uniapp 安卓虹软人脸识别5.0版UTS原生插件

目　录

使用方法

介绍

虹软人脸识别增值版5.0版支持在线激活，离线激活，支持图片人脸识别（可识别网络图片），活体检测，离线识别，相机预览旋转，相机人脸识别，批量注册（支持网络图片）等，识别率更高速度更快，支持保存用户的id和名称

> 本插件为虹软人脸增值版5.0sdk，因为5.0底层模型修改所以不支持4.2版本直接升级
> 4.2版本插件地址：https://ext.dcloud.net.cn/plugin?id=15808
> 3.0免费版插件地址：https://ext.dcloud.net.cn/plugin?id=15096

联系作者

关注微信公众号可联系作者

官方文档

https://ai.arcsoft.com.cn/manual/docs#/190

SDK版本

ArcFace v5.0

插件地址

https://ext.dcloud.net.cn/plugin?id=21584

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

```
import * as module from "@/uni_modules/leven-uts-arc-pro5"
```

- 示例

```
<template>
  <view>
    <uni-card title="激活">
      <button type="primary" @click="requestPermission">申请权限</button>
      <button type="primary" @click="getDeviceFinger">获取设备指纹信息</button>
      <button type="primary" @click="activeOnline">在线激活</button>
      <button type="primary" @click="activeOffline">离线激活</button>
      <button type="primary" @click="isActivated">查询激活状态</button>
      <button type="primary" @click="customFaceDataPath">自定义人脸数据存储路径</b
    </uni-card>
    <uni-card class="uni-card-box" title="日志">
      <view><text style="font-size: 14px; flex-wrap: wrap;">{{logStr}}</text><
    </uni-card>
  </view>
</template>

<script>
  import * as module from "@/uni_modules/leven-uts-arc-pro5"
  export default {
    data() {
      return {
        logStr: "",
      }
    },
    methods: {
      // 申请权限
      requestPermission() {
```

```
      module.requestPermissions(res => {
        this.writeLog("申请权限：" + JSON.stringify(res))
      })
    },
    // 在线激活
    activeOnline() {
      module.activeOnline({
        activeKey: "8551-11UP-712L-WWEY",
        appId: "FPMZ54pFtPgHYQWoHgPWCJEfkh9X435ZbKCeJzbZc1N3",
        sdkKey: "7yRrFX68rAFGPShnZ6t7FLXPdCoAxcmGJfhQ2w1owpFc"
      }, res => {
        this.writeLog("在线激活：" + JSON.stringify(res))
      })
    },
    // 获取设备指纹信息
    getDeviceFinger() {
      module.getDeviceFinger(res => {
        this.writeLog("获取设备指纹信息：" + JSON.stringify(res))
      })
    },
    // 查询激活状态
    isActivated() {
      module.isActivated(res => {
        this.writeLog("查询激活状态：" + JSON.stringify(res))
      })
    },
    // 离线激活
    activeOffline() {
      module.activeOffline({
        path: "/storage/emulated/0/Download/WeiXin/85Q111L6B11R1WNE.dat"
      }, res => {
        this.writeLog("离线激活：" + JSON.stringify(res))
      })
    },
    // 自定义人脸数据存储路径
    customFaceDataPath() {
      module.customFaceDataPath({
        //文件夹路径
        path: "/storage/emulated/0/levenArcFacePro"
      }, res => {
        this.writeLog("自定义人脸数据存储路径：" + JSON.stringify(res))
      })
    },
    showToast(content) {
      console.log(content)
      // uni.showToast({
      //   icon: "none",
      //   title: content
      // })
    },
```

```
        // 写日志
        writeLog(str) {
          console.log(str);
          let logStr = uni.$lv.date.format(null, "yyyy-mm-dd hh:MM:ss") + " " +
          this.logStr = logStr + this.logStr;
        }
      }
    }
</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-uts-arc-face5 ref="refLevenArcFacePro" style="flex:1; height: 500px;" :
      @onCameraOpened="onCameraOpened" @onCameraClosed="onCameraClosed" @onCam
      @onInitResult="onInitResult" @onFaceResult="onFaceResult">
    </leven-uts-arc-face5>
```

- 示例

```
  <template>
  <view>
    <uni-card title="人脸识别">
      <view style="flex:1; height: 500px; position: relative;">
        <leven-uts-arc-face5 ref="refLevenArcFacePro" style="flex:1; height: 5
          @onCameraOpened="onCameraOpened" @onCameraClosed="onCameraClosed" @o
          @onInitResult="onInitResult" @onFaceResult="onFaceResult">
        </leven-uts-arc-face5>
        <!-- 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>
        <button type="primary" @click="getCameraData">摄像头数据</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: {
            // 相机预览旋转角度
            rotation: 0,
            //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: {
            // 视频检测角度，可接收参数，0,90,180,270,360(默认)
            orient: 360,
            // 人脸框是否处于X反向状态，如果未设置该参数人脸框和人脸处于反向请将该参数设置为
            isContraryX: false,
            // 人脸框是否处于Y反向状态，如果未设置该参数人脸框和人脸处于反向请将该参数设置为
            isContraryY: false,
            // 识别阈值（默认：0.8）
            similar: 0.9,
            // 是否进行活体检测(默认为true)
            liveness: true,
            //识别是否展示面部信息(默认为true)
            showFaceInfo: false,
            //活体检测阈值设置
            livenessParams: {
              //可见光活体检测阈值，默认：0.5
              rgb: 0.5,
              //红外活体检测阈值，默认：0.7
              ir: 0.7
            },
            //是否显示人脸上方识别状态提示，默认：true
            showFaceResultNotice: true,
            //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 => {
            if (res.code == 0) {
              uni.$lv.func.toast("注册成功");
            } else {
              uni.$lv.func.toast(res.message);
            }
          });
        }
      },
      // 切换相机
      switchCamera() {
        if (this.$refs.refLevenArcFacePro) {
          this.$refs.refLevenArcFacePro.switchCamera(res => {
            uni.$lv.func.toast(JSON.stringify(res));
          });
        }
      },
      // 摄像头数据
      getCameraData() {
        if (this.$refs.refLevenArcFacePro) {
          this.$refs.refLevenArcFacePro.getCameraData(res => {
            uni.$lv.func.toast(JSON.stringify(res));
          });
        }
      },
      // 关闭预览
      stop() {
        if (this.$refs.refLevenArcFacePro) {
          this.$refs.refLevenArcFacePro.stop(res => {
            uni.$lv.func.toast(JSON.stringify(res));
          });
        }
```

```
    },
    // 开启预览
    start() {
      if (this.$refs.refLevenArcFacePro) {
        this.$refs.refLevenArcFacePro.start(res => {
          uni.$lv.func.toast(JSON.stringify(res));
        });
      }
    },
    // 关闭人脸检测
    closeFace() {
      if (this.$refs.refLevenArcFacePro) {
        this.$refs.refLevenArcFacePro.closeFace(res => {
          uni.$lv.func.toast(JSON.stringify(res));
        });
      }
    },
    // 开启人脸检测
    openFace() {
      if (this.$refs.refLevenArcFacePro) {
        this.$refs.refLevenArcFacePro.openFace(res => {
          uni.$lv.func.toast(JSON.stringify(res));
        });
      }
    },
    // 错误事件
    onError(e) {
      let detail = e.detail || {};
      uni.$lv.func.toast("错误事件：" + JSON.stringify(detail));
    },
    // 相机打开事件
    onCameraOpened(e) {
      console.log(e)
    },
    // 相机关闭事件
    onCameraClosed(e) {
      console.log(e)
    },
    // 相机配置改变事件
    onCameraConfigurationChanged(e) {
      console.log(e)
    },
    // 初始化结果事件
    onInitResult(e) {
      console.log(e)
    },
    // 人脸识别结果
    onFaceResult(e) {
      console.log(e)
      let detail = e.detail;
```

使用方法

```
          if (detail.compareImageBase64) {
            this.imageBase64 = "" + detail.compareImageBa
          }
        }
      }
    }
</script>

<style>

</style>
```

状态码

状态码

> 延续了4.2版本的状态码，新增了部分状态

状态码	描述
0	成功
-1	通用错误码
-100	注册时带口罩
-101	裁剪图片错误
-102	未发现人脸
-103	注册失败
-104	已注册
-105	存在相同人脸
-106	人脸未注册
-107	未初始化
-10001	缺少必要的权限
-10002	文件不存在
-10003	url地址有误

更新日志

- [优化] 相机属性新增参数是否水平镜像

- [优化] 解决targetSdk大于29时存储权限报错问题
- [新增] 视频检测属性新增参数 识别成功后是否返回当前识别的图片Base64数据

  > 设置该参数为true时在性能低的设备上数据返回会有延迟

- [优化] 优化人脸库为空识别时不返回识别结果的问题
- [新增] 新增方法【设置识别成功的文案】

- [优化] 人脸识别组件检测属性新增参数是否绘制人脸识别框

- [优化] 人脸识别视频检测属性新增参数“是否单人脸识别”

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

获取设备指纹信息

方法名

`getDeviceFinger`

用法

- 用法如下：

```
module.getDeviceFinger(res => {
  // this.showToast(JSON.stringify(res))
  console.log(res);
})
```

- 参数说明

  无

回调

- 示例

```
{
    "data": {
        "info": "wOkJ7Om5ySZUXb1rif2MiZaMLZUWTmrmUA13rDtzmkBm1pgkpGJJ/5DZHZvLC
    },
    "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.info	String	指纹数据

获取设备指纹信息

参数名	参数类型	参数描述
code	Integer	返回类型，0.成功，其他：请参考状态码

在线激活

方法名

`activeOnline`

用法

- 用法如下：

```
module.activeOnline({
  activeKey: "85Q1-11MH-H13X-QK97",
  appId: "BPeKN1LAULmr364FMisKF3Cds2U3cNaec8PYvxkiSdWm",
  sdkKey: "8Hzo8nFpMj8DWgAaTsQySH2QJoQ1YKUDEAX4w2RL24BU"
}, res => {
  this.showToast(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
activeKey	String	是	无	虹软申请的激活码
appId	String	是	无	虹软申请的appId
sdkKey	String	是	无	虹软申请的sdkKey

回调

- 示例

```
{
    "data": {},
    "message": "激活成功",
    "code": 0
}
```

- 回调说明：

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

离线激活

方法名

`activeOffline`

用法

- 用法如下：

```
module.activeOffline({
  path: "/storage/emulated/0/Download/WeiXin/85Q111L6B11R1WNE.dat"
}, 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
path	String	是	无	从虹软下载的离线激活文件

回调

- 示例

```
{
    "data": {},
    "message": "激活成功",
    "code": 0
}
```

- 回调说明：

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

离线激活

参数名	参数类型	参数描述
code	Integer	返回类型，0.成功，其他：请参考状态码

查询激活状态

方法名

`isActivated`

用法

- 用法如下：

```
module.isActivated(res => {
  this.writeLog("查询激活状态：" + JSON.stringify(res))
})
```

- 参数说明

  无

回调

- 示例

```
{
  "data": {
    "isActivated": true
  },
  "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.isActivated	Boolean	是否激活，true：已激活，false：未激活
code	Integer	返回类型，0.成功，其他：请参考状态码

自定义人脸数据存储路径

方法名

`customFaceDataPath`

> 自定义人脸数据存储路径需要在激活和初始化之前调用

用法

- 用法如下：

```
module.customFaceDataPath({
  //文件夹路径
  path: "/storage/emulated/0/levenArcFacePro"
}, 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
path	String	是	无	自定义的存储文件夹路径

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示

自定义人脸数据存储路径

参数名	参数类型	参数描述
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

初始化人脸库

方法名

`init`

用法

- 用法如下：

```
module.init(res => {
  this.showToast(JSON.stringify(res))
})
```

- 参数说明

  无

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：请参考状态码

获取图片人脸信息

方法名

`getImageFace`

用法

- 用法如下：

```
module.getImageFace({
  //本地或网络url地址
  url: "/sdcard/DCIM/Camera/IMG_20230225_175710.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": {
        "list": [
            {
                "age": 34,
                "face3D": {
                    "yaw": 0.4799368977546692,
                    "pitch": -4.52293586730957,
                    "roll": 2.4426019191741943
                },
                "liveness": 1,
                "mask": 0,
                "gender": 1
```

```
            },
            {
                "age": 9,
                "face3D": {
                    "yaw": -3.784116744995117,
                    "pitch": -14.625612258911133,
                    "roll": 2.119079828262329
                },
                "liveness": -2,
                "mask": 0,
                "gender": 0
            }
        ]
    },
    "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.list	Array	图片人脸信息集合
data.list.age	int	年龄
data.list.face3D	Object	3D角度信息
data.list.face3D.yaw	Float	偏航角
data.list.face3D.roll	Float	横滚角
data.list.face3D.pitch	Float	俯仰角
data.list.mask	int	是否戴口罩，0.未戴口罩，1.戴口罩
data.list.gender	int	性别，0：男性，1：女性，-1：未知
data.list.liveness	Integer	活体信息，0：非真人，1.真人，-1.不确定，-2.传入人脸数 > 1，-3. 人脸过小，-4.角度过大，-5.人脸超出边界
data.list.face	Object	人脸面部信息

参数名	参数类型	参数描述
data.list.face.bottom	Integer	面部信息底部点位
data.list.face.top	Integer	面部信息顶部点位
data.list.face.left	Integer	面部信息左边点位
data.list.face.right	Integer	面部信息右边点位
data.list.face.width	Integer	面部信息宽度
data.list.face.height	Integer	面部信息高度
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（可以不传该参数，默认时间戳）
  id: 123,
  //保存的姓名（可以不传该参数，默认时间戳）
  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	否	时间戳	保存的id
name	String	否	时间戳	保存的名称
registerMultiple	Boolean	否	true	同一个人脸是否可以多次注册

回调

- 示例

图片注册人脸


```
{
    "data": {
        "userId": "123",
        "userName": "leven",
        "faceId": 1,
        "imagePath": "/storage/emulated/0/Android/data/test.leven.uniplugin.co
        "registerTime": 1702027511229,
        "featureData": "AID6RAAAnEJ0JB09VAttvVYnEL3JJfG7vGGKvEFAlj3BmDA87mJqvI
    },
    "message": "注册成功",
    "code": 0
}
```


- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
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	String	人脸特征数据，base64加密
data.imageBase64	String	人脸图片base64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": {
        "userId": "123",
        "userName": "leven",
        "faceId": 3,
        "imagePath": "/storage/emulated/0/Android/data/test.leven.uniplugin.co
        "registerTime": 1702027931893,
        "featureData": "AID6RAAAnEJ0JB09VAttvVYnEL3JJfG7vGGKvEFAlj3BmDA87mJqvI
    },
    "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	String	人脸特征数据，base64加密
data.imageBase64 `v1.4.0`	String	人脸图片base64数据
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": "123",
                "userName": "leven",
                "faceId": 3,
                "imagePath": "/storage/emulated/0/Android/data/test.leven.unip
                "registerTime": 1702027931893,
                "featureData": "AID6RAAAnEJ0JB09VAttvVYnEL3JJfG7vGGKvEFAlj3BmD
            }
        ]
    },
    "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.featureData	String	人脸特征数据，base64加密
data.list.imageBase64	String	人脸图片base64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.featureData	Integer	人脸特征数据，base64加密
data.user.imageBase64	Integer	注册人脸图片的base64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

参数名	参数类型	参数描述
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

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": {
                "userId": "123",
                "userName": "leven",
                "faceId": 3,
                "imagePath": "/storage/emulated/0/Android/data/test.leven.unip
                "registerTime": 1702027931893,
```

```
                "featureData": "AID6RAAAnEJ0JB09VAttvVYnEL3JJfG7vGGKvEFAlj3BmD
            },
    "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	String	人脸特征数据，base64加密
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-arc-face5`

> 仅支持在nvue下使用

用法

```
<leven-uts-arc-face5 ref="refLevenArcFacePro" style="flex:1; height: 500px;" :con
  @onCameraOpened="onCameraOpened" @onCameraClosed="onCameraClosed" @onCameraConf
  @onInitResult="onInitResult" @onFaceResult="onFaceResult">
</leven-uts-arc-face5>
```

> 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.rotation	Integer	否	无	相机旋转角度
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
camera.isHorizontalMirror `v1.1.3`	Boolean	否	false	是否水平镜像
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.orient	Integer	否	360	视频检测角度，可接收参数，0,90,180,270,360
video.isContraryX	Integer	否	false	人脸框是否处于X反向状态，如果未设置该参数人脸框和人脸处于反向请将该参数设置为true

组件属性

参数名	参数类型	是否必填	默认值	参数描述
video.isContraryY	Integer	否	false	脸框是否处于Y反向状态，如果未设置该参数人脸框和人脸处于反向请将该参数设置为true
video.similar	float	否	0.8	识别阈值
video.liveness	Boolean	否	true	是否进行活体检测
video.showFaceInfo	Boolean	否	true	识别是否展示面部信息
video.livenessParams	Object	否	无	活体检测阈值设置
video.livenessParams.rgb	float	否	0.5	可见光活体检测阈值
video.livenessParams.ir	float	否	0.5	红外活体检测阈值
video.showFaceResultNotice	Boolean	否	true	是否显示人脸上方识别状态提示
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
video.isKeepMaxFace `v1.0.4`	boolean	否	true	是否单人脸识别，如果是多人脸识别将参数设置为false，同时设置maxDetectFaces参数
video.maxDetectFaces `v1.0.4`	Integer	否	1	最大识别的人脸数量
video.drawFaceBorder `v1.0.10`	boolean	否	true	是否绘制人脸识别框

组件属性

参数名	参数类型	是否必填	默认值	参数描述
video.compareImage `v1.1.2`	boolean	否	true	识别成功后是否返回当前识别的图片Base64数据

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（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)
  });
}
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
id	String	否	时间戳	注册后保存的id
name	String	否	时间戳	注册后保存的名字
registerMultiple	Boolean	否	true	同一人脸是否可以多次注册

回调

- 示例

```
{
    "data": {
        "userId": "123",
```

```
        "userName": "leven",
        "faceId": 1431,
        "imagePath": "/storage/emulated/0/Android/data/test.leven.uniplugin.co
        "registerTime": 1702028708670,
        "featureData": "AID6RAAAnELE15+8LXZpPVYABT2KwR09e00ePvjJhrzCVQA9YPr3Ow
    },
    "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	String	人脸特征数据，base64加密
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

摄像头数据

方法名

`getCameraData`

用法

- 用法如下：

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

- 参数说明

  无

回调

- 示例

```
{
  "data": {
    "supportedPreviewSizes": [
      {
        "height": 1080,
        "width": 1920
      },
      {
        "height": 1200,
        "width": 1600
      },
      {
        "height": 1080,
        "width": 1440
      },
      {
        "height": 1080,
        "width": 1260
```

```
        },
        {
          "height": 960,
          "width": 1280
        },
        {
          "height": 1080,
          "width": 1080
        },
        {
          "height": 720,
          "width": 1584
        },
        {
          "height": 1080,
          "width": 960
        },
        {
          "height": 768,
          "width": 1280
        },
        {
          "height": 720,
          "width": 1280
        },
        {
          "height": 768,
          "width": 1024
        },
        {
          "height": 738,
          "width": 1024
        },
        {
          "height": 720,
          "width": 960
        },
        {
          "height": 540,
          "width": 960
        },
        {
          "height": 720,
          "width": 720
        },
        {
          "height": 600,
          "width": 800
        },
        {
```

```
        "height": 480,
        "width": 864
      },
      {
        "height": 540,
        "width": 720
      },
      {
        "height": 480,
        "width": 800
      },
      {
        "height": 480,
        "width": 720
      },
      {
        "height": 480,
        "width": 640
      },
      {
        "height": 400,
        "width": 640
      },
      {
        "height": 360,
        "width": 640
      },
      {
        "height": 288,
        "width": 352
      },
      {
        "height": 240,
        "width": 320
      },
      {
        "height": 180,
        "width": 320
      },
      {
        "height": 144,
        "width": 176
      }
    ],
    "cameraId": 1
  },
  "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.supportedPreviewSizes	Object	相机支持的预览分辨率
data.supportedPreviewSizes.width	Integer	分辨率的宽
data.supportedPreviewSizes.height	Integer	分辨率的高
data.cameraId	Integer	当前预览的摄像头，1.前置，0.后置
code	Integer	返回类型，0.成功，其他：失败

自定义识别成功文案 v1.1.0

方法名

`setFaceResult` `v1.1.0`

用法

- 用法如下：

```
if (this.$refs.refLevenArcFacePro) {
    //自定义识别成功文案
  let message = "识别人员：" + detail.userName;
  this.$refs.refLevenArcFacePro.setFaceResult({
    //识别返回的人脸检测id
    trackId: detail.trackId,
    message: message
  }, 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
trackId	Integer	是	无	人脸识别的人脸镜像id，识别成功后会返回
message	String	是	无语	自定义文案内容

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

组件事件

错误事件

介绍

onError

用法

```
<leven-uts-arc-face5 ref="refLevenArcFacePro" @onError="onError">
</leven-uts-arc-face5>
```

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

相机打开事件

事件名称

`onCameraOpened`

用法

```
<leven-uts-arc-face5 ref="refLevenArcFacePro" @onCameraOpened="onCameraOpened">
</leven-uts-arc-face5>
```

返回示例

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

回调说明：

参数名	参数类型	参数描述
message	String	消息提示

相机关闭事件

事件名称

`onCameraClosed`

用法

```
<leven-uts-arc-face5 ref="refLevenArcFacePro" @onCameraClosed="onCameraClosed" </
```

返回示例

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

回调说明：

参数名	参数类型	参数描述
message	String	消息提示

相机配置改变事件

事件名称

`onCameraConfigurationChanged`

用法

```
<leven-uts-arc-face5 ref="refLevenArcFacePro"@onCameraConfigurationChanged="onCame
```

返回示例

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

回调说明：

参数名	参数类型	参数描述
cameraID	Integer	相机id
displayOrientation	Integer	相机旋转角度

初始化结果事件

介绍

onInitResult

用法

```
<leven-uts-arc-face5 ref="refLevenArcFacePro"
          @onInitResult="onInitResult">
        </leven-uts-arc-face5>
```

回调示例

```
onInitResult(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
data.status	Boolean	初始化结果，true:初始化成功，false：初始化失败

初始化结果事件

参数名	参数类型	参数描述
code	Integer	返回类型，0.成功，其他：失败

人脸识别结果

事件名称

`onFaceResult`

用法

```
<leven-arcFacePro ref="refLevenArcFacePro" style="flex:1; height: 500px;" @onFaceR
```

示例

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

回调示例

```
"detail": {
    "userId": "123",
    "status": true,
    "userName": "leven",
    "imagePath": "/storage/emulated/0/Android/data/test.leven.uniplugin.com/files/
    "registerTime": 1702028708670,
    "code": 0,
    "message": "",
    "faceId": 1431,
    "faceSize": 400,
    "similar": 0.9784093499183655,
    "featureData": "AID6RAAAnELE15+8LXZpPVYABT2KwR09e00ePvjJhrzCVQA9YPr3OwpYXLz3Q
}
```

回调说明：

人脸识别结果

参数名	参数类型	参数描述
trackId `v1.1.0`	String	人脸镜像id
userId	String	用户id
status	Boolean	识别结果，true.识别成功，false.识别失败
userName	String	用户的姓名
imagePath	String	识别后的头像图片
registerImageBase64	String	识别后的头像图片base64格式
imageBase64	String	人脸图片base64数据
faseSize	Integer	当前人脸识别成功的人脸框尺寸
registerTime	String	注册时间
code	String	识别结果的code值，0.识别成功，其他：识别失败(虹软的错误码)
message	String	消息提示
faceId	String	SDK保存的id
similar	Float	识别相似度
compareImageBase64	String	当前识别的人脸图片
featureData	String	识别的人脸特征，base64加密