C1000 SDK Instruction

Catalogue

Catalogue 1

1.Release Notes 1

2. Project configuration 1

3. Permission list 2

4. Initialize face recognition interface 3

5. Face detection interface 3

6. Face feature extraction interface 4

7. Face comparison interface 4

8.Close the face recognition engine interface 5

9. Obtain temperature interface 5

10. Turn on the fill light 6

11. Turn off the fill light 6

1.Release Notes

|  |  |  |
| --- | --- | --- |
| **Version** | **Explanation** | **Date** |
| 1.0 | Basic functions of face recognition | 2020-03-19 |
|  |  |  |
|  |  |  |
|  |  |  |

2. Project configuration

Import aar file FacerTempSdk \* .aar and configure project dependencies (Face Recognition Temperature Detection SDK). E.g.

implementation (name: ' FACE\_DETECT\_release\_05261415', ext: 'aar')

3. Permission list

Including file read and write permissions and camera call permissions

<uses-permission android:name="android.permission.CAMERA"/>  
<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
  
<uses-permission android:name="android.permission.INTERNET"></uses-permission>  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"></uses-permission>  
  
<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"></uses-permission>  
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"></uses-permission>  
  
<uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED" />  
<uses-permission android:name="android.permission.WAKE\_LOCK" />  
  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE"></uses-permission>  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"></uses-permission>  
<uses-permission android:name="android.permission.REQUEST\_INSTALL\_PACKAGES"></uses-permission>  
<uses-permission android:name="android.permission.READ\_CALENDAR"></uses-permission>

4. Initialize face recognition interface

Use SDKFaceManager getInstance () to initialize；

|  |  |  |
| --- | --- | --- |
| API Function | | public boolean init(Context context) |
| Function Description | | Initialize the face recognition system |
| Parameter Description | Import | \* @param context Context |
| Export | True: Initialization succeeded False: Initialization failed |

* **Example：**
* SDKManager.getInstance().init(DemoActivity.this);

5. Face detection interface

|  |  |  |
| --- | --- | --- |
| API function | | public Face[] detectFace(byte[] nv21Data,int PREVIEW\_WIDTH,int PREVIEW\_HEIGHT, Constants.TS\_ROTATION rot) |
| Function Description | | Input face preview data of NV21 to get face information |
| Parameter Description | Import | \* @param nv21Data nv21 Face preview data in format  \* @param PREVIEW\_WIDTH Preview data width  \* @param PREVIEW\_HEIGHT Preview data height  \* @param rot Picture rotation angle 0,90,180,270 |
| Export | Face [] is the detected face set. If the Face [] set is not null and the length is larger than 0, the face related information is obtained. |

* **Example:**
* Face[] faces = SDKFaceManager.*getInstance*().detectFace(nv21Data,width,
* height, Constants.TS\_ROTATION.*TS\_ROTATION\_90*);  
  if (faces!=null && faces.length>0){  
   // Face information obtained successfully  
  }

6. Face feature extraction interface

|  |  |  |
| --- | --- | --- |
| API function | | public byte[] extractFeature(byte[] nv21Data,boolean isBackCam, int cameraOrientation) |
| Function Description | | Extract facial feature information |
| Parameter Description | Import | \* @param nv21Data Face preview data in nv21 format  \* @param isBackCam Whether it is a rear camera  \* @param cameraOrientation Camera rotation angle |
| Export | Face [] is the set of detected faces. If the Face [] set is not null and the length is greater than 0, face-related information is obtained |

* **Example**

byte[] feature = SDKManager.*getInstance*().extractFeature(nv21Data,isBackCam,cameraOrientation);

7. Face comparison interface

|  |  |  |
| --- | --- | --- |
| API function | | public float compareFace(Face face1,Face face2) |
| Function Description | | Incoming 2 eigenvalue comparisons to get matching degree |
| Parameter Description | Import | Face1 is the first face feature value  Face2 is the second face feature value |
| Export | Float：compare results（0-1.0）  Returns -1: failed to get facial features |

* **Example**

try {

float score =

SDKFaceManager.getInstance().compareFace(face1,face2);

//score is similarity（0-1.0）  
 } catch (Exception e) {  
 e.printStackTrace();  
 }

8.Close the face recognition engine interface

|  |  |
| --- | --- |
| API Function | SDKManager.*getInstance*().release(); |
| Function Description | Close the face recognition engine interface |
| Parameter Description | None |

* **Example**

SDKManager.getInstance().release ();

# 9. Obtain temperature interface

|  |  |
| --- | --- |
| API Function | SDKManager.*getInstance*().receiveTemp(OnTempListener onTempListener); |
| Function Description | 获取设备检测温度 |
| Parameter Description | OnTempListener: callback for receiving temperature data |

* **Example**
* SDKManager.*getInstance*().receiveTemp(new OnTempListener() {  
   @Override  
   public void onDataReceive(final double temp) {  
   runOnUiThread(new Runnable() {  
   @Override  
   public void run() {  
   ToastTool.*show*(DemoActivity.this," Body temperature detected：" + temp);  
   }  
   });  
   }  
  });

# 10. Turn on the fill light

|  |  |
| --- | --- |
| API Function | SDKManager.*getInstance*().openLight(); |
| Function Description | Turn on the fill light |
| Parameter Description |  |

* **Example**

SDKManager.*getInstance*().openLight();

# 11. Turn off the fill light

|  |  |
| --- | --- |
| API Function | SDKManager.*getInstance*().closeLight(); |
| Function Description | Turn off the fill light |
| Parameter Description |  |

* **Example**

SDKManager.*getInstance*().closeLight();

# 12. Device port control

|  |  |
| --- | --- |
| API Function | SDKManager.*getInstance*().setGpioDirection(int gpio, int direction); |
| Function Description | Port control, where the device signal is controlled by the port number and port level, for example:  When the fill light lamp needs to be turned on: setGpioDirection(45,1)  When the fill light needs to be turned off: setGpioDirection(45,0) |
| Parameter Description | Gpio: port Numbers such as: 124 stands for ALARM 45 for fill light 25 and 41 for Wigan OUT  Pull direction level up or down |

* **Example**
* SDKManager.*getInstance*().setGpioDirection(45,1);