uniapp 安卓人脸识别UTS原生插件

目　录

使用方法

介绍

人脸识别UTS原生插件无第三方SDK，无需第三方设备激活码即可直接使用的人脸识别插件，插件支持图片人脸注册，相机人脸识别，批量注册（支持网络图片）等，支持开启和关闭识别功能等，支持保存用户的id和名称

联系作者

关注微信公众号可联系作者

插件使用注意事项

1. 个人照片可以是个人证件照，生活照，肖像照片，要求五官清晰且正脸的照片，不能出现多个人脸的情况
2. 尽量不要出现过度美颜，头发遮挡，低头侧脸等问题
3. 使用过程中，需要用户配合，正对着摄像头，以提高人脸识别准确率的效果
4. 使用插件要先调用初始化

插件地址

权限

1. android.permission.READ_EXTERNAL_STORAGE
2. android.permission.WRITE_EXTERNAL_STORAGE
3. 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

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

- 示例

```
<template>
  <view>
    <uni-card title="安卓人脸识别原生插件">
      <button type="primary" @click="requestPermissions">申请插件所需权限</button
      <button type="primary" @click="init">初始化</button>
      <button type="primary" @click="imageFaceRegister">图片注册人脸</button>
      <button type="primary" @click="clearFace">清空人脸库</button>
      <button type="primary" @click="deleteFace">删除人脸</button>
      <button type="primary" @click="getFace">获取人脸</button>
      <button type="primary" @click="getFaceCount">获取注册的人脸数量</button>
      <button type="primary" @click="getAllFace">获取所有人脸信息</button>
      <button type="primary" @click="faceCompare">人脸对比</button>
      <button type="primary" @click="batchRegister">批量注册</button>
    </uni-card>

    <!-- 注册进度弹窗 -->
    <process ref="refProcess" :processTitle="batchRegisterProcess.processTitle
      :successCount="batchRegisterProcess.successCount" :failedCount="batchReg
    </process>
  </view>
</template>

<script>
  // 批量注册人脸的文件
  import {
    faceList
  } from "@/utils/face.js"
  //加载组件
  import process from "./process.nvue"
  import * as module from "@/uni_modules/leven-uts-face"
  export default {
    components: {
      process
    },
    data() {
      return {
        //批量注册进度
```

```
        batchRegisterProcess: {
          //进度描述
          processTitle: "0",
          //当前进度
          process: 0,
          // 成功数量
          successCount: 0,
          //失败数量
          failedCount: 0,
          //注册总数量
          totalCount: 0
        }
      }
    },
    methods: {
      //申请插件所需权限
      requestPermissions() {
        module.requestPermissions(res => {
          console.log(res)
        });
      },
      //初始化
      init() {
        module.init({
          //开启线程数
          numThread: 1,
          //设置人脸检测置信度阈值，默认：0.5
          detConfThresh: 0.8,
          //设置人脸检测IOU阈值，默认：0.5
          detIouThresh: 0.8,
          //设置人脸识别相似度阈值，默认：0.8
          recConfThresh: 0.95,
          //自定义人脸存储路径
          customFaceDir: "/storage/emulated/0/levenFace"
        }, res => {
          console.log(res)
        });
      },
      // 图片注册人脸
      imageFaceRegister() {
        module.imageFaceRegister({
          //本地或网络url地址
          url: "/storage/emulated/0/DCIM/Camera/IMG_20241013_150358.jpg",
          // url: "http://www.yeyuboke.com/uniplugin/banner/beauty1.jpg",
          // url: "http://www.yeyuboke.com/svga/7.jpg",
          // 保存的id
          id: "111",
          //保存的姓名
          name: "leven1",
          // 同一人是否可以多次注册，默认true
```

```
        registerMultiple: false
      }, res => {
        // this.showToast(JSON.stringify(res))
        console.log(res);
      })
    },
    // 清空人脸库
    clearFace() {
      module.clearFace(res => {
        console.log(res);
      })
    },
    // 删除人脸
    deleteFace() {
      module.deleteFace({
        id: "111"
      }, res => {
        // this.showToast(JSON.stringify(res))
        console.log(res);
      })
    },
    // 获取人脸
    getFace() {
      module.getFace({
        id: "111"
      }, res => {
        // this.showToast(JSON.stringify(res))
        console.log(res);
      })
    },
    // 获取注册的人脸数量
    getFaceCount() {
      module.getFaceCount(res => {
        // this.showToast(JSON.stringify(res))
        console.log(res);
      })
    },
    // 获取所有人脸信息
    getAllFace() {
      module.getAllFace(res => {
        // this.showToast(JSON.stringify(res))
        console.log(res);
      })
    },
    faceCompare() {
      module.faceCompare({
        //本地或网络url地址
        url: "/storage/emulated/0/DCIM/Camera/IMG_20241013_150358.jpg",
        // url: "http://www.yeyuboke.com/uniplugin/banner/beauty1.jpg",
        //相似度，大于或等于该相似度的人脸视为通过，默认：0.85
```

```
            similar: 0.85
          }, res => {
            // this.showToast(JSON.stringify(res))
            console.log(res);
          })
        },
        // 批量注册
        batchRegister() {
          let list = [];
          faceList.map(item => {
            let obj = {
              id: item._id,
              name: item.name,
              url: item.pic
            }
            list.push(obj)
          })
          this.batchRegisterProcess.totalCount = list.length;
          module.batchRegister({
            // 同一人是否可以多次注册，默认true
            registerMultiple: false,
            list: list
          }, res => {
            let data = res.data || {};
            if (res.code == -107) {
              this.showToast(res.message);
              return false;
            }
            if (this.$refs.refProcess) {
              this.$refs.refProcess.open();
            }
            if (res.code == 0) {
              let status = data.status;
              if (status == "registerComplete") {
                //注册完成
                if (this.$refs.refProcess) {
                  this.$refs.refProcess.close();
                }
                console.log("成功数量：" + this.batchRegisterProcess.successCount)
                console.log("失败数量：" + this.batchRegisterProcess.failedCount);
                console.log("总数量：" + this.batchRegisterProcess.totalCount);
                this.batchRegisterProcess.failedCount = 0;
                this.batchRegisterProcess.process = 0;
                this.batchRegisterProcess.processTitle = "0";
                this.batchRegisterProcess.successCount = 0;
                this.batchRegisterProcess.totalCount = 0;
                uni.$lv.func.toast("注册结束");
              } else if (status == "registerProgress") {
                //注册中
                this.setProcessBar(data);
```

```
            }
          } else {
            //注册失败
            console.log(res)
            this.setProcessBar(data);
          }
        })
      },

      // 设置进度条
      setProcessBar(data) {
        // console.log(JSON.stringify(data))
        let successCount = data.success || this.batchRegisterProcess.successCo
        let failedCount = data.failed || this.batchRegisterProcess.failedCount
        let progress = this.batchRegisterProcess.process / 100;
        if (data.progress) {
          progress = parseFloat(data.progress);
        }
        let progressValue = progress.toFixed(4);
        //高精度乘法
        progressValue = uni.$lv.number.mul(progressValue, 100);
        this.batchRegisterProcess.failedCount = failedCount;
        this.batchRegisterProcess.process = Math.floor(progressValue);
        this.batchRegisterProcess.processTitle = progressValue + "";
        this.batchRegisterProcess.successCount = successCount;
      }
    }
  }
</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

```
<template>
  <view>
    <uni-card title="人脸识别">
      <view style="flex:1; height: 400px; position: relative;">
        <leven-uts-face ref="refLevenArcFacePro" style="flex:1; height: 400px;
          @onFaceResult="onFaceResult">
        </leven-uts-face>
```

使用方法

```
        <!-- 组件内部自定义内容 -->
        <cover-view v-if="imageBase64" style="position: absolute; right: 0; bo
          <image :src="imageBase64" style="width: 170px; height: 170px;" mode=
        </cover-view>
      </view>
      <view>
        <button type="primary" @click="register">注册人脸</button>
        <button type="primary" @click="switchCamera">切换相机</button>
        <button type="primary" @click="stop">关闭预览</button>
        <button type="primary" @click="start">开启预览</button>
        <button type="primary" @click="closeFace">关闭人脸检测</button>
        <button type="primary" @click="openFace">开启人脸检测</button>
      </view>
    </uni-card>
  </view>
</template>

<script>
  export default {
    data() {
      return {
        //组件配置
        config: {
          //相机属性，所有的参数都可以不传，不传则按默认的
          camera: {
            //相机模式，1.前置，0.后置（默认）
            facing: 1,
            // 摄像机预览圆角，默认：0
            radius: 50,
            //预览分辨率，默认：[1280,720]
            size: [1280, 720],
            //是否锁定屏幕启动方向，默认：true
            screenLocked: true
          },
          // 视频检测配置，所有参数都可以不传，不传则按默认的
          video: {
            //默认是否开启人脸识别，默认：true
            openFaceRecognizer: false,
            //是否绘制人脸关键点，默认：false
            drawPoints: false,
            //识别成功后延迟识别的时间，单位：秒，默认：3
            successContinueTime: 3,
            //人脸识别成功后是否需要移出识别区域才能再次识别，默认：false
            successOut: true,
            //是否显示人脸上方识别状态提示，默认：true
            showFaceResultNotice: true,
            //识别成功后人脸上方自定义提示信息，默认：识别成功
            successInfo: "通过",
            //人脸识别尺寸，超过该尺寸才识别，否则不识别，可根据识别成功后返回的人脸尺寸进行
            faceSize: 300,
```

```
        },
      },
      //注册的人脸图片
      imageBase64: ""
    }
  },
  methods: {
    // 注册人脸
    register() {
      if (this.$refs.refLevenArcFacePro) {
        this.$refs.refLevenArcFacePro.register({
          // 注册后保存的id（可以不传该参数，默认时间戳）
          id: "123",
          //注册后保存的名字（可以不传该参数，默认时间戳）
          name: "leven",
          //同一人脸是否可以多次注册
          registerMultiple: false,
        }, res => {
          console.log(res)
        });
      }
    },
    // 切换相机
    switchCamera() {
      if (this.$refs.refLevenArcFacePro) {
        this.$refs.refLevenArcFacePro.switchCamera(res => {
          console.log(res)
        });
      }
    },
    // 摄像头数据
    getCameraData() {
      if (this.$refs.refLevenArcFacePro) {
        this.$refs.refLevenArcFacePro.getCameraData(res => {
          console.log(res)
        });
      }
    },
    // 关闭预览
    stop() {
      if (this.$refs.refLevenArcFacePro) {
        this.$refs.refLevenArcFacePro.stop(res => {
          console.log(res)
        });
      }
    },
    // 开启预览
    start() {
      if (this.$refs.refLevenArcFacePro) {
        this.$refs.refLevenArcFacePro.start(res => {
```

```
            console.log(res)
          });
        }
      },
      // 关闭人脸检测
      closeFace() {
        if (this.$refs.refLevenArcFacePro) {
          this.$refs.refLevenArcFacePro.closeFace(res => {
            console.log(res)
          });
        }
      },
      // 开启人脸检测
      openFace() {
        if (this.$refs.refLevenArcFacePro) {
          this.$refs.refLevenArcFacePro.openFace(res => {
            console.log(res)
          });
        }
      },
      // 错误事件
      onError(e) {
        console.log(e)
      },
      // 相机打开事件
      onInitResult(e) {
        console.log(e)
      },
      // 人脸识别结果
      onFaceResult(e) {
        console.log(e)
        let detail = e.detail;
        if (detail.code == 0) {
          //识别成功
          if (detail.user) {
            this.imageBase64 = "" + detail.user.imageBa
          }
        } else {
          //识别失败
          uni.$lv.func.toast(detail.message);
        }
      }
    }
  }
</script>

<style>
```

使用方法

```
</style>
```

状态码

状态码

状态码	描述
0	成功
-1	通用错误码
-10000	创建文件夹失败
-10001	已经初始化
-10002	未初始化
-10003	用户已注册
-10004	已注册相同人脸
-10005	未检测到人脸
-10006	人脸数量大于1
-10007	未提取到人脸特征
-10008	注册失败
-10009	文件不存在
-10010	未注册
-10011	人脸对比失败
-10012	URL地址有误
-10013	创建文件失败

更新日志

2025-01-10

首次发布

申请插件所需权限

方法名

用法

- 用法如下：

```
```

- 参数说明

  无

回调

- 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
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

初始化

方法名

`init`

用法

- 用法如下：

```
module.init({
  //设置人脸检测置信度阈值，默认：0.5
  detConfThresh: 0.8,
  //设置人脸检测IOU阈值，默认：0.5
  detIouThresh: 0.8,
  //设置人脸识别相似度阈值，默认：0.8
  recConfThresh: 0.95,
  //自定义人脸存储路径
  customFaceDir: "/storage/emulated/0/levenFace"
}, 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
detConfThresh	float	否	0.5	设置人脸检测置信度阈值
detIouThresh	float	否	0.5	设置人脸检测IOU阈值
recConfThresh	float	否	0.8	设置人脸识别相似度阈值
customFaceDir	String	否	无	自定义人脸存储路径

回调

- 示例

初始化

```
{
  "data": {
    "count": 1421
  },
  "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.count	Integer	初始化成功的人脸数量
code	Integer	返回类型，0.成功，其他：请参考状态码

图片注册人脸

方法名

`imageFaceRegister`

用法

- 用法如下：

```
module.imageFaceRegister({
  //本地或网络url地址
  // url: "/storage/emulated/0/Pictures/WeiXin/mmexport1701242375957.jpg",
  url: "http://www.yeyuboke.com/svga/4.jpg",
  // 保存的id
  id: 123,
  //保存的姓名
  name: "leven",
  // 同一人是否可以多次注册，默认true
  registerMultiple: false
}, 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url地址
id	String	是	无	保存的id
name	String	是	无	保存的名称
registerMultiple	Boolean	是	无	同一个人脸是否可以多次注册

回调

- 示例

图片注册人脸


```
{
  "data": {
    "url": "/storage/emulated/0/DCIM/Camera/IMG_20241013_150358.jpg",
    "userId": "111",
    "userName": "leven1",
    "faceId": 4265,
    "imagePath": "/storage/emulated/0/levenFace/Pictures/registerFaces/111.jpg
    "registerTime": 1736490443164,
    "feature": []
  },
  "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.faceId	String	sdk保存的用户id
data.imagePath	String	注册后本地保存的头像路径
data.registerTime	Integer	注册时间
data.feature	Array[Integer]	人脸特征数据
code	Integer	返回类型，0.成功，其他：请参考状态码

清空人脸库

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

删除人脸

方法名

`deleteFace`

用法

- 用法如下：

```
module.deleteFace({
  id: "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
id	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

获取人脸

方法名

`getFace`

用法

- 用法如下：

```
module.getFace({
  id: "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
id	String	是	无	保存的用户id

回调

- 示例

```
{
  "data": {
    "url": "/storage/emulated/0/DCIM/Camera/IMG_20241013_150358.jpg",
    "userId": "111",
    "userName": "leven1",
    "faceId": 4265,
    "imagePath": "/storage/emulated/0/levenFace/Pictures/registerFaces/111.jpg
    "registerTime": 1736490443164,
    "feature": []
  },
  "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.faceId	String	sdk保存的用户id
data.imagePath	String	注册后本地保存的头像路径
data.registerTime	Integer	注册时间
data.feature	Array[Integer]	人脸特征数据
code	Integer	返回类型，0.成功，其他：请参考状态码

获取注册的人脸数量

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

获取所有人脸信息

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": "111",
        "userName": "leven1",
        "faceId": 4265,
        "imagePath": "/storage/emulated/0/levenFace/Pictures/registerFaces/111
        "registerTime": 1736490443164,
        "feature": []
      }
    ]
  },
  "message": "",
  "code": 0
}
```


- 回调说明：

获取所有人脸信息

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.faceId	String	sdk保存的用户id
data.list.imagePath	String	注册后本地保存的头像路径
data.list.registerTime	Integer	注册时间
data.list.feature	Array[Integer]	人脸特征数据
code	Integer	返回类型，0.成功，其他：请参考状态码

人脸对比

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",
  //相似度，大于或等于该相似度的人脸视为通过，默认：0.85
  similar: 0.85
}, 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
similar	float	否	0.85	相似度，大于或等于该相似度的人脸视为通过

回调

- 示例

```
{
    "data": {
        "url": "/storage/emulated/0/DCIM/Camera/IMG_20241013_150358.jpg",
        "userId": "111",
        "userName": "leven1",
        "faceId": 4265,
        "imagePath": "/storage/emulated/0/levenFace/Pictures/registerFaces/111
```

```
        "registerTime": 1736490443164,
        "feature": []
    },
    "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.faceId	String	sdk保存的用户id
data.imagePath	String	注册后本地保存的头像路径
data.registerTime	Integer	注册时间
data.featureData	Array[Integer]	人脸特征数据
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({
  // 同一人是否可以多次注册，默认true
  registerMultiple: false,
  list: [{
    //本地或网络url地址
    url: "/sdcard/DCIM/arcface/1.jpg",
    // 保存的id（可以不传该参数，默认时间戳）
    id: 10001,
    //保存的姓名（可以不传该参数，默认时间戳）
    name: "leven1"
  }, {
    //本地或网络url地址
    url: "http://www.yeyuboke.com/svga/2.jpg",
    // 保存的id（可以不传该参数，默认时间戳）
    id: 10002,
    //保存的姓名（可以不传该参数，默认时间戳）
    name: "leven2"
  }, {
    //本地或网络url地址
    url: "/sdcard/DCIM/arcface/3.jpg",
    // 保存的id（可以不传该参数，默认时间戳）
    id: 10003,
    //保存的姓名（可以不传该参数，默认时间戳）
    name: "leven3"
  }, {
    //本地或网络url地址
    url: "http://www.yeyuboke.com/svga/4.jpg",
    // 保存的id（可以不传该参数，默认时间戳）
    id: 10004,
    //保存的姓名（可以不传该参数，默认时间戳）
    name: "leven4"
  }, {
    //本地或网络url地址
    url: "/sdcard/DCIM/arcface/5.jpg",
    // 保存的id（可以不传该参数，默认时间戳）
```

```
    id: 10005,
    //保存的姓名（可以不传该参数，默认时间戳）
    name: "leven5"
  }, {
    //本地或网络url地址
    url: "/sdcard/DCIM/arcface/6.jpg",
    // 保存的id（可以不传该参数，默认时间戳）
    id: 10006,
    //保存的姓名（可以不传该参数，默认时间戳）
    name: "leven6"
  }]
}, 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
registerMultiple	Boolean	否	true	同一人脸是否可以多次注册
list	Array	是	无	注册列表
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.id	String	否	时间戳	注册时保存的id
list.name	String	否	时间戳	注册时保存的名称

回调

- 示例

```
{
    "data": {
        "url": "/sdcard/DCIM/arcface/6.jpg",
        "status": "registering",
        "userName": "邓芳",
        "progress": 0.004140786749482402,
        "faceId": 9,
```

```
        "successCount": 6,
        "imagePath": "/storage/emulated/0/Android/data/test.leven.uniplugin.co
        "registerTime": 1702028122337,
        "userId": "a5782af7653f223b012434247b585ab2",
        "failedCount": 0,
        "featureData": "AID6RAAAnELBfwQ+XXqrvM6JET1GyYm9efORPQBIP71tG3A8MvLMPB
    },
    "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.url	String	当前注册的url
data.status	String	当前注册的状态， registerStart：开始注册， registerProgress：注册中， 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.registerTime	String	注册时间
data.user.faceId	Float	SDK保存的id
data.user.feature	Array[Integer]	人脸特征数据
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

组件用法

组件名称

`leven-uts-face`

> 仅支持在nvue下使用

用法

```
<leven-uts-face ref="refLevenArcFacePro" style="flex:1; height: 400px;" :config="
  @onFaceResult="onFaceResult">
</leven-uts-face>
```

> 组件中具体的方法、属性以及事件请参考各自文档

组件属性

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.facing	Integer	否	0	相机模式，1.前置，0.后置
camera.radius	Integer	否	0	摄像机预览圆角
camera.size	Array[Integer]	否	[1280, 720]	预览分辨率
camera.screenLocked	Boolean	否	true	是否锁定屏幕启动方向
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.openFaceRecognizer	boolean	否	true	默认是否开启人脸识别
video.drawPoints	boolean	否	false	是否绘制人脸关键点
video.successContinueTime	Integer	否	3	识别成功后延迟识别的时间，单位：秒
video.successOut	Boolean	否	true	人脸识别成功后是否需要移出识别区域才能再次识别
video.showFaceResultNotice	Boolean	否	true	是否显示人脸上方识别状态提示

组件属性

参数名	参数类型	是否必填	默认值	参数描述
video.successInfo	String	否	识别成功	识别成功后人脸上方自定义提示信息
video.faceSize	Integer	否	0	人脸识别尺寸，超过该尺寸才识别，否则不识别，可根据识别成功后返回的人脸尺寸进行调整，默认：0，不做人脸尺寸识别

组件方法

注册人脸

方法名

`register`

用法

- 用法如下：

```
if (this.$refs.refLevenArcFacePro) {
  this.$refs.refLevenArcFacePro.register({
    // 注册后保存的id
    id: "123",
    //注册后保存的名字
    name: "leven",
    //同一人脸是否可以多次注册
    registerMultiple: 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
id	String	是	无	注册后保存的id
name	String	是	无	注册后保存的名字
registerMultiple	Boolean	是	无	同一人脸是否可以多次注册

回调

- 示例

```
{
    "data": {
        "url": "/storage/emulated/0/DCIM/Camera/IMG_20241013_150358.jpg",
```

```
        "userId": "111",
        "userName": "leven1",
        "faceId": 4265,
        "imagePath": "/storage/emulated/0/levenFace/Pictures/registerFaces/111
        "registerTime": 1736490443164,
        "feature": []
    },
    "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.faceId	String	sdk保存的用户id
data.imagePath	String	注册后本地保存的头像路径
data.registerTime	Integer	注册时间
data.feature	Array[Integer]	人脸特征数据
code	Integer	返回类型，0.成功，其他：失败

切换相机

方法名

`switchCamera`

用法

- 用法如下：

```
if (this.$refs.refLevenArcFacePro) {
  this.$refs.refLevenArcFacePro.switchCamera(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

关闭预览

方法名

`stop`

用法

- 用法如下：

```
if (this.$refs.refLevenArcFacePro) {
  this.$refs.refLevenArcFacePro.stop(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

开启预览

方法名

`start`

用法

- 用法如下：

```
if (this.$refs.refLevenArcFacePro) {
  this.$refs.refLevenArcFacePro.start(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

关闭人脸检测

方法名

`closeFace`

用法

- 用法如下：

```
if (this.$refs.refLevenArcFacePro) {
  this.$refs.refLevenArcFacePro.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

开启人脸检测

方法名

`openFace`

用法

- 用法如下：

```
if (this.$refs.refLevenArcFacePro) {
  this.$refs.refLevenArcFacePro.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

组件事件

初始化结果事件

介绍

`onInitResult`

用法

```
<leven-uts-face ref="refLevenArcFacePro" style="flex:1; height: 400px;" @onInitRes
```

回调示例

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

回调内容

```
{
    "status": true
}
```

回调说明

参数名	参数类型	参数描述
status	Boolean	初始化状态，true：成功，false：失败

错误事件

介绍

`onError`

用法

```
<leven-uts-face ref="refLevenArcFacePro" style="flex:1; height: 400px;" @onError="
```

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

人脸识别结果事件

事件名称

`onFaceResult`

用法

```
<leven-uts-face ref="refLevenArcFacePro" style="flex:1; height: 400px;" @onFaceRes
```

示例

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

回调示例

```
{
  "user": {
    "userName": "leven1",
    "faceId": 4265,
    "imagePath": "/storage/emulated/0/levenFace/Pictures/registerFaces/111.jpg",
    "imageBase64": "",
    "registerTime": 1736490443164,
    "feature": [],
    "userId": "111",
    "faceSize": {
      "height": 355.4889221191406,
      "width": 302.2303771972656
    }
  },
  "message": "识别成功",
  "code": 0
}
```

回调说明：

参数名	参数类型	参数描述
message	String	消息提示
code	String	识别结果的code值，0.识别成功，其他：识别失败
user	Object	识别的用户信息
user.userId	String	用户id
user.userName	String	用户的姓名
user.imagePath	String	识别后的头像图片
user.imageBase64	String	识别的人脸图片base64数据
user.faseSize	Object	当前人脸识别成功的人脸框尺寸
user.faseSize.width	float	人脸框尺寸宽度
user.faseSize.height	float	人脸框尺寸高度
user.registerTime	String	注册时间
user.faceId	String	SDK保存的id
user.similar	Float	识别相似度
user.feature	Array[Integer]	识别的人脸特征