEFAULT　（DEFAULT） ※After using the handles, it is necessary to call up OKAO_DeleteFaceRecognition() and delete them.

`int OKAO_DeleteFaceRecognition(HFACERECOG hFR)`

Argument	Input : hFR　　Delete face recognition data handle
Return value	Error Code (Refer to Table 1)
Description	This will delete the handles created by OKAO_CreateFaceRecognition ().

`HFRRESULT OKAO_CreateIdentifyResult(unsigned int nMax)`

Argument	Input : nMax　　Maximum number of identification size
Return value	！NULL : Face identification result handle ，　NULL : false
Description	Face identification result handles can be created. nMax is the maximum number as results to identify a face

`int OKAO_DeleteIdentifyResult(HFRRESULT hResult)`

Argument	Input : hReslut　　Delete face recognition result handle
Return value	Error Code (Refer to Table 1)
Description	This will delete the handles created by OKAO_CreateIdentifyResult ().

● Creation/Deletion of Database handle

```
HALBUMDB OKAO_CreateDatabase(enum DB_MODE mode, unsigned int nUser, unsigned int nPicture)
```

Argument	Input：mode　Mode of registration database nUser　Maximum number of personal identification ID nPicture　Maximum number of pictures of each personal identification ID
Return value	！NULL : Database handle ，　NULL : false
Description	Handles for the face-recognition registration database can be created. The mode for the face-recognition registration database is set as follows. DB_MODE_DEFAULT　（DEFAULT） After using the handles, it is necessary to call up OKAO_DeleteDatabase() and delete them. nUser is the maximum number of personal identification ID. nPicture is the maximum number of pictures of each personal identification ID. You cannot register over these values with database.
Restriction	One cannot **simultaneously** call up the album operation function by multi -threads (Note 1). nUser is the integer between 1 and 10000. nPicture is the integer between 1 and 50. In this version of the database, one can only use one system. Therefore, to create more of a handle can not be used

```
int OKAO_DeleteDatabase(HALBUMDB hDB)
```

Argument	Input：hDB　Database handle to delete
Return value	Error Code (Refer to Table 1)
Description	The handles created by OKAO_CreateDatabase() can be deleted and memory free will be run.
Restriction	One cannot **simultaneously** call up the album operation function by multi-threads (Note 1).

● Setting of Face recognition data

```
int OKAO_SetFrDataFromPtResult(HFACERECOG hFR, RAWIMAGE *pImage, int nWidth, int nHeight,
        int nDepth, HPTRESULT hResult)
```

Argument	I/O：hFR　Face recognition data handle Input：pImage　Input image nWidth　width nHeight　height nDepth　depth hResult　facial parts detection result handle
Return value	Error Code (Refer to Table 1)
Description	Face-recognition data can be configured from the facial parts detection result storage handle and image information. In this version, it is necessary to configure the following coordinates of facial-feature position. FEATURE_MOUTH　/* center of moth*/ FEATURE_LEFT_EYE_IN　/* inner corner of left eye */ FEATURE_LEFT_EYE_OUT　/* outer corner of left eye */ FEATURE_RIGHT_EYE_IN　/* inner corner of right eye */ FEATURE_RIGHT_EYE_OUT　/* outer corner right eye */ FEATURE_MOUTH_LEFT　/* mouth left corner */ FEATURE_MOUTH_RIGHT　/* mouth right corner */
Restriction	

● Identification

```
int OKAO_FrIdentify(HFACERECOG hFRGallary, RAWIMAGE *pImage, int nWidth, int nHeight,
        int nDepth, HALBUMDB hDB, int nThreshold, HFRRESULT hResult)
```

Argument	Input：hFRGallary	face recognition data handles
	pImage	Input image
	nWidth	width
	nHeight	height
	nDepth	depth
	hDB	Database handle
	nThreshold	Threshold（Integers 0 - 1000）
	Output：hResult	face identification result handle
Return value	Error Code (Refer to Table 1)	
Description	It is possible to identify a face by comparing its data to the database data.	
Restriction	The interval between the eyes and that between the eye and the mouth need to be 10 pixels or greater. If one uses out-of-spec face images, OKAO_ERR_INVALIDPARAM will return. ※Face-recognition data handles that have been obtained in different version modes may not be identified.	

```
int OKAO_GetFrResult(HFRRESULT hReslut, FR_USER_ID *pUid, int anConf[], int *pnCount)
```

Argument	Input：hResult	face identification result handle
	Output：pUid	Result personal identification ID
	anConf	Reliability list（Integers 0 - 1000）
	pnCount	Number of outputs
Return value	Error Code (Refer to Table 1)	
Description	The result by OKAO_FrIdentify() can be obtained. As for pUid, it is necessary to secure the buffer for pUid[nMax of OKAO_CreateIdentifyResult() * USER_ID_LENGTH] before calling them up. As for anConf, it is necessary to secure the buffer for nMax of OKAO_CreateIdentifyResult() before calling them up. Output is sorted by reliability, as pUid[0]～pUid[USER_ID_LENGTH – 1], anConf[0] = 900 pUid[USER_ID_LENGTH]～pUid[USER_ID_LENGTH * 2 – 1], anConf[1] = 200….	

● Face identification result integration

```
int OKAO_FrIntegrate(HFRRESULT hInResult[], int nResultNum,
        HFRRESULT hOutResult, int *pnOutIndex)
```

Argument	Input : hInResult nResultNum Output : hOutResult pnOutIndex	Face identification result handle Input result handle number Face identification result handle Index of hInResult that is almost hOutResult
Return value	Error Code (Refer to Table 1)	
Description	Integrates the nResultNum of face identification result handle as found in hInResult and output to hOutResult. This function is expected for use in performing multiple face identification on same person and returning the integrated result. Returns the handle number (between 1~nResultNum) closest to the integrated result to pnOutIndex.	
Restriction	nResultNum is the integer between 0 and 100.	

● Database control

```
int OKAO_RegistDatabase(HALBUMDB hDB, HFACERECOG hFR, RAWIMAGE *pImage, int nWidth,
        int nHeight, int nDepth, FR_USER_ID aUid[], int nVectorNo)
```

Argument	I/O : hDB	Database handle
	Input : hFR	Registration of Face recognition data handle
	pImage	Input image
	nWidth	width
	nHeight	height
	nDepth	depth
	aUid	Personal identification ID
	nVectorNo	Picture number
Return value	Error Code (Refer to Table 1)	
Description	It is possible to additionally register face-recognition data to the database. If specified personal identification ID is existed in the database, this causes addition of existing data, otherwise new user is created in the database. As for aUid, it is necessary to secure the buffer for USER_ID_LENGTH before calling them up, and set all bytes. '¥0' is not the end and we will check equality with memcmp(). nVectorNo is the integer between 1 and nPicture of OKAO_CreateDatabase() or *pnVectorNo of OKAO_SetDatabaseData().	
Restriction	Only up to nPicture of OKAO_CreateDatabase() or *pnVectorNo of OKAO_SetDatabaseData() can be registered. When trying to register more data, OKAO_ERR_INVALIDPARAM will return. Face-recognition data handles that have been obtained in different version modes may not be registered.	

```
int OKAO_DeleteDBData(HALBUMDB hDB, FR_USER_ID aUid[], int nVectorNo)
```

Argument	In/Out : hDB	Database handle
	Input : aUid	Personal identification ID to delete
	nVectorNo	Picture number to delete
Return value	Error Code (Refer to Table 1)	
Description	From the database, It is possible to delete face-recognition data that can be specified by aUid and nVectorNo.	

● Setting/Acquisition of Database data

`int OKAO_GetDatabaseSize(HALBUMDB hDB, int *plSize)`

Argument	Input：hDB Output：plSize	Database handle Size of database data
Return value	Error Code (Refer to Table 1)	
Description	It is possible to obtain the database data size obtainable by OKAO_GetDatabaseData(). If database has no data, zero will be returned.	

`int OKAO_GetDatabaseData(HALBUMDB hDB, BYTE *pbyBuffer)`

Argument	Input：hDB Output：pbyBuffer	Database handle Database buffer
Return value	Error Code (Refer to Table 1)	
Description	Data necessary to save database data can be obtained. It is necessary to secure an database data buffer for OKAO_GetDatabaseSize() before calling it up.	

`int OKAO_SetDatabaseData(HALBUMDB hDB, BYTE *pbyBuffer, int *pnUserNum, int *pnVectorNo)`

Argument	In/Out：hDB Input：pbyBuffer Output：pnUserNum pnVectorNo	Database data handle Binary data of database Maximum number of personal identification ID Maximum number of pictures of each personal identification ID
Return value	Error Code (Refer to Table 1)	
Description	It is possible to configure the binary data to be obtained by OKAO_GetDatabaseData() into the album data. ※It may not be possible to configure binary data that have been obtained in different versions.	

3.2.5 Gender estimation

● Creation/Deletion of Gender-Estimation Handles

HGENDER OKAO_CreateGender(enum GE_MODE mode)

Argument	Input：mode Gender estimation mode
Return value	！NULL : Gender estimation handle ， NULL : false
Description	The handle for the gender-estimation module can be created. The setting modes for gender-estimation are as follows. GE_MODE_DEFAULT （DEFAULT） ※After using the handles, it is necessary to call up OKAO_DeleteGender() and delete them.

int OKAO_DeleteGender(HGENDER hGE)

Argument	Input：hGE Gender estimation handle to delete
Return value	Error Code (Refer to Table 1)
Description	The handles created by OKAO_CreateGender() can be deleted.

HGENRESULT OKAO_CreateGenderResult(void)

Argument	-
Return value	！NULL : Gender estimation result handle ， NULL : false
Description	Handles can be created for the gender estimation result storage. ※After using the handles, it is necessary to call up OKAO_DeleteGenderResult() and delete them.

int OKAO_DeleteGenderResult(HGENRESULT hResult)

Argument	Input：hResult Gender estimation result handle to delete
Return value	Error Code (Refer to Table 1)
Description	The handles created by OKAO_CreateGenderResult() must be deleted.

● Setting of position of facial features

int OKAO_SetGePoint(HGENDER hGE, HPTRESULT hResult)

Argument	Input：hGE Gender estimation handle hResult Facial parts detection result handle
Return value	Error Code (Refer to Table 1)
Description	From the handle of facial parts detection result storage, it is possible to configure the facial-feature positions on the gender-estimation handle.

● Gender estimation

```
int OKAO_GenderEstimate(HGENDER hGE, RAWIMAGE *pImage,
        int nWidth, int nHeight, int nDepth, HGENRESULT hResult)
```

Argument	Input：hGE pImage nWidth nHeight nDepth Output：hResult	Gender estimation handle Input image width height depth Gender estimation result handle
Return value	Error Code (Refer to Table 1)	
Description	Gender estimation can be performed and the result stored in hResult.	
Restriction	If the intervals between the eyes and between the eye and mouth is smaller than 20 pixels, accuracy will decrease. At this version, make right decision with Japanese.	

```
int OKAO_GetGenderResult(HGENRESULT hResult, BOOL *pbMale, int *pnConfidence)
```

Argument	Input：hResult Output：pbMale pnConfidence	Gender estimation result handle TRUE:male、FALSE:female confidence（value from 0 to 1000）
Return value	Error Code (Refer to Table 1)	
Description	Gender estimation results detected by OKAO_GenderEstimate() can be obtained. The confidence will be nearly 1000 if observer can estimate gender perfectly.	

● Gender estimation result integration

```
int OKAO_GenderIntegrate(HGENRESULT hInResult[], int nResultNum,
        HGENRESULT hOutResult, int *pnOutIndex)
```

Argument	Input：hInResult nResultNum Output：hOutResult pnOutIndex	Gender estimation result handle Input result handle number Gender estimation result handle Index of hInResult that is almost hOutResult
Return value	Error Code (Refer to Table 1)	
Description	Integrates the nResultNum of age estimation result handle as found in hInResult and output to hOutResult. This function is expected for use in performing multiple age estimation on same person and returning the integrated result. Returns the handle number (between 1~nResultNum) closest to the integrated result to pnOutIndex.	
Restriction	nResultNum is the integer between 0 and 100.	

3.2.6 Age group estimation

● Creation/Deletion of Age-Estimation Handles

HAGE OKAO_CreateAge(enum AGE_MODE mode)

Argument	Input：mode Age group estimation mode
Return value	！NULL : Age group estimation handle ， NULL : false
Description	The handle for the age group estimation module can be created. The setting modes for age group estimation are as follows. AGE_MODE_DEFAULT （DEFAULT） ※After using the handles, it is necessary to call up OKAO_DeleteAge() and delete them.

int OKAO_DeleteAge(HAGE hAGE)

Argument	Input：hAGE Age group estimation handle to delete
Return value	Error Code (Refer to Table 1)
Description	The handles created by OKAO_CreateAge() can be deleted.

HAGERESULT OKAO_CreateAgeResult(void)

Argument	-
Return value	！NULL : Age group estimation result handle ， NULL : false
Description	Handles can be created for the age group estimation result storage. ※After using the handles, it is necessary to call up OKAO_DeleteAgeResult() and delete them.

int OKAO_DeleteAgeResult(HAGERESULT hResult)

Argument	Input：hResult Age group estimation result handle to delete
Return value	Error Code (Refer to Table 1)
Description	The handles created by OKAO_CreateAgeResult() must be deleted.

● Setting of position of facial features

int OKAO_SetAgePoint(HAGE hAGE, HPTRESULT hResult)

Argument	Input：hAGE Age group estimation handle hResult Facial parts detection result handle
Return value	Error Code (Refer to Table 1)
Description	From the storage for the handle of facial parts detection result, it is possible to configure the facial-feature positions on the age-estimation handle.
Restriction	If the intervals between the eyes and between the eye and mouth is smaller than 30 pixels, accuracy will decrease. If out-of-spec face images are used, OKAO_ERR_INVALIDPARAM will return.

● Setting of Age group division

```
int OKAO_SetAgeDivision(HAGE hAge, int anDivision[], int nCount)
```

Argument	Input：hAGE anDivision nCount	Age group estimation handle Division of the age Division number
Return value	Error Code (Refer to Table 1)	
Description	Set division of the age. nCount to the break and put the number of age, the number of anDivision [] to set the age. In this setting OKAO_AgeEstimate () to estimate the output of the lower age limit is determined. Example) nCount = 5, anDivision [] = (10, 20, 30, 40, 50), and if you set the minimum age limit of the combination of values from 0 to take a 9.10 ~ 19.20 to 29 ~ 30 to 39.40 and 49.50 to 999 as of 6. ※ 999 that indicates that there is no maximum number. Was below the limit specified in the case against the returns an error. When the value is not changed	
Restriction	nCount is the integer between 0 and 10. Divided you can specify a minimum age of 5, maximum of 70 must be in ascending order. The division of the age of five or more width is required. ※ recommended a 10-year-old increments. If more detailed decreased performance.	

● Age group estimation

```
int OKAO_AgeEstimate(HAGE hAGE, RAWIMAGE *pImage, int nWidth, int nHeight,
                     int nDepth, HAGERESULT hResult)
```

Argument	Input : hAGE	Age group estimation handle
	pImage	Input image
	nWidth	width
	nHeight	height
	nDepth	depth
	Output : hResult	Age group estimation result handle
Return value	Error Code (Refer to Table 1)	
Description	Age group estimation can be performed.	
Restriction	If the intervals between the eyes and between the eye and mouth is smaller than 30 pixels, accuracy will decrease. At this version, make right decision with Japanese.	

```
int OKAO_GetAgeResult(HAGERESULT hResult, int *pnLower, int *pnUpper, int *pnConfidence)
```

Argument	Input : hResult	Gender estimation result handle
	Output : pnLower	lower limit of estimate age
	pnUpper	higher limit of estimate age
	pnConfidence	confidence （value from 0 to 1000）
Return value	Error Code (Refer to Table 1)	
Description	Age group estimation results detected by OKAO_AgeEstimate() can be obtained.	

● Age group estimation result integration

```
int OKAO_AgeIntegrate(HAGERESULT hInResult[], int nResultNum,
        HAGERESULT hOutResult, int *pnOutIndex)
```

Argument	Input : hInResult	Age group estimation result handle
	nResultNum	Input result handle number
	Output : hOutResult	Age group estimation result handle
	pnOutIndex	Index of hInResult that is almost hOutResult
Return value	Error Code (Refer to Table 1)	
Description	Integrates the nResultNum of age group estimation result handle as found in hInResult and output to hOutResult. This function is expected for use in performing multiple age group estimation on same person and returning the integrated result. Returns the handle number (between 1~nResultNum) closest to the integrated result to pnOutIndex.	
Restriction	nResultNum is the integer between 0 and 100.	

3.2.7 Age estimation

● Creation/Deletion of Age-Estimation Handles

HAGE OKAO_CreateYear(enum YE_MODE mode)

Argument	Input：mode Age estimation mode
Return value	！NULL : Age estimation handle ， NULL : false
Description	The handle for the age-estimation module can be created. The setting modes for age-estimation are as follows. YE_MODE_DEFAULT （DEFAULT） ※After using the handles, it is necessary to call up OKAO_DeleteYear() and delete them.

int OKAO_DeleteYear(HYEAR hYEAR)

Argument	Input：hYEAR Age estimation handle to delete
Return value	Error Code (Refer to Table 1)
Description	The handles created by OKAO_CreateYear() can be deleted.

HYEARRESULT OKAO_CreateYearResult(void)

Argument	-
Return value	！NULL : Age estimation result handle ， NULL : false
Description	Handles can be created for the age estimation result storage. ※After using the handles, it is necessary to call up OKAO_DeleteYearResult() and delete them.

int OKAO_DeleteYearResult(HYEARRESULT hResult)

Argument	Input：hResult Age estimation result handle to delete
Return value	Error Code (Refer to Table 1)
Description	The handles created by OKAO_CreateYearResult() must be deleted.

● Setting of position of facial features

int OKAO_SetYearPoint(HYEAR hYEAR, HPTRESULT hResult)

Argument	Input：hYEAR Age estimation handle hResult Facial parts detection result handle
Return value	Error Code (Refer to Table 1)
Description	From the storage for the handle of facial parts detection result, it is possible to configure the facial-feature positions on the age-estimation handle.

● Age estimation

```
int OKAO_YearEstimate(HYEAR hYEAR, RAWIMAGE *pImage, int nWidth, int nHeight,
                      int nDepth, HYAERRESULT hResult)
```

Argument	Input：hYEAR	Age estimation handle
	pImage	Input image
	nWidth	width
	nHeight	height
	nDepth	depth
	Output：hResult	Age estimation result handle
Return value	Error Code (Refer to Table 1)	
Description	Age estimation can be performed.	
Restriction	If the intervals between the eyes and between the eye and mouth is smaller than 30 pixels, accuracy will decrease. At this version, make right decision with Japanese.	

```
int OKAO_GetYearResult(HYEARRESULT hResult, int *pnYear, int *pnConfidence)
```

Argument	Input：hResult	Gender estimation result handle
	Output：pnYear	year
	pnConfidence	confidence（value from 0 to 1000）
Return value	Error Code (Refer to Table 1)	
Description	Age estimation results detected by OKAO_YearEstimate() can be obtained. The confidence will be nearly 1000 if observer can estimate year perfectly.	

● Age estimation result integration

```
int OKAO_YearIntegrate(HYEARRESULT hInResult[], int nResultNum,
        HYEARRESULT hOutResult, int *pnOutIndex)
```

Argument	Input : hInResult	Age estimation result handle
	nResultNum	Input result handle number
	Output : hOutResult	Age estimation result handle
	pnOutIndex	Index of hInResult that is almost hOutResult
Return value	Error Code (Refer to Table 1)	
Description	Integrates the nResultNum of age estimation result handle as found in hInResult and output to hOutResult. This function is expected for use in performing multiple age estimation on same person and returning the integrated result. Returns the handle number (between 1~nResultNum) closest to the integrated result to pnOutIndex.	
Restriction	nResultNum is the integer between 0 and 100.	

● Getting of tolerance level

```
int OKAO_YearToleranceLevel(HYEAR hYEAR, int nYear, int nConfidence,
        int *pnUpper, int *pnLower)
```

Argument	Input : hYEAR	Age estimation handle
	nYear	year
	nConfidence	confidence
	Output : pnUpper	Upper limit value of tolerance level
	pnLower	Lower limit value of tolerance level
Return value	Error Code (Refer to Table 1)	
Description	Returns the upper and lower limit values of age tolerance level obtained from the age and confidence pairs outputed from OKAO_GetYearResult().	
Restriction	nYear is the integer between 2 and 77. nConfidence is the integer between 0 and 1000.	

3.2.8 Smile degree estimation

● Creation/Deletion of Smile Degree Estimation Handles

```
HSMILE OKAO_CreateSmile(enum SMILE_MODE mode)
```

Argument	Input：mode　　Smile degree estimation mode
Return value	！NULL : Smile degree estimation handle ，　NULL : false
Description	The handle for the Smile degree estimation module can be created. The setting modes for Smile degree estimation are as follows. SMILE_MODE_DEFAULT　（DEFAULT） ※After using the handles, it is necessary to call up OKAO_DeleteSmile() and delete them.

```
int OKAO_DeleteSmile(HSMILE hSMILE)
```

Argument	Input：hSMILE　　Smile degree estimation handle to delete
Return value	Error Code (Refer to Table 1)
Description	The handles created by OKAO_CreateSmile() can be deleted.

● Setting of position of facial features

```
int OKAO_SetSmilePoint(HSMILE hSMILE, HPTRESULT hResult)
```

Argument	Input：hSMILE　　Smile degree estimation handle hResult　　Facial parts detection result handle
Return value	Error Code (Refer to Table 1)
Description	From the storage for the handle of facial parts detection result, it is possible to configure the facial-feature positions on the Smile degree estimation handle.
Restriction	If the intervals between the eyes and between the eye and mouth is smaller than 30 pixels, accuracy will decrease.

● Smile degree estimation

```
int OKAO_SmileCheck(HSMILE hSMILE, RAWIMAGE *pImage, int nWidth, int nHeight,
        int *pnSmile)
```

Argument	Input：hSMILE　　Smile degree estimation handle pImage　　Input image nWidth　　width nHeight　　height nDepth　　depth Output：pnSmile　　smile rate（value from 0 to 100）
Return value	Error Code (Refer to Table 1)
Description	Smile rate can be measured. Rate is the integer between 0 and 100. 100 shows fully smile, 0 shows not smiling.
Restriction	If the intervals between the eyes and between the eye and mouth is smaller than 30 pixels, accuracy will decrease.

3.2.9 Facial feature outline detection

● Creation/Deletion of Handles for Facial-Feature Outline Detection

HCONTOUR OKAO_CreateContour(enum CT_MODE mode)

Argument	Input：mode Facial feature outline detection mode
Return value	！NULL : Facial feature outline detection handle ， NULL : false
Description	It is possible to create the handle for the facial-feature outline detection module. The setting modes for facial-feature contour detection are as follows. CT_MODE_DEFAULT（normal mode） ※After using the handles, it is necessary to call up OKAO_DeleteContour() and delete them.

int OKAO_DeleteContour(HCONTOUR hCT)

Argument	Input：hCT Facial feature outline detection handle to delete
Return value	Error Code (Refer to Table 1)
Description	The handles created by OKAO_CreateContour() can be deleted.

HCTRESULT OKAO_CreateCtResult()

Argument	-
Return value	！NULL : Facial feature outline detection result handle ， NULL : false
Description	Handles for the facial-feature outline detection result storage can be created. ※After using the handles, it is necessary to call up OKAO_DeleteCtResut() and delete them.

int OKAO_DeleteCtResult(HCTRESULT hResult)

Argument	Input：hResult Facial feature outline detection result handle to delete
Return value	Error Code (Refer to Table 1)
Description	Handles created by OKAO_CreateCtResult() can be deleted.

● Setting of position of facial features

int OKAO_SetCtPosition(HCONTOUR hCT, HPTRESULT hResult)

Argument	Input：hCT Facial feature outline detection handle hResult Facial parts detection result handle
Return value	Error Code (Refer to Table 1)
Description	From the storage for the handle of the facial parts detection result, it is possible to configure the facial-feature position on the facial-feature outline detection handle. In this version, it is necessary to configure the following facial-feature position coordinates. FEATURE_LEFT_EYE_IN /*inner corner of left eye */ FEATURE_LEFT_EYE_OUT /* outer corner of left eye */ FEATURE_RIGHT_EYE_IN /* inner corner of right eye */ FEATURE_RIGHT_EYE_OUT /* outer corner right eye */ FEATURE_MOUTH_LEFT /* mouth left corner*/ FEATURE_MOUTH_RIGHT /* mouth right corner */
Restriction	The interval between the eyes and that between the eyes and the mouth need to be 60 pixels or greater; the y-coordinates of the eyes and the mouth need to be different. OKAO_ERR_INVALIDPARAM will return if one uses out-of-spec facial images.

● Facial Feature Outline Detection

```
int OKAO_Contour(HCONTOUR hCT, RAWIMAGE *pImage, int nWidth, int nHeight, int nDepth,
        HCTRESULT hResult)
```

Argument	Input : hCT	Facial feature outline detection handle
	pImage	Input image
	nWidth	width
	nHeight	height
	nDepth	depth
	Output : hResult	Facial feature outline detection result handle
Return value	Error Code (Refer to Table 1)	
Description	The feature contour position can be extracted from the face.	

● Number of Points for Facial-Feature Outline Detection

```
int OKAO_GetCtPointNumber(HCTRESULT hResult, int *pnCount)
```

Argument	Input : hResult	Facial feature outline detection result handle
	Output : pnCount	Number of Points Extracted
Return value	Error Code (Refer to Table 1)	
Description	The number of points to be extracted by OKAO_Contour() can be obtained.	

```
int OKAO_GetCtPoint(HCTRESULT hResult, POINT aptContour[])
```

Argument	Input : hResult	Facial feature outline detection result handle
	Output : actPoint	Coordinates of Facial feature outline detection （Note 3）
Return value	Error Code (Refer to Table 1)	
Description	One obtains coordinates for facial-feature outline detection points, which were found by OKAO_Contour(). This shows that when the returned coordinate was CONTOUR_NO_POINT, they were not detected. As for aptContour, it is necessary to secure the buffer for OKAO_GetCtPointNumber() before calling it up.	

3.3 Error Code

Table 1 shows the errors that may occur during the use of The Library.

Table 1 Error Code Table

Error Code	Error Code	Error Content
OKAO_NORMAL	0	Normal End
OKAO_TIMEOUT	1	Face Detection Processing Timeout
OKAO_NOTIMPLEMENTED	2	Not Implemented now
OKAO_ERR_VARIOUS	-1	Undefined Error
OKAO_ERR_INITIALIZE	-2	Initialization Error
OKAO_ERR_INVALIDPARAM	-3	Argument Error
OKAO_ERR_ALLOCMEMORY	-4	Memory Allocation Error
OKAO_ERR_MODEDIFFERENT	-5	Mode Different Error
OKAO_ERR_NOALLOC	-6	(reserved)
OKAO_ERR_NOHANDLE	-7	Handle Error
OKAO_ERR_DUPLICATE	-8	Duplicate Error

3.4 Notes

（Note 1）
The target album operations are OKAO_CreateDatabase(), OKAO_DeleteDatabase().

（Note 2）
The POINT aptPoint[] and BOOL abPoint[] arrays in this version mean:

```
enum FEATURE_POINT {                /* index of Facial parts detection */
    FEATURE_LEFT_EYE = 0,                   /* center of left eye */
    FEATURE_RIGHT_EYE,              /* center of right eye*/
    FEATURE_MOUTH,                  /* center of moth*/
    FEATURE_LEFT_EYE_IN,                    /* inner corner of left eye */
    FEATURE_LEFT_EYE_OUT,           /* outer corner of left eye */
    FEATURE_RIGHT_EYE_IN,           /* inner corner of right eye */
    FEATURE_RIGHT_EYE_OUT,          /* outer corner of right eye */
    FEATURE_MOUTH_LEFT,                     /* left corner of mouth */
    FEATURE_MOUTH_RIGHT,                    /* right corner of mouth */
    FEATURE_LEFT_EYE_PUPIL,         /* center of left eye pupil */
    FEATURE_RIGHT_EYE_PUPIL,        /* center of right eye pupil */
    FEATURE_KIND_MAX                /* numner of features */
};
```

… and have 11 points. But the size can be changed in the future.

* Center of both eye pupil is not set if you don't use OKAO_PointerAndGaze().

（Note 3）
POINT aptContour[] array in this version means:

```
enum CONTOUR_POINT {                /* index of Facial feature outline detection */
CONTOUR_EYEL_1 = 0,                 /* left eye first */
CONTOUR_EYEL_2,                     /* left eye second */
CONTOUR_EYEL_3,                     /* left eye third */
CONTOUR_EYEL_4,                     /* left eye fourth */
CONTOUR_EYEL_5,                     /* left eye fifth */
CONTOUR_EYEL_6,                     /* left eye sixth */
CONTOUR_EYEL_7,                     /* left eye seventh */
CONTOUR_EYEL_8,                     /* left eye eighth */
CONTOUR_EYEBROWL_1,                 /* left brow first */
CONTOUR_EYEBROWL_2,                 /* left brow second */
CONTOUR_EYEBROWL_3,                 /* left brow third */
CONTOUR_EYEBROWL_4,                 /* left browfourth */
CONTOUR_EYEBROWL_5,                 /* left brow fifth */
CONTOUR_EYEBROWL_6,                 /* left brow sixth */
CONTOUR_EYEBROWL_7,                 /* left brow seventh */
CONTOUR_EYEBROWL_8,                 /* left brow eighth */
CONTOUR_EYER_1,                     /* right eye first */
CONTOUR_EYER_2,                     /* right eye second */
CONTOUR_EYER_3,                     /* right eye third */
CONTOUR_EYER_4,                     /* right eye fourth */
CONTOUR_EYER_5,                     /* right eye fifth */
CONTOUR_EYER_6,                     /* right eye sixth */
CONTOUR_EYER_7,                     /* right eye seventh */
CONTOUR_EYER_8,                     /* right eye eighth */
CONTOUR_EYEBROWR_1,                 /* right brow first */
CONTOUR_EYEBROWR_2,                 /* right browsecond */
CONTOUR_EYEBROWR_3,                 /* right brow third */
CONTOUR_EYEBROWR_4,                 /* right brow fourth */
CONTOUR_EYEBROWR_5,                 /* right brow fifth */
CONTOUR_EYEBROWR_6,                 /* right browsixth */
CONTOUR_EYEBROWR_7,                 /* right brow seventh */
CONTOUR_EYEBROWR_8,                 /* right brow eighth */
CONTOUR_NOSE_1,                     /* nose first */
CONTOUR_NOSE_2,                     /* nose second */
CONTOUR_NOSE_3,                     /* nose third */
CONTOUR_NOSE_4,                     /* nose fourth */
CONTOUR_NOSE_5,                     /* nose fifth */
CONTOUR_NOSE_6,                     /* nose sixth */
CONTOUR_NOSE_7,                     /* nose seventh */
CONTOUR_NOSE_8,                     /* nose eighth */
```

```
CONTOUR_NOSE_9,                 /* nose ninth */
CONTOUR_NOSE_10,                /* nose tenth */
CONTOUR_NOSE_11,                /* nose eleventh */
CONTOUR_NOSE_12,                /* nose twelfth */
CONTOUR_MOUTH_1,                /* mouth first */
CONTOUR_MOUTH_2,                /* mouth second */
CONTOUR_MOUTH_3,                /* mouth third */
CONTOUR_MOUTH_4,                /* mouth fourth */
CONTOUR_MOUTH_5,                /* mouth fifth */
CONTOUR_MOUTH_6,                /* mouth sixth */
CONTOUR_MOUTH_7,                /* mouth seventh */
CONTOUR_MOUTH_8,                /* mouth eighth */
CONTOUR_MOUTH_9,                /* mouth ninth */
CONTOUR_MOUTH_10,               /* mouth tenth */
CONTOUR_MOUTH_11,               /* mouth eleventh */
CONTOUR_MOUTH_12,               /* mouth twelfth */
CONTOUR_MOUTH_13,               /* mouth thirteenth */
CONTOUR_MOUTH_14,               /* mouth fourteenth */
CONTOUR_MOUTH_15,               /* mouth fifteenth */
CONTOUR_MOUTH_16,               /* mouth sixteenth */
CONTOUR_MOUTH_17,               /* mouth seventeenth */
CONTOUR_MOUTH_18,               /* mouth eighteenth */
CONTOUR_MOUTH_19,               /* mouth nineteenth */
CONTOUR_MOUTH_20,               /* mouth twentieth */
CONTOUR_MOUTH_21,               /* mouth twenty first */
CONTOUR_MOUTH_22,               /* mouth twenty second */
CONTOUR_FACE_1,                 /* Face contourfirst */
CONTOUR_FACE_2,                 /* Face contour second */
CONTOUR_FACE_3,                 /* Face contour third */
CONTOUR_FACE_4,                 /* Face contour fourth */
CONTOUR_FACE_5,                 /* Face contour fifth */
CONTOUR_FACE_6,                 /* Face contour sixth */
CONTOUR_FACE_7,                 /* Face contour seventh */
CONTOUR_FACE_8,                 /* Face contour eighth */
CONTOUR_FACE_9,                 /* Face contour ninth */
CONTOUR_FACE_10,                /* Face contour tenth */
CONTOUR_FACE_11,                /* Face contour eleventh */
CONTOUR_FACE_12,                /* Face contour twelfth */
CONTOUR_FACE_13,                /* Face contour thirteenth */
CONTOUR_FACE_14,                /* Face contour fourteenth */
CONTOUR_FACE_15,                /* Face contour fifteenth */
```

```
CONTOUR_FACE_16,                /* Face contour sixteenth */
CONTOUR_FACE_17,                /* Face contour seventeenth */
CONTOUR_FACE_18,                /* Face contour eighteenth */
CONTOUR_FACE_19,                /* Face contour nineteenth */
CONTOUR_FACE_20,                /* Face contour twentieth */
CONTOUR_FACE_21,                /* Face contour twenty first */
CONTOUR_KIND_MAX                /* number of extraction */
};
```

… and has 87 points. But the size can be changed in the future.

※In the diagram explaining the extracted position coordinates of facial-feature outline detection feature in Chapter 1 and 2, the following points are compliant.

```
CONTOUR_EYEL_1 = 0              /* left eye first */
CONTOUR_EYEL_2 = 1              /* left eye second */
CONTOUR_EYEL_3 = 2              /* left eye third */
CONTOUR_EYEL_4 = 3              /* left eye fourth */
CONTOUR_EYEL_5 = 4              /* left eye fifth */
CONTOUR_EYEL_6 = 5              /* left eye sixth */
CONTOUR_EYEL_7 = 6              /* left eye seventh */
CONTOUR_EYEL_8 = 7              /* left eye eighth */
CONTOUR_EYEBROWL_1 = 8          /* left brow first */
CONTOUR_EYEBROWL_2 = 9          /* left brow second */
CONTOUR_EYEBROWL_3 = 10         /* left brow third */
CONTOUR_EYEBROWL_4 = 11         /* left brow fourth */
CONTOUR_EYEBROWL_5 = 12         /* left brow fifth */
CONTOUR_EYEBROWL_6 = 13         /* left brow sixth */
CONTOUR_EYEBROWL_7 = 14         /* left brow seventh */
CONTOUR_EYEBROWL_8 = 15         /* left brow eighth */
CONTOUR_EYER_1 = 16             /* right eye first */
CONTOUR_EYER_2 = 17             /* right eye second */
CONTOUR_EYER_3 = 18             /* right eye third */
CONTOUR_EYER_4 = 19             /* right eye fourth */
CONTOUR_EYER_5 = 20             /* right eye fifth */
CONTOUR_EYER_6 = 21             /* right eye sixth */
CONTOUR_EYER_7 = 22             /* right eye seventh */
CONTOUR_EYER_8 = 23             /* right eye eighth */
CONTOUR_EYEBROWR_1 = 24         /* right brow first */
CONTOUR_EYEBROWR_2 = 25         /* right brow second */
CONTOUR_EYEBROWR_3 = 26         /* right brow third */
CONTOUR_EYEBROWR_4 = 27         /* right brow fourth */
CONTOUR_EYEBROWR_5 = 28         /* right brow fifth */
CONTOUR_EYEBROWR_6 = 29         /* right brow sixth */
```

```
CONTOUR_EYEBROWR_7 = 30          /* right brow seventh */
CONTOUR_EYEBROWR_8 = 31          /* right brow eighth */
CONTOUR_NOSE_1 = 32              /* nose first */
CONTOUR_NOSE_2 = 33              /* nose second */
CONTOUR_NOSE_3 = 34              /* nose third */
CONTOUR_NOSE_4 = 35              /* nose fourth */
CONTOUR_NOSE_5 = 36              /* nose fifth */
CONTOUR_NOSE_6 = 37              /* nose sixth */
CONTOUR_NOSE_7 = 38              /* nose seventh */
CONTOUR_NOSE_8 = 39              /* nose eighth */
CONTOUR_NOSE_9 = 40              /* nose ninth */
CONTOUR_NOSE_10 = 41             /* nose tenth */
CONTOUR_NOSE_11 = 42             /* nose eleventh */
CONTOUR_NOSE_12 = 43             /* nose twelfth */
CONTOUR_MOUTH_1 = 44             /* mouth first */
CONTOUR_MOUTH_2 = 45             /* mouth second */
CONTOUR_MOUTH_3 = 46             /* mouth third */
CONTOUR_MOUTH_4 = 47             /* mouth fourth */
CONTOUR_MOUTH_5 = 48             /* mouth fifth */
CONTOUR_MOUTH_6 = 49             /* mouth sixth */
CONTOUR_MOUTH_7 = 50             /* mouth seventh */
CONTOUR_MOUTH_8 = 51             /* mouth eighth */
CONTOUR_MOUTH_9 = 52             /* mouth ninth */
CONTOUR_MOUTH_10 = 53        /* mouth tenth */
CONTOUR_MOUTH_11 = 54        /* mouth eleventh */
CONTOUR_MOUTH_12 = 55        /* mouth twelfth */
CONTOUR_MOUTH_13 = 56        /* mouth thirteenth */
CONTOUR_MOUTH_14 = 57        /* mouth fourteenth */
CONTOUR_MOUTH_15 = 58        /* mouth fifteenth */
CONTOUR_MOUTH_16 = 59        /* mouth sixteenth */
CONTOUR_MOUTH_17 = 60        /* mouth seventeenth */
CONTOUR_MOUTH_18 = 61        /* mouth eighteenth */
CONTOUR_MOUTH_19 = 62        /* mouth nineteenth */
CONTOUR_MOUTH_20 = 63        /* mouth twentieth */
CONTOUR_MOUTH_21 = 64        /* mouth twenty first */
CONTOUR_MOUTH_22 = 65        /* mouth twenty second */
CONTOUR_FACE_1 = 66              /* Face contour first */
CONTOUR_FACE_2 = 67              /* Face contour second */
CONTOUR_FACE_3 = 68              /* Face contour third */
CONTOUR_FACE_4 = 69              /* Face contour fourth */
CONTOUR_FACE_5 = 70              /* Face contour fifth */
```

CONTOUR_FACE_6 = 71	/* Face contour sixth */
CONTOUR_FACE_7 = 72	/* Face contour seventh */
CONTOUR_FACE_8 = 73	/* Face contour eighth */
CONTOUR_FACE_9 = 74	/* Face contour ninth */
CONTOUR_FACE_10 = 75	/* Face contour tenth */
CONTOUR_FACE_11 = 76	/* Face contour eleventh */
CONTOUR_FACE_12 = 77	/* Face contour twelfth */
CONTOUR_FACE_13 = 78	/* Face contour thirteenth */
CONTOUR_FACE_14 = 79	/* Face contour fourteenth */
CONTOUR_FACE_15 = 80	/* Face contour fifteenth */
CONTOUR_FACE_16 = 81	/* Face contour sixteenth */
CONTOUR_FACE_17 = 82	/* Face contour seventeenth */
CONTOUR_FACE_18 = 83	/* Face contour eighteenth */
CONTOUR_FACE_19 = 84	/* Face contour nineteenth */
CONTOUR_FACE_20 = 85	/* Face contour twentieth */
CONTOUR_FACE_21 = 86	/* Face contour twenty first */

（Note 4）

The POINT aptFeatureDetail[] arrays in this version mean as 1.2.3 D), and has 38 points. But the size can be changed in the future.

4. How to Use the Library

4.1 Folder Compositions

The folder composition of The Library is shown in the following:

include	The header file of the OKAO Vision Software Library is saved here.
lib	The library file of the OKAO Vision Software Library is saved here.
doc	This specification manual is saved here.

●file composition

Header file

CommonDef.h

OkaoAPI.h

OkaoDtAPI.h

OkaoPtAPI.h

OkaoFrAPI.h

OkaoDbAPI.h

OkaoGeAPI.h

OkaoAgeAPI.h

OkaoYearAPI.h

OkaoSmileAPI.h

OkaoCtAPI.h

ModeDT.h

ModePT.h

ModeFR.h

ModeDB.h

ModeGE.h

ModeAGE.h

ModeYE.h

ModeSMILE.h

ModeCT.h

library

OkaoXXX.dll

OkaoXXX.lib

* XXX is depend on version of SDK.

4.2 How to Use Library

Microsoft Visual C++6.0

In the application project, it is necessary to add the folder where the header file is to the include path. It is also necessary to specify the folder where individual library files are for the library path.

To use the Library, it is necessary to specify to include header files. It is also necessary to specify to link individual library files.

4.3 Restrictions of the Library

We do not guarantee the operation in the event that the user inputs numerical values that are not assumed in our software specifications. One handle cannot be operated with different threads simultaneously. When operating with different threads, it is necessary to create a handle for each thread.
However, only the album handle is not thread-free. The processing subject for this is OKAO_CreateDatabase(), OKAO_DeleteDatabase().

4.4 Sample code

4.4.1 Sample of face detection

```
void sample( void )
{
        unsigned char       *pbySrcImage = NULL;        /* original input image */
        HDETECTION          hDT;                        /* Face detection handle */
        HDTRESULT           hResult;                    /* Face detection result handle */
        int                 nWidth, nHeight, nDepth;
        int                 nCount;

        /* Initialization of the library */
        OKAO_Initialize( OKAO_DETECTION );

        /* Create the handle of face detection module*/
        hDT = OKAO_CreateDetection( DT_MODE_DEFAULT );
        /* Create the handle of face detection result */
        hResult = OKAO_CreateDtResult();

        /* Set the image data to pbySrcImage, nWidth, nHeight, nDepth */
        /* read images */

        /* perform face detection*/
        OKAO_Detection(hDT, pbySrcImage, nWidth, nHeight, nDepth, hResult);
        /* Get the number of face detection*/
        OKAO_GetDtFaceCount(hResult, &nCount);

        /* Free memory*/
        free(pbySrcImage);
        OKAO_DeleteDtResult(hResult);
        OKAO_DeleteDetection(hDT);

        /* Termination of the library*/
        OKAO_Terminate( OKAO_DETECTION );

        return;
}
```

4.4.2 Sample of default parameters for face detection

```
void sample( void )
{
        /* The sample of default parameters for face detection */

        HDETECTION      hDT;            /* Face detection handle */
        int             nMax;           /* Maximum number of faces detected */
        int             nSize;          /* Size of the face detected*/
        int             nDirection;     /* Direction of face detection*/
        enum DETECT_ANGLE angle;                /* Range of the angle for face detection*/
        int             nTimeout;       /* Timeout (milli-seconds)*/
        int             nStep;          /* Search step*/
        int             nThreshold;     /* Threshold*/

        /* Initialization of the library*/
        OKAO_Initialize( OKAO_DETECTION );

        /* Create the handle of face detection module*/
        hDT = OKAO_CreateDetection( DT_MODE_DEFAULT );

        /*Get the maximum number of faces detected*/
        OKAO_GetDtMaxFaceNumber(hDT, &nMax);
        printf("Face detection max num = %d¥n", nMax);

        /* Minimum/Maximum size of faces detected*/
        OKAO_GetDtFaceSizeRatioRange(hDT, &nMinSize, &nMaxSize);
        printf("max size of faces detected = %d%¥n", nMaxSize);
        printf("min size of faces detected = %d%¥n", nMinSize);

        /* Direction of the face*/
        OKAO_GetDtDetectDirection(hDT, &nDirection);
        if (nDirection & DETECT_UP) {
                printf("Facial direction  ↑ ");
        } else {
                printf("Facial direction ");
        }
        if (nDirection & DETECT_RIGHT) {
                printf("→");
        }
```

```
        if (nDirection & DETECT_LEFT) {
                printf("←");
        }
        if (nDirection & DETECT_DOWN) {
                printf("↓¥n");
        } else {
                printf("¥n");
        }

        /* Range of the facial angle*/
        OKAO_GetDtDetectAngle(hDT, &angle);
        printf("Range of the facial angle    %s¥n", (angle == DETECT_1ANGLE) ?
                "One angl（0°）" : " Three angles（0°、-30°、+30°）");    e

        /* Search step*/
        OKAO_GetDtStep(hDT, &nStep);
        printf("Search step = %2.1fpixel¥n", (float)nStep / 10);

        /* Threshold*/
        OKAO_GetDtThreshold(hDT, &nThreshold);
        printf("Threshold    = %d¥n", nThreshold);


        OKAO_DeleteDetection(hDT);

        /* Termination of the library for face detection*/
        OKAO_Terminate( OKAO_DETECTION );

        return;
}
```

4.4.3 Sample of facial parts detection

```
void sample( void )
{
        HDETECTION      hDT ;
        HPOINTER        hPT ;

        HDTRESULT       hDtResult ;
        HPTRESULT       hPtResult ;

        unsigned char   *pbySrcImage1 ;
        int             nWidth, nHeight, nDepth ;
        int nFaceCount;
        long lRet;
        int nI;
        POINT ptLeftTop, ptRightTop, ptLeftBottom, ptRightBottom ;
        int nConf;
        POINT aptFeatures[ FEATURE_KIND_MAX ] ;
        int     aConf[ FEATURE_KIND_MAX ] ;
        int nUpDown, nLeftRight, nRoll;
        int nDirConf;

        OKAO_Initialize( OKAO_ALL ) ;

        /*Create the handle of the face detection module*/
        hDT = OKAO_CreateDetection( DT_MODE_DEFAULT ) ;
        /*Create the handle of the face detection result*/
        hDtResult = OKAO_CreateDtResult() ;

        /* Create the handle of the facial parts detection module*/
        hPT = OKAO_CreatePointer( PT_MODE_DEFAULT ) ;
        /* Create the handle of the facial parts detection result */
        hPtResult = OKAO_CreatePtResult();

        /* Set the image data to pbySrcImage1, nWidth, nHeight, nDepth */
        /* Read images for the registration*/

        /* Perform face detection*/
        OKAO_Detection( hDT, pbySrcImage1, nWidth, nHeight, nDepth, hDtResult );
```

```
        /*Get the face detection result*/
        lRet = OKAO_GetDtFaceCount( hDtResult, &nFaceCount );
        for( nI = 0; nI < nFaceCount; nI++ ) {
                OKAO_GetDtCorner( hDtResult, nI,
                        &ptLeftTop, &ptRightTop, &ptLeftBottom, &ptRightBottom, &nConf );
        }

        /* Perform facial parts detection*/
        OKAO_SetPtPosition( hPT, hDtResult, 0 ) ;
        OKAO_Pointer( hPT, pbySrcImage1, nWidth, nHeight, nDepth, hPtResult, &nConf ) ;

        /* Get the facial parts detection result*/
        OKAO_GetPtPoint( hPtResult , aptFeatures, aConf ) ;

        /* Get the face direction result */
        OKAO_GetPtDirection(hPtResult, &nUpDown, &nLeftRight, &nRoll, &nDirConf);

        /* Free memory*/
        free( pbySrcImage1 ) ;
        OKAO_DeletePtResult( hPtResult ) ;
        OKAO_DeletePointer( hPT ) ;
        OKAO_DeleteDtResult( hDtResult ) ;
        OKAO_DeleteDetection( hDT ) ;
        OKAO_Terminate( OKAO_ALL ) ;

        return;
}
```

4.4.4 Sample of face recognition

```
void sample( void )
{
        HDETECTION      hDT;
        HPOINTER        hPT;
        HFACERECOG      hFR;
        HDTRESULT       hDtResult;
        HPTRESULT       hPtResult;
        HFRRESULT       hFrResult;
        HALBUMDB        hDB;

        unsigned char   *pbySrcImage1;
        unsigned char   *pbySrcImage2;
        int             nWidth, nHeight, nDepth;
        int nFaceCount;
        long lRet;
        int nI;
        POINT ptLeftTop, ptRightTop, ptLeftBottom, ptRightBottom;
        int nConf;
        FR_USER_ID uID[USER_ID_LENGTH * 5], rID[USER_ID_LENGTH * 5];
        int     aIDConf[5];

        OKAO_Initialize( OKAO_ALL );

        /* Create the handle of the face detection module*/
        hDT = OKAO_CreateDetection( DT_MODE_DEFAULT );
        /* Create the handle of the face detection result */
        hDtResult = OKAO_CreateDtResult();
        /* Create the handle of the facial parts detection module*/
        hPT = OKAO_CreatePointer( PT_MODE_DEFAULT );
        /* Create the handle of the facial parts detection result*/
        hPtResult = OKAO_CreatePtResult();
        /* Create the handle of the face recognition */
        hFR = OKAO_CreateFaceRecognition( FR_MODE_DEFAULT );
        /* Create the handle of the face identification result */
        hFrResult = OKAO_CreateIdentifyResult( 5 );
        /* Create the handle of the database */
        hDB = OKAO_CreateDatabase( DB_MODE_DEFAULT, 100, 3 );
```

```
/* Set the image data to pbySrcImage1, nWidth, nHeight, nDepth*/
/* Read images for the registration*/

/* Perform face detection*/
OKAO_Detection( hDT, pbySrcImage1, nWidth, nHeight, nDepth, hDtResult );

/* Get the face detection result*/
lRet = OKAO_GetDtFaceCount( hDtResult, &nFaceCount );
for( nI = 0; nI < nFaceCount; nI++ ) {
        OKAO_GetDtCorner( hDtResult, nI,
                &ptLeftTop, &ptRightTop, &ptLeftBottom, &ptRightBottom, &nConf );
}

/* Perform facial parts detection*/
OKAO_SetPtPosition( hPT, hDtResult, 0 );
OKAO_Pointer( hPT, pbySrcImage1, nWidth, nHeight, nDepth, hPtResult, &nConf );

/* Create the face recognition data*/
OKAO_SetFrDataFromPtResult( hFR, pbySrcImage1, nWidth, nHeight, nDepth,
                                        hPtResult );

/* Regist to the database */
memcpy(rID, "0000000000000001", USER_ID_LENGTH);
OKAO_RegistDatabase( hDB, hFR, nWidth, nHeight, nDepth, rID, 1 );

/* Set the image data to pbySrcImage2, nWidth, nHeight, nDepth*/
/* Read images for the verification */

/* Perfrom face detection*/
OKAO_Detection( hDT, pbySrcImage2, nWidth, nHeight, nDepth, hDtResult );

/* Get the face detction result*/
OKAO_GetDtFaceCount( hDtResult, &nFaceCount );
for( nI = 0; nI < nFaceCount; nI++ ) {
        OKAO_GetDtCorner( hDtResult, nI,
                &ptLeftTop, &ptRightTop, &ptLeftBottom, &ptRightBottom, &nConf);
}

/* Perform facial parts detection*/
OKAO_SetPtPosition( hPT, hDtResult, 0 );
```

```
        OKAO_Pointer( hPT, pbySrcImage2, nWidth, nHeight, nDepth, hPtResult, &nConf );

        /* Create the face recognition data*/
        OKAO_SetFrDataFromPtResult( hFR, pbySrcImage2, nWidth, nHeight, nDepth,
                                                    hPtResult );

        /* Identification*/
        OKAO_FrIdentify( hFR, nWidth, nHeight, nDepth, hDB, 0, hFrResult );
        /* Get the face identification result */
        OKAO_GetFrResult( hFrResult, uID, nIDConf, &nI );
        /* Looking for uID… */

        OKAO_Deletedatabase( hDB );
        OKAO_DeleteFrResult( hFrResult );
        OKAO_DeleteFaceRecognition( hFR );
        free( pbySrcImage1 ) ;
        free( pbySrcImage2 ) ;
        OKAO_DeletePtResult( hPtResult ) ;
        OKAO_DeletePointer( hPT ) ;
        OKAO_DeleteDtResult( hDtResult ) ;
        OKAO_DeleteDetection( hDT ) ;
        OKAO_Terminate( OKAO_ALL ) ;

        return;
}
```

4.4.5 Sample of gender estimation

```
void sample( void )
{
    HPTRESULT          hPtResult;        /* Facial parts detection result handle */
    HGENDER            hGE;              /* Gender estimation handle */
    HGENRESULT         hGeResult;        /* Gender estimation result handle */

    unsigned char   *pbySrcImage = NULL;
    int             nWidth, nHeight, nDepth;
    BOOL            bMale;
    int             nConfidence;

    /* Initialization of the library for gender estimation*/
    OKAO_Initialize( OKAO_GENDER );
    /* Create the handle of the gender estimation module*/
    hGE = OKAO_CreateGender( GE_MODE_DEFAULT );
    hGeResult = OKAO_CreateGenderResult();

    /* Set the image data to pbySrcImage, nWidth, nHeight, nDepth */
    /*Read images*/

    /*Perform facial parts detection*/
    /* Set the coordinates of facial features*/
    OKAO_SetGePoint(hGE, hPtResult);

    /* Perform gender estimation*/
    OKAO_GenderEstimate(hGE, pbySrcImage, nWidth, nHeight, nDepth, hGeResult);
    OKAO_GetGenderResult(hGeResult, &bMale, &nConfidence);

    /* Free memory */
free(pbySrcImage);
    OKAO_DeleteGenderResult(hGeResult );
    OKAO_DeleteGender( hGE );

    /* Termination of the library for gender estimation*/
    OKAO_Terminate( OKAO_GENDER );

return;
```

```
}
```

4.4.6 Sample of age group estimation

```
void sample( void )
{
    HPTRESULT           hPtResult;          /* Facial parts detection result handle */
    HAGE                hAGE;               /* Age group estimation handle */
    HAGERESULT          hAgeResult;         /* Age group estimation result handle */

    unsigned char   *pbySrcImage = NULL;
    int             nWidth, nHeight, nDepth;

    int             nLower, nUpper;
    int             nConfidence;
    int             nDivision[4] = {10, 20, 40, 60};        /* the age division */

    /*Initialization of the library for age group estimation*/
    OKAO_Initialize( OKAO_AGE );

    /* Create the handle of the age group estimation module*/
    hAGE = OKAO_CreateAge( AGE_MODE_DEFAULT );
    hAgeResult = OKAO_CreateAgeResult( );

    /* Set the image data to pbySrcImage, nWidth, nHeight, nDepth */
    /* Read images*/

    /* Perform facial parts detection*/

    /* Set the coordinates of facial features*/
    OKAO_SetAgePoint(hAGE, hPtResult);

    /* Setting of the age division */
    OKAO_SetAgeDivision(hAGE, nDivision, 4);

     /* Perform age group estimation*/
    OKAO_AgeEstimate(hAGE, pbySrcImage, nWidth, nHeight, nDepth, hAgeResult);
OKAO_GetAgeResult(hAgeResult, &nLower, &nUpper, &nConfidence);

    /* Free memory*/
free(pbySrcImage);
```

```
    OKAO_DeleteAgeResult( hAgeResult );
    OKAO_DeleteAge( hAGE );

    /*Termination of the library for age group estimation*/
    OKAO_Terminate( OKAO_AGE );

    return;
}
```

4.4.7 Sample of age estimation

```
void sample( void )
{
    HPTRESULT           hPtResult;          /* Facial parts detection result handle */
    HYEAR               hYEAR;              /* Age estimation handle */
    HYEARRESULT         hYearResult;        /* Age estimation result handle */

    unsigned char   *pbySrcImage = NULL;
    int             nWidth, nHeight, nDepth;
    int             nYear, nConfidence;

    /*Initialization of the library for age estimation*/
    OKAO_Initialize( OKAO_YEAR );

    /* Create the handle of the age estimation module*/
    hYEAR = OKAO_CreateYear( YEAR_MODE_DEFAULT );
    hYearResult = OKAO_CreateYearResult();

    /* Set the image data to pbySrcImage, nWidth, nHeight, nDepth */
    /* Read images*/

    /* Perform facial parts detection*/

    /* Set the coordinates of facial features*/
    OKAO_SetYearPoint(hYEAR, hPtResult);

    /* Perform age estimation*/
    OKAO_YearEstimate(hAGE, pbySrcImage, nWidth, nHeight, nDepth, hYearResult);
    OKAO_GetYearResult(hYearResult, &nYear, &nConfidence);

    /* Free memory*/
free(pbySrcImage);
    OKAO_DeleteYearResult( hYearResult );
    OKAO_DeleteYear( hYEAR );

    /*Termination of the library for age estimation*/
    OKAO_Terminate( OKAO_YEAR);
    return;
```

```
}
```

4.4.8 Sample of smile degree estimation

```
void sample( void )
{
    HPTRESULT           hPtResult;          /* Facial parts detection result handle */
    HSMILE              hSMILE;             /* Smile degree estimation handle */

    unsigned char   *pbySrcImage = NULL;
    int             nWidth, nHeight, nDepth;
    int             nSmile;

    /* Initialization of the library for smile degree estimation*/
    OKAO_Initialize( OKAO_SMILE );

    /* Create the handle of the smile degree estimation module*/
    hSMILE = OKAO_CreateSmile( SMILE_MODE_DEFAULT );

    /* Set the image data to pbySrcImage, nWidth, nHeight, nDepth */
    /*Read images*/

    /*Perform facial parts detection*/

    /* Set the coordinates of facial features*/
    OKAO_SetSmilePoint(hSMILE, hPtResult);

    /* Perform smile degree estimation */
    OKAO_SmileCheck(hSMILE, pbySrcImage, nWidth, nHeight, nDepth, &nSmile);

    /* Free memory */
free(pbySrcImage);
    OKAO_DeleteSmile( hSMILE );

    /* Termination of the library for smile degree estimation */
    OKAO_Terminate( OKAO_SMILE );

return;
}
```

4.4.9 Sample of facial feature outline detection

```
void sample( void )
{
        HDETECTION      hDT ;
        HPOINTER        hPT ;
        HCONTOUR        hCT ;

        HDTRESULT       hDtResult ;
        HPTRESULT       hPtResult ;
        HCTRESULT       hCtResult ;

        unsigned char   *pbySrcImage1 ;
        int             nWidth, nHeight, nDepth ;
        int nFaceCount;
        long    lRet;
        int     nI;
        POINT ptLeftTop, ptRightTop, ptLeftBottom, ptRightBottom ;
        int nConf;
        POINT aptFeatures[ FEATURE_KIND_MAX ] ;
        int     aConf[ FEATURE_KIND_MAX ] ;
        POINT aptContour[ CONTOUR_KIND_MAX ] ;

        OKAO_Initialize( OKAO_ALL ) ;

        /* Create the face detection module*/
        hDT = OKAO_CreateDetection( DT_MODE_DEFAULT ) ;
        /* Create the handle of the face detection result*/
        hDtResult = OKAO_CreateDtResult() ;

        /* Create the handle of the facial parts detection module*/
        hPT = OKAO_CreatePointer( PT_MODE_DEFAULT ) ;
        /* Create the handle of the facial parts detection result*/
        hPtResult = OKAO_CreatePtResult();

        /* Create the handle of the facial feature outline detection module*/
        hCT = OKAO_CreateContour( CT_MODE_DEFAULT ) ;
        /*Create the handle of the facial feature outline detection result*/
        hCtResult = OKAO_CreateCtResult();
```

```
        /*Set the image data to pbySrcImage1, nWidth, nHeight, nDepth */
        /* Read images for the registration*/

        /*    Perform face detection*/
        OKAO_Detection( hDT, pbySrcImage1, nWidth, nHeight, nDepth, hDtResult );

        /* Get the face detection result*/
        lRet = OKAO_GetDtFaceCount( hDtResult, &nFaceCount );
        for(nI = 0; nI < nFaceCount; nI++ ) {
                OKAO_GetDtCorner( hDtResult, nI,
                        &ptLeftTop, &ptRightTop, &ptLeftBottom, &ptRightBottom, &nConf );
        }

        /* Perform facial parts detection */
        OKAO_SetPtPosition( hPT, hDtResult, 0 ) ;
        OKAO_Pointer( hPT, pbySrcImage1, nWidth, nHeight, nDepth, hPtResult, &nConf ) ;

        /* Get the facial featu re deteciton result*/
        OKAO_GetPtPoint( hPtResult , aptFeatures, aConf ) ;

        /* Perform facial feature outline detection*/
        OKAO_SetCtPosition( hCT, hPtResult ) ;
        OKAO_Contour( hCT, pbySrcImage1, nWidth, nHeight, nDepth, hCtResult ) ;

        /* Get the facial feature outline deteciton result*/
        OKAO_GetCtPoint( hCtResult, aptContour ) ;

        /* Free memory */
        free( pbySrcImage1 ) ;
        OKAO_DeleteCtResult( hCtResult ) ;
        OKAO_DeleteContour( hCT ) ;
        OKAO_DeletePtResult( hPtResult ) ;
        OKAO_DeletePointer( hPT ) ;
        OKAO_DeleteDtResult( hDtResult ) ;
        OKAO_DeleteDetection( hDT ) ;
        OKAO_Terminate( OKAO_ALL ) ;

        return;
}
```

OMRON Corporation
Core Technology Center
OKAO VISION Project

【2010.02.24（OkaoXB1）】