# uniapp 安卓商米人脸识别UTS版原生插件

## 目　录

# 使用方法

## 介绍

安卓商米人脸识别UTS版原生插件集成于商米人脸识别离线SDK包，包含人脸识别、人脸对比以及活体检测等功能，商米人脸识别SDK几乎全面适配配备摄像头的商米安卓设备，包括台式，自助，手持，称重等几乎所有畅销型号。

## 联系作者

关注微信公众号可联系作者

## 官方文档

https://developer.sunmi.com/docs/zh-CN/cdixeghjk491/drdeghjk524

## SDK版本

2.0.1

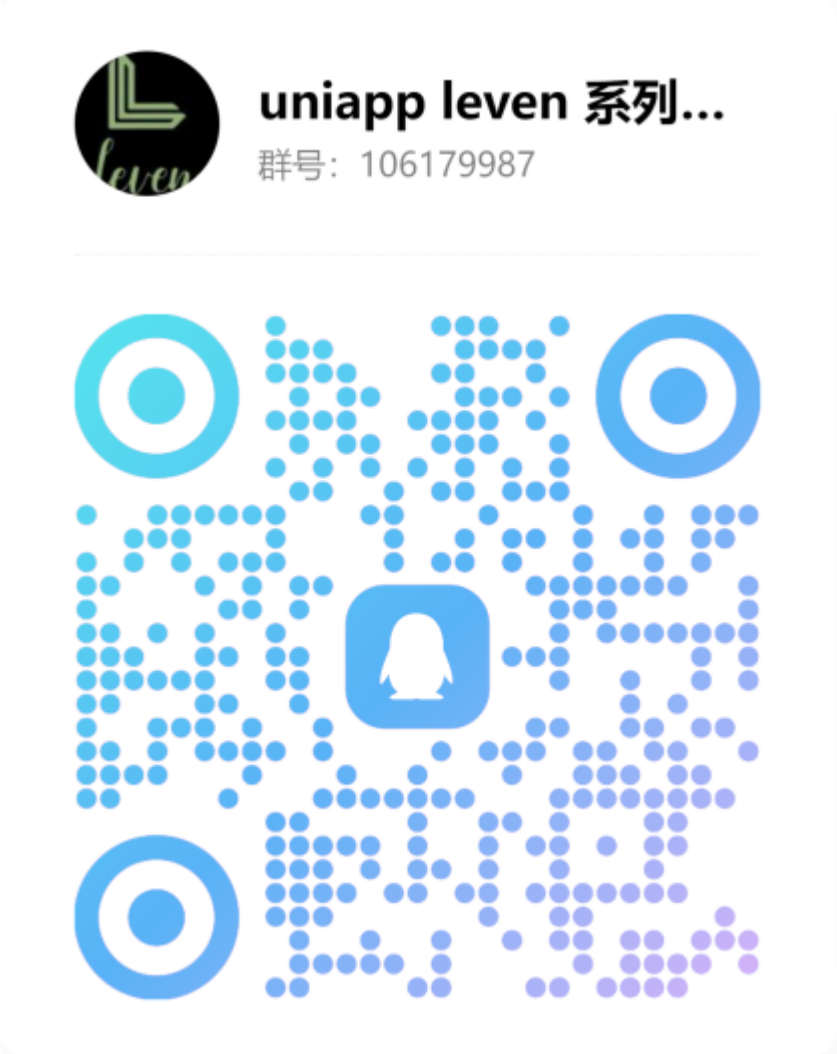
## 插件地址

https://ext.dcloud.net.cn/plugin?id=23857

## 演示程序下载

使用方法

加QQ群下载演示程序




# 权限

1. android.permission.READ_EXTERNAL_STORAGE
2. android.permission.READ_PHONE_STATE
3. android.permission.WRITE_EXTERNAL_STORAGE
4. android.permission.CAMERA

# API用法

> 仅提供了部分示例，完整示例请参考示例文件

- 用法

  在需要使用插件的页面加载以下代码

使用方法

```
import * as module from "@/uni_modules/leven-uts-sunmiFace"
```

- 示例

```
<template>
  <view>
    <uni-card title="商米人脸识别UTS原生插件DEMO">
      <view style="margin-bottom: 10px;">
        <uni-easyinput v-model="appId" placeholder="请输入appId"></uni-easyinpu
      </view>
      <view style="margin-bottom: 10px;">
        <button style="margin-bottom: 10px;" @click="init">激活</button>
        <button style="margin-bottom: 10px;" @click="getSdkVersion">sdk版本号</
      </view>
    </uni-card>
  </view>
</template>

<script>
  const module = uni.requireNativePlugin("leven-sunmiFace-FaceModule");
  // import * as module from "@/uni_modules/leven-uts-sunmiFace"
  export default {
    data() {
      return {
        // appId: "32e7c4bac4b04b7a9a86b4008bc7b2d7"
        appId: "a4f5808048964fb08753db44550649bb"
      }
    },
    onReady() {
      //申请插件权限
      this.requestPermissions();
    },
    methods: {
      requestPermissions() {
        module.requestPermissions(res => {
          console.log(res)
        })
      },
      //获取sdk版本号
      getSdkVersion() {
        module.getSdkVersion(res => {
          this.showToast(JSON.stringify(res))
        })
      },
      //激活
      init() {
        module.init({
          //商米申请的应用appId
          appId: this.appId,
```

```
            //是否开启调试模式
            debuggable: false,
            //是否强制刷新
            isForceRefresh: false,
            //自定义人脸存储目录
            customFaceDir: "/storage/emulated/0/levenSunmiFace"
          }, res => {
            try {
              console.log(JSON.stringify(res))
              if (res.code != 0) {
                throw new Error(res.message);
              }
              //激活成功
              uni.navigateTo({
                url: "/pages/index/index"
              })
            } catch (error) {
              //激活失败
              this.showToast(error.message)
            }
          })
        },
        showToast(title) {
          uni.showToast({
            icon: "none",
            title: title
          })
        }
      }
    }
</script>

<style>

</style>
```

# 人脸识别组件使用

- 用法

  在需要使用插件的页面添加以下代码

```
<leven-sunmiFace ref="refLevenSunmiFace" style="width: 400px; height: 400px; m
  @onError="onError" @onFaceResult="onFaceResult"></leven-sunmiFace>
```

- 示例

使用方法

```
  <template>
  <view class="face-compare">
    <leven-sunmiFace ref="refLevenSunmiFace" style="width: 400px; height: 400p
      @onError="onError" @onFaceResult="onFaceResult"></leven-sunmiFace>
    <!-- <view class="face-button" @click="register"><text style="color: #FFFF
    <view class="face-button" @click="openFace"><text style="color: #FFFFFF; f
    <view class="face-button" @click="closeFace"><text style="color: #FFFFFF;
  </view>
</template>

<script>
  export default {
    data() {
      return {
        //配置
        config: {
          //相机配置
          camera: {
            //视频旋转角度，可选值：0，90,180,270
            degree: 0,
            //组件圆角值
            radius: 200,
            //是否水平翻转
            isHorizontalFlip: false,
            //是否垂直翻转
            isVerticalFlip: false
          },
          //视频检测配置
          video: {
            //是否开启活体检测
            //0.不开启活体检测（默认）
            //1.开启RGB活体检测
            //2.开启NIR红外活体检测
            //4.开启3D结构光活体检测
            liveness: 1,
            //是否自动开启人脸识别，默认：false
            autoRecognition: false,
            //识别成功后延迟识别时间，单位：秒
            recognitionTime: 5
          }
        }
      }
    },
    methods: {
      //注册人脸，同时会关闭人脸识别，注册成功后需要手动打开人脸识别
      register() {
        this.$refs.refLevenSunmiFace.register({
          // 保存的id
          userId: "123",
```

```
            //保存的姓名
            userName: "leven2"
          }, res => {
            console.log(res)
          })
        },
        //开启人脸识别
        openFace() {
          this.$refs.refLevenSunmiFace.openFace(res => {
            console.log(res)
          })
        },
        //关闭人脸识别
        closeFace() {
          this.$refs.refLevenSunmiFace.closeFace(res => {
            console.log(res)
          })
        },
        //错误事件
        onError(e) {
          console.log("onError:" + JSON.stringify(e.detail))
        },
        //人脸识别结果
        onFaceResult(e) {
          console.log("onFaceResult:" + JSON.stringify(e.detail))
        }
      }
    }
</script>

<style>
  .face-compare {
    position: fixed;
    left: 0;
    right: 0;
    top: 0;
    bottom: 0;
    background-color: #FFFFFF;
    flex-direction: column;
    justify-content: center;
    align-items: center;
  }

  .face-button {
    width: 150px;
    height: 40px;
    background-color: #FF6000;
    flex-direction: row;
    justify-content: center;
    align-items: center;
```

使用方法

```
    border-radius: 20px;
    margin-bottom: 5px;
  }
</style>
```

# 状态码

## 状态码

| 状态码 | 描述 |
| --- | --- |
| 0 | 成功 |
| -1 | 通用错误码 |
| -10000 | 未初始化 |
| -10001 | 创建文件夹失败 |
| -10002 | 相机初始化失败 |
| -10003 | SDK初始化失败 |
| -10004 | 用户已存在 |
| -10005 | 提取人脸特征失败 |
| -10006 | 人脸被遮挡 |
| -10007 | 人脸图片质量较差 |
| -10008 | 光照条件较差 |
| -10009 | 人脸姿态较差 |
| -10010 | 文件不存在 |
| -10011 | 清空数据失败 |
| -10012 | url地址有误 |
| -10013 | 用户未注册 |
| -10014 | 没有检测到人脸 |
| -10015 | 未通过活体检测 |
| -10016 | 对比未通过 |
| -10017 | 人脸运动模糊 |

# 更新日志

## 2025-08-30 v1.2.6

- [优化] api人脸注册和批量注册新增参数 `是否保存人脸注册图片到本地`
- [优化] 优化人脸识别组件内存问题

## 2025-08-22 v1.2.4

- [优化] 激活接口新增参数 `默认的质量检测模式`

## 2025-08-21 v1.2.2

- [新增] 新增接口【设置默认的质量检测模式】

## 2025-06-15 v1.2.1

- [优化] 人脸检测属性新增参数 `isReturnRecognitionBase64` 识别完成后是否返回当前识别的图片的base64数据
- [优化] 人脸检测属性新增参数 `isSaveRecognitionImage` 是否保存识别成功的图片

## 2025-06-14 v1.2.0

- [新增] API新增方法【人脸对比】
- [新增] 组件新增方法【注册人脸】
- [新增] 组件新增方法【开启预览】
- [新增] 组件新增方法【关闭预览】

## 2025-06-14 v1.1.0

- [优化] 优化关闭人脸识别无效问题
- [优化] 激活接口新增部分参数
- [新增] 组件新增方法【获取当前识别参数】

# 2025-06-13

首次发布

# 申请插件所需权限

## 方法名

`requestPermissions`

## 用法

- 用法如下：

```
module.requestPermissions(res => {
  console.log(JSON.stringify(res))
});
```

- 参数说明

  无

## 回调

- 示例

```
{
  "data": {
    "grantedList": [
      "android.permission.READ_EXTERNAL_STORAGE",
      "android.permission.WRITE_EXTERNAL_STORAGE"
    ]
  },
  "message": "权限被拒绝",
  "code": -1
}
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |

申请插件所需权限

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| data.grantedList | Array | 申请的权限列表，**被拒绝的时候会返回** |
| code | Integer | 返回类型，0.成功，其他：失败 |

# 激活

## 方法名

`init`

## 用法

- 用法如下：

```
module.init({
      //商米申请的应用appId
      appId: this.appId,
      //是否开启调试模式
      debuggable: false,
      //是否强制刷新
      isForceRefresh: false,
      //自定义人脸存储目录
      customFaceDir: "/storage/emulated/0/levenSunmiFace",
      //人脸距离阈值，当SDK应用于支付场景，推荐值是0.9；当应用于人证比对场景时，推荐值为1.1
      distanceThreshold: 0.8,
      //人脸置信度阈值，低于此阈值的人脸信息将被丢弃（默认：0.7）
      faceScoreThreshold: 0.8,
      //最小人脸检测尺寸，较大较清晰的人脸识别较准确（默认：80）
      minFaceSize: 60,
      //使用cpu核心执行人脸检测个数，人脸识别（默认：2）
      threadNum: 3,
      //人脸框排序方式，0.按 人脸框置信度进行排序，1.按人脸框尺寸进行排序（默认），2.按人脸
      boxSortMode: 1,
      //RGB活体检测阈值，默认：0.95
      rgbLivenessThreshold: 0.9,
      //红外活体检测阈值，默认：0.1
      irLivenessThreshold: 0.1,
      //质量检测模式，0.不做质量检测，1.检测姿态，2.人脸遮挡检测，3.姿态检测和人脸遮挡（默认
      mode: 3
    }, res => {
      try {
        console.log(JSON.stringify(res))
      } catch (error) {
        //激活失败
        this.showToast(error.message)
      }
    })
```

激活

- 参数说明

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| appId | String | 是 | 无 | 商米申请的appId |
| debuggable | Boolean | 否 | false | 是否开启调试模式 |
| isForceRefresh | Boolean | 否 | false | 是否强制从服务器拉取token |
| customFaceDir | String | 否 | 空 | 自定义人脸存储目录 |
| distanceThreshold `v1.1.0` | float | 否 | 0.9 | 人脸距离阈值，当SDK应用于支付场景，推荐值是0.9；当应用于人证比对场景时，推荐值为1.1；阈值越小，SDK对于相同人的判定越严格（默认：0.9） |
| faceScoreThreshold `v1.1.0` | float | 否 | 0.7 | 人脸置信度阈值，低于此阈值的人脸信息将被丢弃（默认：0.7） |
| minFaceSize `v1.1.0` | Integer | 否 | 80 | 最小人脸检测尺寸，较大较清晰的人脸识别较准确（默认：80） |
| threadNum `v1.1.0` | Integer | 否 | 2 | 使用cpu核心执行人脸检测个数，人脸识别（默认：2） |

激活

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| boxSortMode `v1.1.0` | Integer | 否 | 1 | 人脸框排序方式，0.按 人脸框置信度进行排序，1.按人脸框尺寸进行排序（默认），2.按人脸框位置进行排序（优先选择居中位置） |
| rgbLivenessThreshold `v1.1.0` | float | 否 | 0.95 | RGB活体检测阈值，默认：0.95 |
| irLivenessThreshold `v1.1.0` | float | 否 | 0.1 | 红外活体检测阈值，默认：0.1 |
| mode `v1.2.4` | Integer | 否 | 3 | <//质量检测模式，0.不做质量检测，1.检测姿态，2.人脸遮挡检测，3.姿态检测和人脸遮挡（默认）> |

# 回调

- 示例


```
{
  "message": "",
  "data": {
    "sdkVersion": "v2.0.1@c9e84d8",
    "license": "WjdDbzdSVUF2TVF1Yk40eGVFMDR1UmhxbkJTczlwMzdUQTU4M1NUYUQvT3VPMk
  },
  "code": 0
}
```


- 回调说明：

激活

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| data.sdkVersion | String | 当前sdk版本号 |
| data.license | String | token许可 |
| code | Integer | 返回类型，0.成功，其他：请参考状态码 |

# 获取sdk版本号

## 方法名

`getSdkVersion`

## 用法

- 用法如下：

```
module.getSdkVersion(res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

  无

## 回调

- 示例

```
{
  "message": "",
  "data": {
    "version": "1.0.1"
  },
  "code": 0
}
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| data.version | String | 版本号 |
| code | Integer | 返回类型，0.成功，其他：请参考状态码 |

# 人脸注册

## 方法名

`imageRegister`

## 用法

- 用法如下：

```
module.imageRegister({
  //本地或网络url地址
  url: "/storage/emulated/0/DCIM/1234.jpg",
  // 保存的id
  userId: "456",
  //保存的姓名
  userName: "leven1",
  //备注
  userInfo: "leven1",
  //是否保存注册的图片
  isSaveImage: false
}, res => {
  console.log(res)
})
```

- 参数说明

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| url | String | 是 | 无 | 本地或网络url地址 |
| userId | String | 是 | 无 | 保存的id |
| userName | String | 是 | 无 | 保存的名称 |
| userInfo | String | 否 | 无 | 备注 |
| isSaveImage `v1.2.6` | Boolean | 否 | false | 是否保存注册的图片到本地 |

## 回调

- 示例

```
{
  "message": "",
  "data": {
    "userName": "leven1",
    "imagePath": "/storage/emulated/0/levenSunmiFace/faceImages/456@@leven1.jp
    "userId": "456",
    "userInfo": ""
  },
  "code": 0
}
```


- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| data.userId | String | 注册的用户id |
| data.userName | String | 注册的用户名字 |
| data.imagePath | String | 注册后本地保存的头像路径 |
| data.userInfo | String | 用户备注 |
| code | Integer | 返回类型，0.成功，其他：请参考状态码 |

# 批量注册

## 方法名

`batchRegister`

## 用法

- 用法如下：

```
module.batchRegister({
  //是否保存注册的图片
  isSaveImage: false
  list: [{
    //本地或网络url地址
    url: "/sdcard/DCIM/arcface/1.jpg",
    // 保存的id
    userId: 10001,
    //保存的姓名
    userName: "leven1",
    //用户备注
    userInfo: "leven1"
  }, {
    //本地或网络url地址
    url: "http://www.yeyuboke.com/svga/2.jpg",
    // 保存的id
    userId: 10002,
    //保存的姓名
    userName: "leven2",
    //用户备注
    userInfo: "leven2"
  }]
}, res => {
  console.log(res)
})
```

- 参数说明

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| isSaveImage `v1.2.6` | Boolean | 否 | false | 是否保存注册的图片到本地 |
| list | Array | 是 | 无 | 注册列表 |

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| list.url | String | 是 | 无 | 注册的图片本地或网络地址，当本地地址和网络地址共存时优先注册本地地址 |
| list.userId | String | 是 | 无 | 注册时保存的id |
| list.userName | String | 是 | 无 | 注册时保存的名称 |
| list.userInfo | String | 否 | 无 | 用户备注 |

# 回调

- 示例


```
{
  "status": "registerProgress",
  "failed": 3,
  "user": {
    "userName": "涂琳",
    "url": "http://download.iyouxin.com/upload/06c-43870d68-e7cb-4e1a-b915-3e2
    "imagePath": "/storage/emulated/0/levenSunmiFace/faceImages/a5782af7653f22
    "userId": "a5782af7653f223b0124343046a6888b",
    "userInfo": ""
  },
  "progress": 0.0124223605,
  "count": 1449,
  "success": 15
}
```


- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| data.url | String | 当前注册的url |

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| data.status | String | 当前注册的状态，<br>registerStart：开始注册，<br>registerProgress：注册中，<br>registerComplete：注册完成 |
| data.user | Object | 注册的用户信息 |
| data.user.userName | String | 保存的名称 |
| data.user.userId | String | 保存的id |
| data.user.url | String | 注册地址 |
| data.user.imagePath | String | 当前注册的人脸图片路径 |
| data.user.userInfo | String | 用户备注 |
| data.failed | Integer | 当前注册失败的数量 |
| data.progress | Float | 当前注册的进度 |
| data.count | Integer | 注册的所有数量 |
| data.success | Integer | 当前已成功注册的数量 |
| code | Integer | 返回类型，0.成功，其他：请参考状态码 |

# 人脸注册总数

## 方法名

`getFaceCount`

## 用法

- 用法如下：

```
module.getFaceCount(res => {
  console.log(res)
})
```

- 参数说明

  无

## 回调

- 示例

```
{
    "message": "",
    "code": 0,
    "data": {
        "count": 6
    }
}
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| data.count | Integer | 人脸数量 |
| code | Integer | 返回类型，0.成功，其他：请参考状态码 |

# 清空人脸

## 方法名

`clearFace`

## 用法

- 用法如下：

```
module.clearFace(res => {
  console.log(res)
})
```

- 参数说明

  无

## 回调

- 示例

```
{
    "message": "",
    "code": 0,
    "data": {}
}
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| code | Integer | 返回类型，0.成功，其他：请参考状态码 |

# 获取人脸

## 方法名

`getFace`

## 用法

- 用法如下：

```
module.getFace({
  userId: "123"
}, res => {
  console.log(res)
})
```

- 参数说明

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| userId | String | 是 | 无 | 保存的用户id |

## 回调

- 示例

```
{
    "data": {
        "userName": "leven1",
        "imagePath": "/storage/emulated/0/levenSunmiFace/faceImages/456@@leven
        "userId": "456",
        "userInfo": ""
    },
    "message": "",
    "code": 0
}
```

- 回调说明：

获取人脸

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| data.userId | String | 注册的用户id |
| data.userName | String | 注册的用户名字 |
| data.imagePath | String | 注册后本地保存的头像路径 |
| data.userInfo | String | 用户备注 |
| code | Integer | 返回类型，0.成功，其他：请参考状态码 |

# 删除人脸

## 方法名

`deleteFace`

## 用法

- 用法如下：

```
module.deleteFace({
  userId: "123"
}, res => {
  console.log(res)
})
```

- 参数说明

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| userId | String | 是 | 无 | 保存的用户id |

## 回调

- 示例

```
{
    "message": "",
    "code": 0,
    "data": {}
}
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| code | Integer | 返回类型，0.成功，其他：请参考状态码 |

# 获取所有人脸

## 方法名

`getAllFace`

## 用法

- 用法如下：

```
module.getAllFace(res => {
  console.log(res)
})
```

- 参数说明

  无

## 回调

- 示例


```
{
    "data": {
        "list": [
            {
                "userId": "123",
                "userName": "leven",
                "faceId": 3,
                "imagePath": "/storage/emulated/0/Android/data/test.leven.unip
                "userInfo": ""
            }
        ]
    },
    "message": "",
    "code": 0
}
```


- 回调说明：

获取所有人脸

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| data.list | Array | 人脸列表 |
| data.list.userId | String | 注册的用户id |
| data.list.userName | String | 注册的用户名字 |
| data.list.imagePath | String | 注册后本地保存的头像路径 |
| data.list.userInfo | String | 用户备注 |
| code | Integer | 返回类型，0.成功，其他：请参考状态码 |

# 人脸对比 v1.2.0

## 方法名

`faceCompare`

## 用法

- 用法如下：

```
module.faceCompare({
  //本地或网络url地址
  url: "http://www.yeyuboke.com/svga/8.jpg"
}, res => {
  // this.showToast(JSON.stringify(res))
  console.log(res);
})
```

- 参数说明

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| url | String | 是 | 无 | 本地或网络图片路径 |

## 回调

- 示例

```
{
    "data": {
        "userName": "leven1",
        "imagePath": "/storage/emulated/0/levenSunmiFace/faceImages/456@@leven
        "userId": "456",
        "userInfo": ""
     },
    "message": "",
    "code": 0
}
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| data.userId | String | 注册的用户id |
| data.userName | String | 注册的用户名字 |
| data.imagePath | String | 注册后本地保存的头像路径 |
| data.userInfo | String | 用户备注 |
| code | Integer | 返回类型，0.成功，其他：请参考状态码 |

# 设置默认质量检测模式 v1.2.2

## 方法名

`setFaceQualityMode`

## 用法

- 用法如下：

```
module.setFaceQualityMode({
  mode: 3
}, res => {
  console.log(res)
})
```

- 参数说明

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| mode | Integer | 否 | 3 | 质量检测模式，0.不做质量检测，1.检测姿态，2.人脸遮挡检测，3.姿态检测和人脸遮挡（默认） |

## 回调

- 示例

```
{
    "message": "",
    "code": 0,
    "data": {}
}
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| code | Integer | 返回类型，0.成功，其他：请参考状态码 |

# 用法

## 组件名称

`leven-uts-arc-face5`

> 仅支持在nvue下使用

## 用法

```
<leven-sunmiFace ref="refLevenSunmiFace" style="width: 400px; height: 400px; marg
      @onError="onError" @onFaceResult="onFaceResult"></leven-sunmiFace>
```

> 组件中具体的方法、属性以及事件请参考各自文档

# 属性

## config

> 组件初始化配置

## 参数说明

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| camera | Object | 否 | 无 | 相机属性，所有的参数都可以不传，不传则按默认的 |
| camera.degree | Integer | 否 | 0 | 相机旋转角度，可选值：0，90,180,270 |
| camera.radius | Integer | 否 | 0 | 摄像机预览圆角 |
| camera.isHorizontalFlip | Boolean | 否 | false | 是否水平翻转 |
| camera.isVerticalFlip | Boolean | 否 | false | 是否垂直翻转 |
| video | Object | 否 | 无 | 频检测配置，所有参数都可以不传，不传则按默认的 |
| video.liveness | Integer | 否 | 0 | 是否开启活体检测，0.不开启活体检测（默认），1.开启RGB活体检测，2.开启NIR红外活体检测，4.开启3D结构光活体检测 |
| video.autoRecognition | Boolean | 否 | false | 是否自动开启人脸识别 |
| video.recognitionTime | Integer | 否 | 5 | 识别成功后延迟识别时间，单位：秒 |

属性

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| video.sunmiFaceQualityMode `v1.2.1` | Integer | 否 | 3 | 质量检测模式，0.不做质量检测，1.检测姿态，2.人脸遮挡检测，3.姿态检测和人脸遮挡（默认） |
| video.isReturnRecognitionBase64 `v1.2.1` | boolean | 否 | false | 识别完成后是否返回当前识别的图片的base64数据 |
| video.isSaveRecognitionImage `v1.2.1` | boolean | 否 | false | 是否保存识别成功的图片 |

# 方法

# 人证对比

## 方法名

`idCardCompare`

> 对比结果在“人证对比结果”事件中回调，人证对比时会自动关闭人脸识别功能，如果需要对比后人脸识别需要手动调用开启人脸识别的方法

## 用法

- 用法如下：

```
if (this.$refs.refLevenSunmiFace) {
  this.$refs.refLevenSunmiFace.idCardCompare({
    // 图片地址支持网络地址
    url: "/storage/emulated/0/DCIM/1234.jpg"
  }, res => {
    console.log(res)
  });
}
```

- 参数说明

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| url | String | 是 | 时间戳 | 图片地址支持网络地址 |

## 回调

- 示例

```
{
    "data": {},
    "message": "",
    "code": 0
}
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| code | Integer | 返回类型，0.成功，其他：失败 |

# 开启人脸识别

## 方法名

`openFace`

## 用法

- 用法如下：

```
if (this.$refs.refLevenSunmiFace) {
  this.$refs.refLevenSunmiFace.openFace(res => {
    console.log(res)
  });
}
```

- 参数说明

  无

## 回调

- 示例

```
{
    "message": "",
    "code": 0,
    "data": {}
}
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| code | Integer | 返回类型，0.成功，其他：失败 |

# 关闭人脸识别

## 方法名

`closeFace`

## 用法

- 用法如下：

```
if (this.$refs.refLevenSunmiFace) {
  this.$refs.refLevenSunmiFace.closeFace(res => {
    console.log(res)
  });
}
```

- 参数说明

  无

## 回调

- 示例

```
{
    "message": "",
    "code": 0,
    "data": {}
}
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| code | Integer | 返回类型，0.成功，其他：失败 |

# 获取当前识别参数 v1.1.0

## 方法名

`getParams` `v1.1.0`

## 用法

- 用法如下：

```
if (this.$refs.refLevenSunmiFace) {
  this.$refs.refLevenSunmiFace.getParams(res => {
    console.log(res)
  });
}
```

- 参数说明

  无

## 回调

- 示例

```
{
    "message": "",
    "data": {
        "distanceThreshold": 0.800000011920929,
        "depthYOffset": 0,
        "boxSortMode": 1,
        "minFaceSize": 60,
        "depthXOffset": 0,
        "threadNum": 3,
        "faceScoreThreshold": 0.800000011920929
    },
    "code": 0
}
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| data.distanceThreshold | float | 人脸距离阈值 |
| data.depthYOffset | float | 深度图相对于彩色图y方向偏移 |
| data.boxSortMode | float | 人脸框排序方式 |
| data.minFaceSize | float | 最小人脸检测尺寸 |
| data.depthXOffset | float | 深度图相对于彩色图x方向偏移 |
| data.threadNum | float | 使用cpu核心执行人脸检测个数 |
| data.faceScoreThreshold | float | 人脸置信度阈值 |
| code | Integer | 返回类型，0.成功，其他：失败 |

# 注册人脸 v1.2.0

## 方法名

`register`

> 注册人脸会自动关闭人脸识别功能，如果需要对比后人脸识别需要手动调用开启人脸识别的方法

## 用法

- 用法如下：

```
if (this.$refs.refLevenArcFacePro) {
  this.$refs.refLevenArcFacePro.register({
    // 注册后保存的id
    userId: "123",
    //注册后保存的名字
    userName: "leven",
    //用户备注
    userInfo: "",
  }, res => {
    console.log(res)
  });
}
```

- 参数说明

| 参数名 | 参数类型 | 是否必填 | 默认值 | 参数描述 |
| --- | --- | --- | --- | --- |
| userId | String | 是 | 无 | 注册后保存的id |
| userName | String | 是 | 无 | 注册后保存的名字 |
| userInfo | String | 否 | 无 | 用户备注 |

## 回调

- 示例

```
{
  "message": "",
```

```
  "data": {
    "userName": "leven1",
    "imagePath": "/storage/emulated/0/levenSunmiFace/faceImages/456@@leven1.jp
    "userId": "456",
    "userInfo": ""
  },
  "code": 0
}
```


- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| data.userId | String | 注册的用户id |
| data.userName | String | 注册的用户名字 |
| data.imagePath | String | 注册后本地保存的头像路径 |
| data.userInfo | String | 用户备注 |
| code | Integer | 返回类型，0.成功，其他：失败 |

# 开启预览 v.1.2.0

> 开启预览会自动关闭人脸识别功能，如果需要对比后人脸识别需要手动调用开启人脸识别的方法

- 用法如下：

```
```

- 参数说明

  无

- 示例

```
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| code | Integer | 返回类型，0.成功，其他：失败 |

# 关闭预览 v1.2.0

## 方法名

`closePreview`

> 关闭预览会自动关闭人脸识别功能，如果需要对比后人脸识别需要手动调用开启人脸识别的方法

## 用法

- 用法如下：

```
if (this.$refs.refLevenSunmiFace) {
  this.$refs.refLevenSunmiFace.closePreview(res => {
    console.log(res)
  });
}
```

- 参数说明

  无

## 回调

- 示例

```
{
    "message": "",
    "code": 0,
    "data": {}
}
```

- 回调说明：

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| code | Integer | 返回类型，0.成功，其他：失败 |

# 事件

# 错误事件

## 介绍

onError

## 用法

```
<leven-sunmiFace ref="refLevenSunmiFace" @onError="onError">
</leven-sunmiFace>
```

## 回调示例

```
onError(e) {
    console.log(e.detail)
  }
```

## 回调内容

```
{
  "data": {},
  "message": "",
  "code": 0
}
```

## 回调说明

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| code | Integer | 返回类型，0.成功，其他：失败 |

# 人证对比结果

## 介绍

`onIdCardCompare`

## 用法

```
<leven-sunmiFace ref="refLevenSunmiFace" style="width: 400px; height: 400px; marg
```

## 回调示例

```
onIdCardCompare(e) {
    console.log(e.detail)
  }
```

## 回调内容

```
{
  "message": "对比通过",
  "data": {
    "type": "onIdCardCompare",
    "data": {}
  },
  "code": 0
}
```

## 回调说明

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| code | Integer | 返回类型，0.成功，其他：失败 |

# 人脸识别结果

## 介绍

`onFaceResult`

## 用法

```
<leven-sunmiFace ref="refLevenSunmiFace" style="width: 400px; height: 400px; marg
```

## 回调示例

```
onFaceResult(e) {
    console.log(e.detail)
  }
```

## 回调内容

```
{
  "message": "识别成功",
  "data": {
    "type": "onFaceResult",
    "data": {
      "userName": "leven1",
      "imagePath": "/storage/emulated/0/levenSunmiFace/faceImages/456@@leven1.jpg
      "userId": "456",
      "userInfo": ""
    }
  },
  "code": 0
}
```

## 回调说明

人脸识别结果

| 参数名 | 参数类型 | 参数描述 |
| --- | --- | --- |
| message | String | 消息提示 |
| data | Object | 数据对象 |
| data.userId | String | 用户id |
| data.userName | String | 用户名字 |
| data.imagePath | String | 保存的头像路径 |
| data.userInfo | String | 用户备注 |
| data.userInfo | String | 用户备注 |
| data.compareImagePath `v1.2.1` | String | 识别成功保存的图片 |
| data.compareImageBase64 `v1.2.1` | String | 当前识别的人脸图片base64数据 |
| code | Integer | 返回类型，0.成功，其他：失败 |