



API Development Manual:

AMTFaceLite SDK For Android

API Version: 12.0

Doc Version: 1.0

June 2022

Thank you for choosing our product. Please read the instructions carefully before operation. Follow these instructions to ensure that the product is functioning properly. The images shown in this manual are for illustrative purposes only.



For further details, please visit our Company's website
www.armatura.us.

Copyright © 2022 ARMATURA LLC. All rights reserved.

Without the prior written consent of ARMATURA LLC, no portion of this manual can be copied or forwarded in any way or form. All parts of this manual belong to ARMATURA and its subsidiaries (hereinafter the "Company" or "ARMATURA").

Trademark

ARMATURA is a registered trademark of ARMATURA LLC. Other trademarks involved in this manual are owned by their respective owners.

Disclaimer

This manual contains information on the operation and maintenance of the ARMATURA product. The copyright in all the documents, drawings, etc. in relation to the ARMATURA supplied product vests in and is the property of ARMATURA. The contents hereof should not be used or shared by the receiver with any third party without express written permission of ARMATURA.

The contents of this manual must be read as a whole before starting the operation and maintenance of the supplied product. If any of the content(s) of the manual seems unclear or incomplete, please contact ARMATURA before starting the operation and maintenance of the said product.

It is an essential pre-requisite for the satisfactory operation and maintenance that the operating and maintenance personnel are fully familiar with the design and that the said personnel have received thorough training in operating and maintaining the machine/unit/product. It is further essential for the safe operation of the machine/unit/product that personnel have read, understood, and followed the safety instructions contained in the manual.

In case of any conflict between terms and conditions of this manual and the contract specifications, drawings, instruction sheets or any other contract-related documents, the contract conditions/documents shall prevail. The contract specific conditions/documents shall apply in priority.

ARMATURA offers no warranty, guarantee, or representation regarding the completeness of any information contained in this manual or any of the amendments made thereto. ARMATURA does not extend the warranty of any kind, including, without limitation, any warranty of design, merchantability, or fitness for a particular purpose.

ARMATURA does not assume responsibility for any errors or omissions in the information or documents which are referenced by or linked to this manual. The entire risk as to the results and performance obtained from using the information is assumed by the user.

ARMATURA in no event shall be liable to the user or any third party for any incidental, consequential, indirect, special, or exemplary damages, including, without limitation, loss of business, loss of profits, business interruption, loss of business information or any pecuniary loss, arising out of, in connection with, or relating to the use of the information contained in or referenced by this manual, even if ARMATURA has been advised of the possibility of such damages.

This manual and the information contained therein may include technical, other inaccuracies, or typographical errors. ARMATURA periodically changes the information herein which will be incorporated into new additions/amendments to the manual. ARMATURA reserves the right to add, delete, amend, or modify the information contained in the manual from time to time in the form of circulars, letters, notes, etc. for better operation and safety of the machine/unit/product. The said additions or amendments are meant for improvement /better operations of the machine/unit/product and such amendments shall not give any right to claim any compensation or damages under any circumstances.

ARMATURA shall in no way be responsible (i) in case the machine/unit/product malfunctions due to any non-compliance of the instructions contained in this manual (ii) in case of operation of the machine/unit/product beyond the rate limits (iii) in case of operation of the machine and product in conditions different from the prescribed conditions of the manual.

The product will be updated from time to time without prior notice. The latest operation procedures and relevant documents are available on <http://www.armatura.com>.

If there is any issue related to the product, please contact us.

ARMATURA Headquarters

Address 190 Bluegrass Valley Pkwy,
 Alpharetta, GA 30005, USA.

For business-related queries, please write to us at: info@armatura.us.

To know more about our global branches, visit www.armatura.us.

About the Company

ARMATURA is a leading global developer and supplier of biometric solutions which incorporate the latest advancements in biometric hardware design, algorithm research & software development. ARMATURA holds numerous patents in the field of biometric recognition technologies. Its products are primarily used in business applications which require highly secure, accurate and fast user identification.

ARMATURA biometric hardware and software are incorporated into the product designs of some of the world's leading suppliers of workforce management (WFM) terminals, Point-of-Sale (PoS) terminals, intercoms, electronic safes, metal key lockers, dangerous machinery, and many other products which heavily rely on correctly verifying & authenticating user's identity.

About the Manual

This manual introduces the operations of **AMTFaceLite SDK For Android**.

All figures displayed are for illustration purposes only. Figures in this manual may not be exactly consistent with the actual products.

Document Conventions

Conventions used in this manual are listed below:

GUI Conventions

For Software	
Convention	Description
Bold font	Used to identify software interface names e.g. OK , Confirm , Cancel .
>	Multi-level menus are separated by these brackets. For example, File > Create > Folder.
For Device	
Convention	Description
< >	Button or key names for devices. For example, press <OK>.
[]	Window names, menu items, data table, and field names are inside square brackets. For example, pop up the [New User] window.
/	Multi-level menus are separated by forwarding slashes. For example, [File/Create/Folder].

Symbols






Convention	Description
	This represents a note that needs to pay more attention to.
	The general information which helps in performing the operations faster.
	The information which is significant.
	Care taken to avoid danger or mistakes.
	The statement or event that warns of something or that serves as a cautionary example.

Table of Contents

1	INTRODUCTION	6
1.1	OVERVIEW OF THE SDK.....	6
1.2	FEATURE OF THE SDK.....	7
1.3	ADVANTAGE OF THE SDK.....	8
2	TECHNICAL SPECIFICATIONS	9
2.1	ARCHITECTURE.....	10
2.1.1	SDK FILES.....	10
2.1.2	DEVELOPMENT SETUP.....	10
2.1.3	USB INFORMATION AND PERMISSION CONFIGURATION.....	11
2.2	PROGRAMMING GUIDE.....	11
2.2.1	REGISTRATION PROCESS.....	11
2.2.2	VERIFICATION/IDENTIFICATION PROCESS.....	13
3	SDK INTERFACE DESCRIPTION	16
3.1	TEMPLATE FORMAT.....	16
3.1.1	AMTNIRFACEERVICE12.CLASS	17
	APPENDIX	32
	APPENDIX 1: GLOSSARY	32

1 Introduction

This document will provide with basic SDK development guide and technical background to help with better use of our AMTFaceLite SDK document. From the perspective of a developer, the key design objective of this SDK is its compatibility and ease of execution.

This development manual contains the product development documentation for developers that describes the functions provided by the SDK and its related usage, which eases the development environment.

The following sections explain all the required information on how to perform and integrate AMTFaceLite SDK.

1.1 Overview of the SDK

AMTFaceLite SDK is a wrapper of Armatura near-Infrared light face recognition algorithm. It is an excellent near-infrared face recognition algorithm based on the indoor face recognition algorithm, developed to resist complex ambient light and the needs of large capacity recognition. In the case of ensuring a very low FRR, the algorithm focuses on improving the wide adaptation to the environment and user habits, thereby greatly improving the robustness and success rate of face recognition.

The SDK provides rich interfaces to access the algorithm's functionalities for face recognition process, including face detection, feature extraction, liveness detection, template creation, and face identification.

The FaceLite SDK utilizes the widely supported libusb API for face module communication, supports common-used operation systems, and frees the developers from intimidating hardware operations. It is a developer-friendly toolkit to empower the biometric features on the software application with easy pickup.

The simple library components aid in supporting and enhancing the security requirements through biometric facial recognition which avoids spoofing and has been widely used in various systems, including attendance, security, video monitoring, and so on.

1.2 Feature of the SDK

- **Face Focusing Method to Enhance Image Quality:**

The FaceLite algorithm takes face focusing method to enhance the image quality which significantly reduces the facing-light and back-light impact on the captured image.

- **Stable Face Features Boost Recognition Accuracy and Performance**

The FaceLite algorithm can detect different levels (18,40 or 120) of key face feature points and their positions in milliseconds, such as eyes, lips, nose tips, and contours. Such key points are stable face features and can be recognized from the deliberated and unintentional variations in the captured face images. It boosts the algorithm to achieve face recognition accuracy and performance.

- **Multi-dimensional Face Feature Template for Robust Face Recognition:**

The FaceLite algorithm calculates multi-dimensional features from the collected multiple templates (5 consecutive templates) to generate one enrollment template which minimizes the side impact from hats, scarves, dark glasses, or other attachments during the registration process. This improves the recognition robustness.

- **Liveness Detection:**

The FaceLite algorithm can effectively detect a fake face from a digital photo, printed color photo, Black & White face image, or a recorded video of a live face.

- **High Recognition Performance**

Based on the stable face features, the FaceLite algorithm takes the multi-level matching mode with optimized classifier parameters to match the candidate in the large-volume template library within a second.

- **Automatic Update the Template Library:**

The FaceLite algorithm tracks face features and automatically updates the face template into the template library, such an adaptive approach can keep the template stay with the user's current appearance and lower the rejection rate caused by changes in the user's appearance and hairstyle.

- **Algorithm Integrity:**

Combined with Armatura near-infrared light face module, the FaceLite algorithm ensures the quality of images by maintaining data integrity for a genuine and accurate image process.

1.3 Advantage of the SDK

- Easy to use by other developers.
- Thorough documentation to explain how your code works.
- Enough functionality so it adds value to other applications.
- Does not negatively impact.
- Plays well with other SDKs.

2 Technical Specifications

Development Language

This SDK provides a jar package to support Java development.

Platform Requirements

This SDK supports Android 4.0 or later.

Technical Parameters

Parameter	Description
Template size	< 29 KB
Posture adaptability	Yaw $\leq 25^\circ$, Pitch $\leq 25^\circ$, Roll $\leq 25^\circ$
Face detection	< 50 ms
Face feature extraction	< 200 ms
Face verification/identification (1:6000)	< 100 ms
Number of face templates supported	6000
Accuracy	TAR=98.6% when FAR=0.001%

The preceding algorithm capability indicators are all measured based on an actual image data set (resolution of 480 x 640) and quad-core Cortex-A9, 1.5 GHz processor.

2.1 Architecture

2.1.1 SDK Files

- Copy the following files from the libs directory to the app libs directory.
- Add the jniLib dependent library to the android node in the gradle file.

```
sourceSets.main
{
    jniLibs.srcDir 'libs'
    jni.srcDirs = []
}
```

- Algorithm model file.

To call the AMTNIRFaceService12.init function in order to initialize the algorithm, it is required to set the data of the matching face.dat file on the algorithm interface.

File Name	Description
amtandroidnirfaceservice.jar	Java library for the algorithm interface
libamtirface.so	Dynamic link library for the algorithm
face.dat	Algorithm model file

2.1.2 Development Setup

The dynamic link libraries cannot be compressed and need to be added in the app/build.gradle file.

```
android{
    packagingOptions {
        doNotStrip "**/armeabi-v7a/*.so"
        doNotStrip "**/arm64-v8a/*.so"
    }
}
```

2.1.3 USB Information and Permission Configuration

USB dongle

- The AMTNIRFace12.0 algorithm uses a dongle for user authorization. The dongle is usually built into the face recognition devices. Therefore, you do not require an external dongle.
- Vendor ID: 0x34c9
- Product ID: 0x0600

Permission configuration

See the demo or "Android USB Host Helper.md."

2.2 Programming Guide

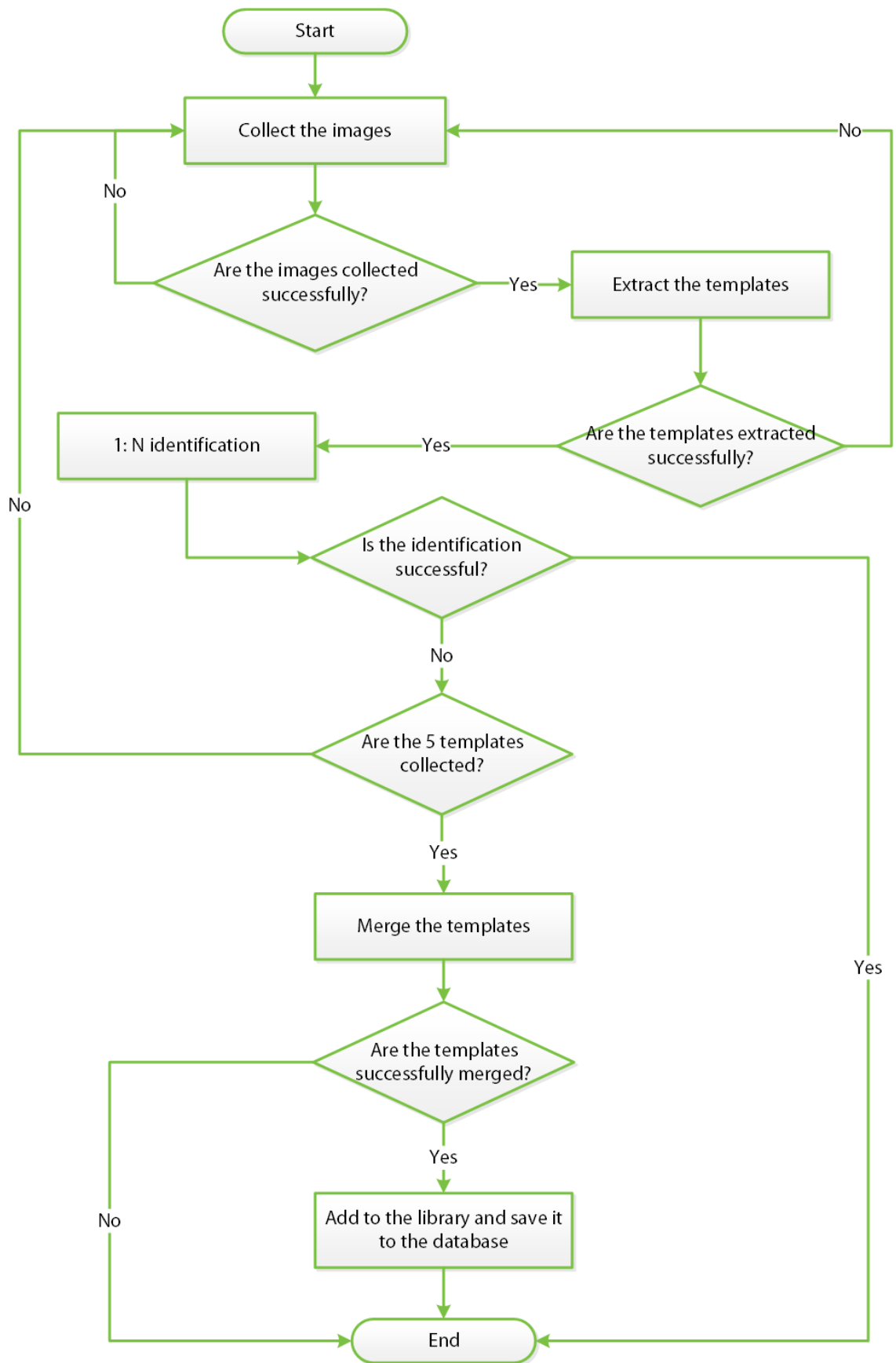
This section describes the key processes of the face recognition to help the developers understand the face registration and verification/identification processes implemented by the AMTNIRFace12.0 algorithm.

2.2.1 Registration Process

In the face registration process, the face recognition application must capture five verification/identification templates and merge them into a registered template.

For more details about the different types of templates, see the [SDK Interface Description](#).

Registration Process Flow Using 1:N Identification



Process Description:

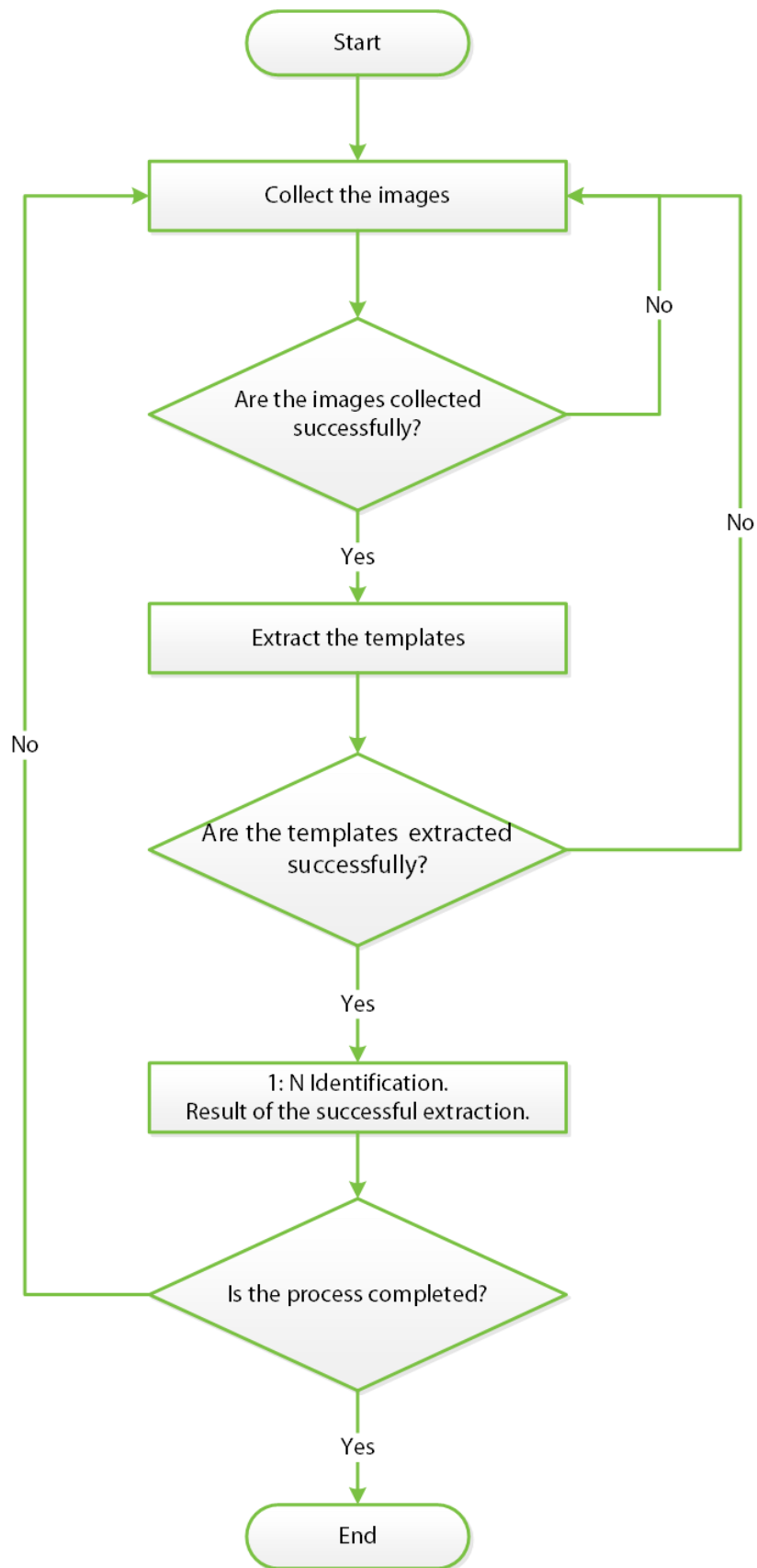
- The application calls the face capturing SDK to capture the face images.
- Once the face images are captured successfully, the application calls the extract function to extract the template.
- The application calls the dbIdentify 1:N function to determine whether the current extracted template has been registered.
- And, if it has been registered, the application returns a message and ends the registration process.
- But, if the extracted template is not registered, the application checks whether five templates have been captured.
- And if less than five templates have been captured, the application continues to capture the next template.
- After capturing five templates, the application merges the templates into a registered template. If the registration fails, the application returns a message and ends the registration process.
- If the registration succeeds, the application calls the dbAdd function to add the registered template to the database.
- And thus, ends the process.

2.2.2 Verification/Identification Process

1:N Identification Process

To implement 1:N face identification, it is required to add all the registered templates to the database. It is recommended to call the dbAdd function to add all the registered templates to the database after successful initialization of the algorithm.

Identification Process Flow



Process Description:

- The application calls the face capturing SDK to capture the face images.
- After the face image is captured successfully, the application calls the extract function to extract template.
- The application calls the dbIdentify 1:N function to compare the current template with the registered templates.
- And once the registered template is identified, the application ends the registration process.

3 SDK Interface Description

3.1 Template Format

Template Type	Data Length	Description
Verification/ Identification template	<ul style="list-style-type: none">• 21072 Bytes• MTNIRFaceService12.FIX_VER_TEMPLA TE_LEN	The data length of a pre-registered or verification/identification template
Registration template	<ul style="list-style-type: none">• 28992 Bytes• MTNIRFaceService12.FIX_REG_TEMPL ATE_LEN	The data length of a registration template.

3.1.1 AMTNIRFaceService12.class

This is an algorithm interface class.

Function List

Class/Interface	Description
com.armatura.amtinfraredservice.nirface AMTNIRFaceService12.class	Algorithm interface class
init	Set the algorithm model and initializes the algorithm
free	Release the algorithm resources
extractFromBitmap	Extract a verification/identification template from the bitmap object.
extractFromNV21	Extract a verification/identification template from NV21 image data.
extractFromGrayscaleData	Extract a verification/identification template from 256-gray scale pixel data
getTemplateQlt	Get the quality of the verification/identification template
mergeRegTemplate	Merge the verification/identification templates into a registration template
verify	Perform 1:1 face verification
dbAdd	Add the registered template to the database
dbDel	Remove the specified face template from the database
dbClear	Clear the database
dbCount	Get the total number of face templates stored in the database
dbIdentify	Perform 1:N face identification
dbVerify	Perform 1:1 face verification
getFacePosition	Get the position coordinates of the face

init

Function Syntax

```
public static int init(Context context, byte[] faceData)
```

Description

Initializes the algorithm.

Parameters

Parameter	Description
context	In: Application context.
faceData	In: Data of the algorithm model file

Returns

0	Success
-11001	Parameter error
-11006	Dongle connection failure
Other values	Initialization failure (wrong license, data loading by face.dat, or memory allocation failure)

Remarks

It is required to get the USB permission for the dongle before initializing the algorithm

Click [here](#) to view the Function List.

free

Function Syntax

```
public static void free()
```

Description

Releases algorithm resources.

Remarks

Click [here](#) to view the Function List.

extractFromBitmap

Function Syntax

```
public static int extractFromBitmap(Bitmap bitmap, byte[] verTemplate, int[]exp,int expmode)
```

Description

Detects the face from the bitmap and extracts a verification/identification template.

Parameters

Parameter	Description
bitmap	In: Infrared face image.
verTemplate	Out: Face verification/identification template, for which the caller needs to allocate memory.
exp	Out: exposure value(If exp[0] !=0, you need set the exposure value to camera, ex:amtFaceCamera.setParameter(10107, ToolUtils.intToByteArray(exp[0]), 4))
expmode	In: 0 means register, 1 means verify or identify.
For the fixed length of the verification/identification template, see AMTNIRFaceService12. FIX VER TEMPLATE LEN .	

Returns

0	Success
-1	Face recognition failure caused by an error in image size conversion
-15	Template extraction failure
-1000	Dongle error
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized
-11003	Invalid bitmap

Remarks

Click [here](#) to view the Function List.

extractFromNV21

Function Syntax

```
public native int extractFromNV21
(
    byte[] nv21Data,
    int width,
    int height,
    byte[] verTemplate,
    int[]exp,
    int expmode)
```

Description

Detects the face from NV21 image data and extracts a verification/identification template.

Parameters

Parameter	Description
nv21Data	In: NV21 image data
width	In: Image width
height	In: Image height
verTemplate	Out: Face verification/identification template (See extractFromBitmap.)
exp	Out: exposure value(If exp[0] !=0, you need set the exposure value to camera, ex:amtFaceCamera.setParameter(10107, ToolUtils.intToByteArray(exp[0]), 4))
expmode	In: 0 means register, 1 means verify or identify.

Returns

0	Success
-1	Face recognition failure caused by an error in image size conversion
-15	Template extraction failure
-1000	Dongle error
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

Click [here](#) to view the Function List.

extractFromGrayscaleData

Function Syntax

```
public native int extractFromGrayscaleData
(
    byte[] grayPixels,
    int width,
    int height,
    byte[] verTemplate,
    int[]exp,
    int expmode)
```

Description

Detects the face from 256-gray scale pixel data and extracts a verification/identification template.

Parameters

Parameter	Description
grayPixels	In: 256-gray scale pixel data
width	In: Image width
height	In: Image height
verTemplate	Out: Face verification/identification template (See extractFromBitmap)
exp	Out: exposure value(If exp[0] !=0, you need to set the exposure value to camera, ex:amtFaceCamera.setParameter(10107, ToolUtils.intToByteArray(exp[0], 4))
expmode	In: 0 means register, 1 means verify or identify.

Returns

0	Success
-1	Face recognition failure caused by an error in image size conversion
-15	Template extraction failure
-1000	Dongle error
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

Click [here](#) to view the Function List.

getTemplateQIt**Function Syntax**

```
public static int getTemplateQIt(byte[] verTemplate)
```

Description

Gets the quality of the template.

Parameters

Parameter	Description
verTemplate	In: Template data (The template must be a verification/identification template, not a registered template generated by mergeRegTemplate).

Returns

>=0	Quality score
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

The return value range is 0-255.

A template with a quality score below 50 is not recommended for face registration or for verification/identification.

Click [here](#) to view the Function List.

mergeRegTemplate**Function Syntax**


```
public static int mergeRegTemplate
(
    byte[] verTemplates,
    int count,
    byte[] regTemplate
)
```

Description

Merges the five verification/identification templates into a registered template.

Parameters

Parameter	Description
verTemplates	In: Bytes of the five merged verification/identification templates [5*AMTNIRFaceService12.FIX_VER_TEMPLATE_LEN].
count	In: Enter 5
regTemplate	Out: Registered template, for which caller needs to allocate memory. For the fixed length of the registered template, see AMTNIRFaceService12.FIX_REG_TEMPLATE_LEN.

Returns

0	Success
-5	Template merging failure
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

Click [here](#) to view the Function List.

verify

Function Syntax

```
public static int verify(byte[] regTemplate, byte[] verTemplate)
```

Description

Performs the face verification process.

Parameters

Parameter	Description
regTemplate	In: A registered template (See mergeRegTemplate).
verTemplate	In: A verification/identification template (See extractFromBitmap and other extract functions.)

Returns

>=0	Score
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

The score range is 0-1000.

The recommended threshold value is 575.

Click [here](#) to view the Function List.

dbAdd

Function Syntax

```
public static int dbAdd(String id, byte[] regTemplate)
```

Description

Adds the registered face template to the database.

Parameters

Parameter	Description
id	In: Face ID
regTemplate	In: A registered template (See mergeRegTemplate.)

Returns

0	Success
-1	Incorrect ID
-106	ID registered already
-200	Full database
-8	Memory allocation error
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

Click [here](#) to view the Function List.

dbDel**Function Syntax**

```
public static int dbDel(String id)
```

Description

Removes the face template of the specified ID from the database.

Parameters

Parameter	Description
id	In: Face ID

Returns

0	Success
-103	ID not found in the database
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks: Click [here](#) to view the Function List.

dbCount

Function Syntax

```
public static int dbCount()
```

Description

Gets the total number of face templates stored in the database.

Returns

>=0	Total number of face templates
-11002	Algorithm not initialized

Remarks

Click [here](#) to view the Function List.

dbClear

Function Syntax

```
public static int dbClear()
```

Description

Clears the database.

Returns

0	Success
-11002	Algorithm not initialized

Remarks: Click [here](#) to view the Function List.

dbIdentify

Function Syntax

```
public static int dbIdentify
(
    byte[] verTemplate,
    String[] id,
    boolean isAdapt,
    boolean[] bUpdated,
    byte[] updatedTempate
)
```

Description

Performs the face identification process.

Parameter

Parameter	Description
verTemplate	In: A verification/identification template (See extractFromBitmap and other extract functions.)
id	In: ID of the face recognized, for which the caller needs to allocate memory (exp:id = new String[1]).
isAdapt	In: Whether to learn the face features automatically
bUpdated	Out: Whether to update the registered template automatically, for which the caller needs to allocate memory (e.g. : bUpdated = new boolean[1])
updatedTempate	Out: When bUpdated[0] = true; updatedTempate returns the registered template with revised features, and the updated registered template can be added to the database. The memory is allocated by the caller. See mergeRegTemplate.

Returns

>=0	Score
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

The score range is 0-1000.

The recommended threshold value is 585.

The obtained template does not need to be added to the algorithm database (which will be automatically updated). And it is required to add the updated template to the application database.

Click [here](#) to view the Function List.

dbVerify

Function Syntax

```
public static int dbVerify
(
    byte[] verTemplate,
    String id,
    boolean isAdapt,
    boolean[] bUpdated,
    byte[] updatedTempate
)
```

Description

Performs the face verification process.

Parameter

Parameter	Description
verTemplate	In: A verification template (See extractFromBitmap and other extract functions.)
id	In: Face ID
isAdapt	In: Whether to learn the face features automatically
bUpdated	Out: Whether to update the registered template automatically, for which the caller needs to allocate memory (e.g. : bUpdated = new boolean[1])
updatedTempate	Out: When bUpdated[0] = true;

	<p>updatedTempate returns the registered template with revised features, and the updated registered template can be added to the database.</p> <p>The memory is allocated by the caller. See mergeRegTemplate.</p>
--	--

Returns

>=0	Score
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

The score range is 0-1000.

The recommended threshold value is 575.

The obtained template does not need to be added to the algorithm database (which will be automatically updated). And it is required to add the updated template to the application database.

Click [here](#) to view the Function List.

getFacePosition

Function Syntax

```
public static int getFacePosition(int[] facePosition, int[] eyesPosition)
```

Description

Gets the coordinates of key points on the face.

Parameter

Parameter	Description
facePosition	Out: Coordinates of key points on the face, for which the caller needs to allocate memory (e.g. :positions = new int[8])
eyesPosition	In: Coordinates of eyes, for which the caller needs to allocate memory (e.g. : eyesPosition = new int[4])

Returns

0	Success
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

facePosition[0]	X coordinate of the upper left corner of the face frame
facePosition[1]	Y coordinate of the upper left corner of the face frame
facePosition[2]	X coordinate of the upper right corner of the face frame
facePosition[3]	Y coordinate of the upper right corner of the face frame
facePosition[4]	X coordinate of the lower right corner of the face frame
facePosition[5]	Y coordinate of the lower right corner of the face frame
facePosition[6]	X coordinate of the lower left corner of the face frame
facePosition[7]	Y coordinate of the lower left corner of the face frame

eyesPosition[0]	X coordinate of the left eye
eyesPosition[1]	Y coordinate of the left eye
eyesPosition[2]	X coordinate of the right eye
eyesPosition[3]	Y coordinate of the right eye

Click [here](#) to view the Function List.

Appendix

Appendix 1: Glossary

The following definitions will help you understand basic functions of the near-infrared face recognition application and complete integrated development of the application.

Verification/Identification template

Verification/Identification templates uses either 1:1 or 1:N face verification/identification and merge the extracted templates into a registered template as the final face registration.

1:1 face Verification

1:1 face verification, is a process of verifying whether a user has a valid identity based on the user ID and face template or determining whether the registered template and the several verification templates extracted matches the same captured face image.

That is, 1:1 biometric verification process authenticates a person's identity by comparing the captured biometric template with a biometric template of that person pre-stored in the database.

1:N face Identification

1:N face identification, is a process of determining whether a user exists in the system based on the face of the user, without the user ID. Specifically, the application looks up the database of registered face templates based on the input face template and returns the name of the user by meeting the threshold of face similarity degree, and other related information.

So thus, A one-to-many (1:N) biometric identification process instantly compares the person's captured biometric template against ALL stored biometric templates in the system.

Registered / Registration template

A registered or registration template is the face template returned by the **mergeRegTemplate** function.

Registered face

The face recognition device captures five face images of the same user to extract verification/identification templates. Then merges the verified or identified templates into a registered template, and then loads it to the backend database as a registered face for subsequent face recognition.

190 Bluegrass Valley Pkwy,

Alpharetta, GA 30005, USA

E-mail: info@armatura.us

www.armatura.us



Copyright © 2022 ARMATURA LLC. All Rights Reserved.