第 5 章

安全(案例 24：手机人脸识别)

本章通过 1 个案例讲解生物特征识别的基本开发方法和步骤，案例为手机人脸识别。

本案例通过 BiometricAuthentication 实现手机人脸识别功能。本案例需要在真机上调试运行，真机调试操作详情可参考官方文档。首先，创建 Java 模板空工程 FaceRecognition；接着，实现 UI 布局设计；然后，获取 BiometricAuthentication 对象并初始化，实现人脸识别功能；最后，实现取消人脸识别功能。

(1) 创建 Java 模板空工程 FaceRecognition。

(2) 在配置文件 config.json 中添加人脸识别权限，参考代码如下：

```
//第 5 章\FaceRecognition\...\main\config.json
"reqPermissions": [
  {
    "name": "ohos.permission.ACCESS_BIOMETRIC"
  }
]
```

(3) 人脸识别 UI 布局设计，参考布局如图 5-1 所示。

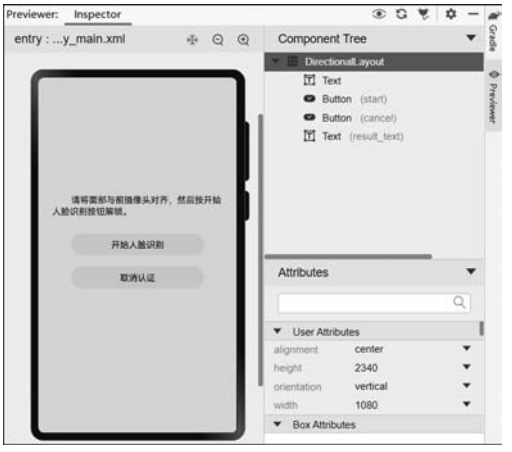

图 5-1　手机人脸识别 UI 布局

（4）手机人脸识别 UI 布局，参考代码如下：

```
//第 5 章\FaceRecognition\...\layout\ability_main.xml
<?xml version = "1.0" encoding = "UTF - 8"?>
< DirectionalLayout
    xmlns:ohos = "http://schemas.huawei.com/res/ohos"
    ohos:height = "match_parent"
    ohos:width = "match_parent"
    ohos:alignment = "center"
    ohos:orientation = "vertical"
    ohos:padding = "5vp">

    < Text
        ohos:height = "match_content"
        ohos:width = "match_content"
        ohos:margin = "20vp"
        ohos:multiple_lines = "true"
        ohos:text = "          请将面部与前摄像头对齐，然后按开始人脸识别按钮解锁."
        ohos:text_size = "16fp"/>

    < Button
        ohos:id = " $ + id:btn_start"
        ohos:height = "40vp"
        ohos:width = "240vp"
        ohos:background_element = " $graphic:button_bg"
        ohos:margin = "10vp"
        ohos:text = "开始人脸识别"
        ohos:text_size = "16fp"/>

    < Button
        ohos:id = " $ + id:btn_cancel"
        ohos:height = "40vp"
        ohos:width = "240vp"
        ohos:background_element = " $graphic:button_bg"
        ohos:margin = "10vp"
        ohos:text = "取消认证"
        ohos:text_size = "16fp"/>

    < Text
        ohos:id = " $ + id:t_result"
        ohos:height = "match_content"
        ohos:width = "match_content"
        ohos:margin = "20vp"
        ohos:multiple_lines = "true"
        ohos:text_size = "16fp"/>
</DirectionalLayout >
```

(5) 创建背景文件 button_bg.xml,参考代码如下：

```
//第 5 章\FaceRecognition\...\graphic\button_bg.xml
<?xml version = "1.0" encoding = "utf - 8"?>
< shape
    xmlns:ohos = "http://schemas.huawei.com/res/ohos"
    ohos:shape = "rectangle">

    < corners
        ohos:radius = "75"/>

    < solid
        ohos:color = "＃0d000000"/>
</shape >
```

(6) 在 MainAbilitySlice 中创建 showDialog 方法,实现弹窗显示人脸识别是否成功功能,参考代码如下：

```
//第 5 章\FaceRecognition\...\slice\MainAbilitySlice.java
private void showDialog(String message) {
    getUITaskDispatcher().asyncDispatch(new Runnable() {
        @Override
        public void run() {
            //创建 Toast 弹窗,设置文本
            new ToastDialog(MainAbilitySlice.this).setText(message).show();
        }
    });
}
```

(7) 在 MainAbilitySlice 中声明类成员常量与成员变量,参考代码如下：

```
//第 5 章\FaceRecognition\...\slice\MainAbilitySlice.java
public class MainAbilitySlice extends AbilitySlice {
    private static final int EVENT_MESSAGE_SUCCESS = 0x1000001;

    private static final int EVENT_MESSAGE_FAIL = 0x1000002;

    private static final int BA_CHECK_NOT_ENROLLED = 4;

    private Text t_result;

    private String result;

    //创建 EventHandler 对象
    private final EventHandler handler = new EventHandler(EventRunner.current()) {
```

```
        @Override
        protected void processEvent(InnerEvent event) {
            switch (event.eventId) {
                case EVENT_MESSAGE_SUCCESS:
                    showDialog(result);
                    t_result.setText(result);
                    break;
                case EVENT_MESSAGE_FAIL:
                    t_result.setText(result);
                    break;
            }
        }
    };

    private BiometricAuthentication biometricAuthentication;

    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_ability_main);
    }

    private void showDialog(String message) {
        getUITaskDispatcher().asyncDispatch(new Runnable() {
            @Override
            public void run() {
                //创建 Toast 弹窗,设置文本
                new ToastDialog(MainAbilitySlice.this).setText(message).show();
            }
        });
    }
}
```

(8) 创建 initComponents 方法,初始化 UI 组件,参考代码如下:

```
//第 5 章\FaceRecognition\...\slice\MainAbilitySlice.java
@Override
public void onStart(Intent intent) {
    super.onStart(intent);
    super.setUIContent(ResourceTable.Layout_ability_main);
    initComponents(); //初始化 UI 组件
}

private void initComponents() {
    t_result = (Text) findComponentById(ResourceTable.Id_t_result);
}
```

(9) 创建 execAuthentication 方法，在子线程中完成人脸识别功能，参考代码如下：

```
//第 5 章\FaceRecognition\...\slice\MainAbilitySlice.java
private void execAuthentication() {
    new Thread(() -> {
        //开始人脸识别
        int authenticationAction = biometricAuthentication.execAuthenticationAction(
BiometricAuthentication.AuthType.AUTH_TYPE_BIOMETRIC_FACE_ONLY,
BiometricAuthentication.SecureLevel.SECURE_LEVEL_S2, true, false, null);
        if (authenticationAction == 0) {
            //解锁成功
            result = "解锁成功";
            //给 EventHandler 对象发送人脸识别成功信息
            handler.sendEvent(EVENT_MESSAGE_SUCCESS);
        } else {
            result = "人脸识别失败，面部生物特征不匹配";
            //给 EventHandler 对象发送人脸识别失败信息
            handler.sendEvent(EVENT_MESSAGE_FAIL);
        }
    }).start();
}
```

(10) 创建 startFaceUnlock 方法，实例化 BiometricAuthentication 对象，检查设备是否有二维人脸识别功能，如果支持调用人脸识别方法，则进行人脸识别，参考代码如下：

```
//第 5 章\FaceRecognition\...\slice\MainAbilitySlice.java
private synchronized void startFaceUnlock() {
    try {
        //实例化 BiometricAuthentication 对象
        biometricAuthentication = BiometricAuthentication.getInstance(getAbility());
        //检查设备是否支持二维人脸识别
        int availability = biometricAuthentication.checkAuthenticationAvailability(
              BiometricAuthentication.AuthType.AUTH_TYPE_BIOMETRIC_FACE_ONLY,
              BiometricAuthentication.SecureLevel.SECURE_LEVEL_S2, true);
        if (availability == 0) {
            //设备支持二维人脸识别，开始人脸识别
            execAuthentication();
        } else if (availability == BA_CHECK_NOT_ENROLLED) {
            //设备未开启人脸识别功能
            result = "人脸识别功能未开启，请先开启人脸识别功能，并设置人脸图像信息";
        } else {
            //设备不支持二维人脸识别
            result = "此设备不支持二维人脸识别功能";
        }
```

```
        } catch (IllegalAccessException e) {
            result = "人脸识别错误";
        }
        //给 EventHandler 对象发送人脸识别失败信息
        handler.sendEvent(EVENT_MESSAGE_FAIL);
    }
```

(11) 给人脸识别按钮创建单击监听器,并调用 startFaceUnlock 方法,进行人脸识别,参考代码如下:

```
findComponentById(ResourceTable.Id_btn_start).setClickedListener(component -> {
startFaceUnlock();
});
```

(12) 创建 cancelRecognition 方法实现取消认证功能,参考代码如下:

```
//第 5 章\FaceRecognition\...\slice\MainAbilitySlice.java
private void cancelRecognition() {
    if (biometricAuthentication != null) {
        //取消人脸识别
        int resultCode = biometricAuthentication.cancelAuthenticationAction();
        if (resultCode == 0) {
            //取消成功
            t_result.setText("取消成功");
        } else {
            //取消失败,返回 resultCode 值
            t_result.setText("取消失败,返回值 = " + resultCode);
        }
    }
}
```

(13) 在取消认证按钮的单击监听器中调用 cancelRecognition 方法,取消人脸识别认证,参考代码如下:

```
findComponentById(ResourceTable.Id_btn_cancel).setClickedListener(component -> {
cancelRecognition();
});
```

(14) 登录 AppGallery Connect 官网,并创建项目“人脸识别”项目,如图 5-2 所示。

(15) 添加应用,如图 5-3 所示。

(16) 设置自动签名,如图 5-4 所示。

图 5-2　创建项目

图 5-3　添加应用

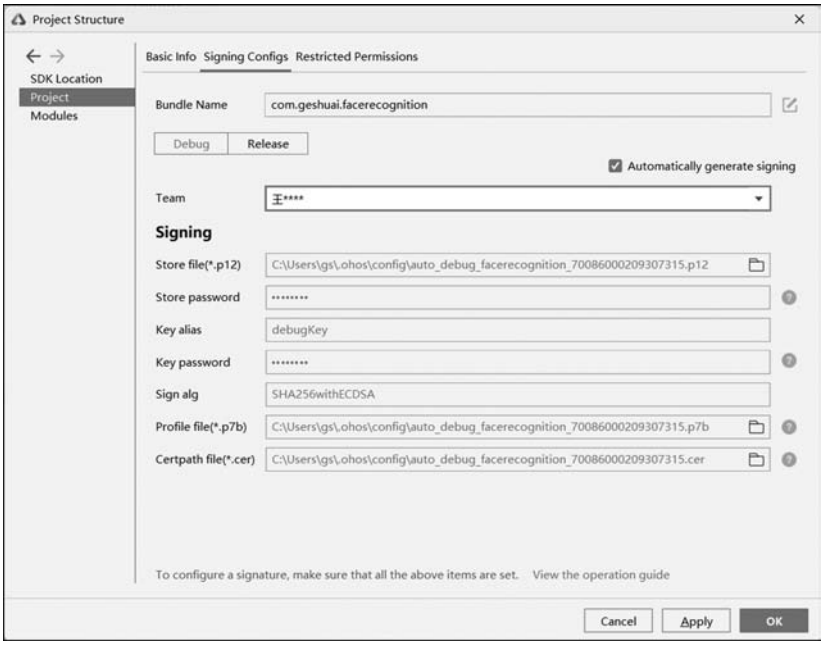

图 5-4　设置自动签名

（17）将程序运行到真机，运行效果如图 5-5 所示。

图 5-5　人脸识别运行效果