TransAM Framework Face Recognition SDK

TransAM Framework Cognitive is a Microsoft face recognition framework based on the Softbank Pepper robot supported by NaoQi 2.9.5 and above. TransAM Framework Cognitive provides business-oriented face recognition interface, encapsulates the details of the Microsoft SDK logic, allowing users to focus on the development. It is recommended that beginner developers can get more priority to the use of TransAM Framework Cognitive to implement the face recognition business, if there are more advanced functionality customization needs can use Microsoft SDK interface or the If you have more advanced customization needs, you can use Microsoft SDK interface or TransAM Framework Cognitive API implementation.

Introduction to Microsoft Face Recognition

Microsoft Face Recognition Service provides the ability to do face detection, face recognition and face analysis through images, which can be widely applied to financial, pan-security, retail and other industry scenarios to meet the business needs of identity verification, face attendance and so on. However, it requires developers to create their own Microsoft Azure face recognition service account and the corresponding basic information needed to use the face service. It provides an interface to limit the frequency of invocation for testing purposes, and requires developers to pay for the cost if it is applied to product development.

Using Face Recognition on Pepper

Currently, the Pepper robot supports face recognition in NAOqi version 2.9.3 and above, which provides developers with a solution to implement face recognition on Pepper, and developers only need to load the SDK to integrate the face recognition function. The current SDK uses Microsoft Azure Face Recognition Service.

Features

- Video preview
- Face registration and capture
- Face detection and attribute analysis
- Face matching function
- Face Recognition

Steps to implement Pepper's face recognition function

- Create the Android Application project.
- Create the robot application. ReferenceLink
- Import TransAM Framework Cognitive to the project.
- Call TransAM Framework Cognitive SDK to implement the face recognition function.

Register and get Microsoft Face Services

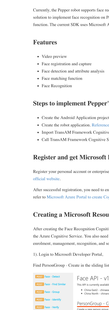
Register your personal account or enterprise account through Microsoft Azure official website, please refer to Microsoft Azure official website.

After successful registration, you need to enable Microsoft Cognitive Services, which includes face recognition services, please refer to Microsoft Azure Portal to create Cognitive Services resources.

Creating a Microsoft Resource Group

After creating the Face Recognition Cognitive Service, you have obtained a unique endpoint and a subscription_key to access the Azure Cognitive Service. You also need to create an Azure Resource Group, which is used to store face information for enrolment, management, recognition, and so on.

1). Login to Microsoft Developer Portal.



2). Based on the image above, select a region to create the group resource, for example, here select China East 2 as an example

Fill in the information according to the above figure. Note the default value of recognitionModel must be 'recognition01', if the creation is successful, it will return the success code 200.

Introducing dependencies

In the Android project, add the maven repository:

```
maven {
    url "http://maven.softbankrobotics.com.cn/release1"
}
```

Add the following dependency to the **build.gradle** file of your Android project:

```
implementation 'cn.softbankrobotics.transamcognitive:1.0.5'
```

Recognitive Framework Cognitive SDK usage examples

FaceService

The FaceService interface is used for video preview, face registration and capture, face detection and attribute analysis, face authentication and face recognition services.

Call description

```
FaceService faceService =
// Method to pass in a robot configuration instance,
new FaceServiceRobot(.setContext(qiContext)robotConnectionConfiguration)
// Method passes in a configuration instance for the Microsoft Face Recognition Service.
.setFaceServiceConfiguration(faceConfiguration)
// Turn on the video preview, passing in the Activity context and an ImageView instance.
.withPreview(mContext, imageView)
// Add a listener for the face recognition result.
.addListener(new FaceListenerImpl())
// Need the QiContext for the robot.
.build(qiContext);
```

Parameter Description

- Creates a RobotConnectionConfiguration configuration instance:

```
/**
* Configure the connection parameters for the robot:
* 1. 'account' is Pepper's robot username.
* 2. 'password' is Pepper's robot password.
*/
RobotConnectionConfiguration robotConnectionConfiguration = new RobotConnectionConfiguration("accou
```

- Create a Map to manage the authentication information for Microsoft Face Recognition. Because the SDK already provides a wrapper around the Microsoft Face Recognition implementation, the endpoint, subscription_key and resource group information for the Registered Face service is needed here.

```
/**
* Configure the authorization parameters for the Microsoft Face service:
* 1. 'endpoint' is the endpoint value provided by Microsoft.
* 2. 'subscription_key' Provides the subscription_key value for Microsoft.
* 3. 'face_group_name' for the group name created on the Microsoft Face Service engine.
*/

Map configuration = new LinkedHashMap<String, String>();
configuration.put(FaceConfiguration.ENDPOINT, "endpoint_value");
configuration.put(FaceConfiguration.AUTHENTICATION_KEY, "authentication_key_value"); configuration.
configuration.put(FaceConfiguration.GROUP_NAME, "face_group_name");
```

- FaceListener is an event listener for the FaceService service, including initialization information, registration information, face detection and attribute analysis, face authentication and face recognition results.

```
public class FaceListenerImpl implements FaceListener {}
```

Init

Initialize the FaceService service, mainly for loading the face service engine.

call description

By default, Microsoft Face Service is used, using the init() function without parameters.

```
faceService.init(); // java
```

The TransAM Framework Cognitive SPI implementation requires the init() function to be called with parameters. For SPI development, please refer to the 'Face Recognition SPI sections' document.

```
/**
* @param localSingleName The name of the local engine.
* @param remoteSingleName The name of the remote engine.
*/
faceService.init(localSingleName, remoteSingleName);
```

Return parameters

The onInitialized() function of the listener function will return if the initialization was successful.

```
// java public void onInitialized() {}

### createPerson
Create a personal information, it will be saved to the face recognition service engine.

### call description
```

java faceService.createPerson(String userName, String identifier);

```
#### Parameter description

| Parameter Name | Property | Description |
| ---------- | ---------- | ---------- |
| userName | String | The title of the user, this can be the user's real name, pronoun, etc |
| identifier | String | The user's unique identifier, it is the user's mobile phone number, 2

#### Return Parameters
The onPersonCreated() function will return if the creation of the Identity was successful.

java void onPersonCreated(Person person) {}

Information about the Person class includes:

| parameter name | property | description |
| ---------- | ---------- | ---------- |
| userName | String | The title of the user, this can be the user's real name, pronoun, etc |
| identifier | String | The identifier of the user, it is the user's mobile phone number |
| personId | String | The user's personal ID, the unique identifier generated by the face recognitio

#### deletePerson
Deletes a person's personal information from the Face Recognition Service Engine.
#### Calling Instructions

java faceService.deletePerson(String personId);

#### Parameter Description

| Parameter Name | Attribute | Description |
| ---------- | ---------- | ---------- |
| personId | String | Attribute of the Person class, a unique identifier generated by the Face Reco

#### Return Parameters
The listener function onPersonDeleted() will return if the deletion of the personal information is

java void onPersonDeleted(Person person) {}

The information of the Person class includes:

| parameter name | property | description |
| ---------- | ---------- | ---------- |
| userName | String | The title of the user, this can be the user's real name, pronoun, etc |
| identifier | String | The identifier of the user, this can be the user's mobile phone numb
| personId | String | P the user's personal ID, the unique identifier generated by the face recogni

#### registerFace
Registers the face information to the Microsoft Face Recognition Service and binds the personId int

#### call description

java faceService.registerFace(String personId);

#### Parameter Description

| Parameter Name | Property | Description |
| ---------- | ---------- | ---------- |
| personId | String | Attribute of the Person class, a unique identifier generated by the face reco

#### Return Parameters
The listener function onFaceRegistered will return if the registration with Microsoft Face Recognit

java public void onFaceRegistered(Person person, String faceId) {}

| Parameter Name | Properties | Description |
| ---------- | ---------- | ---------- |
| person | Person | Returns registered information, including username, identifier, personId |
| faceId | String | has a face ID that was successfully returned by the Microsoft Face Recognition

#### detect
Face detection and attribute information returned.

#### call description

java faceService.detect();

#### Return parameters
The listener function onFaceRegistered returns if registration with the Microsoft Face Recognition

java public void onFaceDetectedWithFaceAttributesResult(FaceDetectResult) {}

The FaceDetectResult class contains information about face attributes.

| ParameterName | Property | Description |
| ---------- | ---------- | ---------- |
| faceNumber | int | Number of faces identified |
| faceRectangles | RectangleEl | Positions of rectangles of recognized faces |

#### verify
Face authentication.

#### call description

java faceService.verify(String identifier, InputStream compareImage); // java

#### Parameter description

| Parameter Name | Attribute | Description |
| ---------- | ---------- | ---------- |
| identifier | String | The unique identifier of the user, this can be the user's mobile phone numb
| compareImage | InputStream | need to compare the user image, can be a local image or a photo |

#### Return Parameters
Get the confidence level of the face matching, the listener function onVerified will return.

java public void onVerified(String identifier, double confidence) {}

| Parameter Name | Attribute | Description |
| ---------- | ---------- | ---------- |
| identifier | String | The unique identifier of the user, this can be the user's mobile phone numb
| confidence | double | Returns the confidence level of the face authentication |

#### recognize
Recognize the face.

#### call description

java faceService.recognize();

#### Return parameters
The listener function onFaceRecognized will return if the face recognition was successful.

java public void onFaceRecognized(Person person) {}

| Parameter Name | Attribute | Description |
| ---------- | ---------- | ---------- |
| person | Person | Returns the registered information, including username, identifier, personId |

#### Stop
Stop the FaceService face recognition service.
#### Call description

java faceService.stop();
```