uniapp 安卓虹软人脸识别增值版原生插件

目　录

使用方法

介绍

虹软人脸识别增值版支持在线激活，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，支持保存用户的id和名称

> 本插件是增值版插件，如果您想使用免费版插件请点击这里

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 v4.2

插件地址

https://ext.dcloud.net.cn/plugin?id=15808

权限

使用方法

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

示例文件下载

> 如果下载的文件不是zip文件，请将扩展名改为zip即可

激活

- 用法

  在需要使用插件的页面加载以下代码

```
const module = uni.requireNativePlugin("leven-arcFacePro-ActiveModule");
```

- 示例

```
<template>
  <view>
    <uni-card title="激活">
      <button type="primary" @click="activeOnline">在线激活</button>
      <button type="primary" @click="getDeviceFinger">获取设备指纹信息</button>
      <button type="primary" @click="activeOffline">离线激活</button>
    </uni-card>
  </view>
</template>

<script>
  const module = uni.requireNativePlugin("leven-arcFacePro-ActiveModule");
  export default {
    data() {
      return {

      }
    },
    methods: {
      // 在线激活
      activeOnline() {
        module.activeOnline({
          activeKey: "85Q1-11MH-H13X-QK97",
          appId: "BPeKN1LAULmr364FMisKF3Cds2U3cNaec8PYvxkiSdWm",
          sdkKey: "8Hzo8nFpMj8DWgAaTsQySH2QJoQ1YKUDEAX4w2RL24BU"
        }, res => {
```

```
            this.showToast(JSON.stringify(res))
          })
        },
        // 获取设备指纹信息
        getDeviceFinger() {
          module.getDeviceFinger(res => {
            this.showToast(JSON.stringify(res))
          })
        },
        // 离线激活
        activeOffline() {
          module.activeOffline({
            path: "/storage/emulated/0/Download/WeiXin/85Q111L6B11R1WNE.dat"
          }, res => {
            this.showToast(JSON.stringify(res))
          })
        },
        showToast(content) {
          console.log(content)
          // uni.showToast({
          //   icon: "none",
          //   title: content
          // })
        }
      }
    }
</script>

<style>

</style>
```

人脸管理

- 用法

  在需要使用插件的页面添加以下代码

```
    const module = uni.requireNativePlugin("leven-arcFacePro-FaceManagerModule
```

- 示例

```
<template>
  <view>
    <uni-card title="人脸管理">
      <button type="primary" @click="init">初始化</button>
```

```
      <button type="primary" @click="getImageFace">获取图片人脸信息</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="getAllFace">获取所有人脸信息</button>
      <button type="primary" @click="batchRegister">批量注册</button>
      <button type="primary" @click="getFaceCount">获取注册的人脸数量</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"
  const module = uni.requireNativePlugin("leven-arcFacePro-FaceManagerModule")
  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: {
      // 初始化
      init() {
        module.init(res => {
```

```
            this.showToast(JSON.stringify(res))
          })
        },
        // 获取图片人脸信息
        getImageFace() {
          module.getImageFace({
            //本地或网络url地址
            url: "/sdcard/DCIM/Camera/IMG_20230225_175710.jpg",
          }, res => {
            this.showToast(JSON.stringify(res))
          })
        },
        // 图片注册人脸
        imageFaceRegister() {
          module.imageFaceRegister({
            //本地或网络url地址
            // url: "/storage/emulated/0/Pictures/WeiXin/mmexport1701242375957.j
            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))
          })
        },
        // 清空人脸库
        clearFace() {
          module.clearFace(res => {
            this.showToast(JSON.stringify(res))
          })
        },
        // 删除人脸
        deleteFace() {
          module.deleteFace({
            id: "123"
          }, res => {
            this.showToast(JSON.stringify(res))
          })
        },
        // 获取人脸
        getFace() {
          module.getFace({
            id: "123"
          }, res => {
            this.showToast(JSON.stringify(res))
          })
        },
```

```
        // 批量注册
        batchRegister() {
          if (this.$refs.refProcess) {
            this.$refs.refProcess.open();
          }
          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 => {
            // console.log(res);
            if (res.code == 0) {
              let data = res.data || {};
              let status = data.status;
              if (status == "registerFinish") {
                //注册完成
                if (this.$refs.refProcess) {
                  this.$refs.refProcess.close();
                }
                uni.$lv.func.toast("注册结束");
              } else if (status == "registering") {
                //注册中
                let successCount = data.successCount || 0;
                let failedCount = data.failedCount || 0;
                let progress = data.progress || 0;
                let progressValue = progress.toFixed(4);
                //高精度乘法
                progressValue = uni.$lv.number.mul(progressValue, 100);
                this.batchRegisterProcess.failedCount = failedCount;
                this.batchRegisterProcess.process = Math.floor(progressValue);
                this.batchRegisterProcess.processTitle = progressValue + "";
                this.batchRegisterProcess.successCount = successCount;
              }
            }
          })
        },
        // 获取所有人脸信息
        getAllFace() {
          module.getAllFace(res => {
            this.showToast(JSON.stringify(res))
```

```
        })
      },
      // 获取注册的人脸数量
      getFaceCount() {
        module.getFaceCount(res => {
          this.showToast(JSON.stringify(res))
        })
      },
      showToast(content) {
        uni.showToast({
          icon: "none",
          title: content
        })
      }
    }
  }
</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-arcFacePro ref="refLevenArcFacePro" style="flex:1; height: 500px;" :cam
      @onCameraOpened="onCameraOpened" @onCameraClosed="onCameraClosed" @onFac
    </leven-arcFacePro>
```

- 示例

```
  <template>
    <view>
    <uni-card title="人脸识别">
      <view style="flex:1; height: 500px; position: relative;">
        <leven-arcFacePro ref="refLevenArcFacePro" style="flex:1; height: 500p
          @onCameraOpened="onCameraOpened" @onCameraClosed="onCameraClosed" @o
        </leven-arcFacePro>
        <!-- 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="setCamera">更新摄像机属性</button>
        <button type="primary" @click="setVideo">更新视频检测属性</button>
        <button type="primary" @click="reInit">重载组件</button>
        <button type="primary" @click="deleteFace">删除人脸</button>
      </view>
    </uni-card>
    </view>
    </template>

    <script>
    export default {
    data() {
      return {
        //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t
          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t
          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,
            // 活体 FQ 检测阈值，默认：0.65
            fq: 0.65
          },
          //识别结果是否返回识别注册图片的base64数据，可能影响性能，默认：false
          resultRegisterBase64: true,
          //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调
          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)
          });
        }
      },
      // 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)
        });
      }
    },
    // 设置摄像机属性，设置完属性后需调用一次重载组件方可生效
    setCamera() {
      if (this.$refs.refLevenArcFacePro) {
        let params = JSON.parse(JSON.stringify(this.camera));
        params.rotation = 90;
        params.facing = 1;
        params.radius = 10;
        params.size = [800, 600];
        this.$refs.refLevenArcFacePro.setCamera(params, res => {
          console.log(res)
        });
      }
    },
    // 设置视频检测属性，设置完属性后需调用一次重载组件方可生效
    setVideo() {
      if (this.$refs.refLevenArcFacePro) {
        let params = JSON.parse(JSON.stringify(this.video));
        params.orient = 360;
        params.liveness = false;
        params.isContraryX = true;
        params.isContraryY = true;
        params.similar = 0.95;
        params.showFaceInfo = true;
        this.$refs.refLevenArcFacePro.setVideo(params, res => {
          console.log(res)
        });
      }
```

```
    },
    // 重载组件
    reInit() {
      if (this.$refs.refLevenArcFacePro) {
        this.$refs.refLevenArcFacePro.reInit(res => {
          console.log(res)
        });
      }
    },
    // 重载组件
    deleteFace() {
      if (this.$refs.refLevenArcFacePro) {
        this.$refs.refLevenArcFacePro.deleteFace({
          id: "123"
        }, res => {
          console.log(res)
        });
      }
    },
    // 错误事件
    onError(e) {
      console.log(e)
    },
    // 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)
    },
    // 人脸识别结果
    onFaceResult(e) {
      console.log(e)
      let detail = e.detail;
      if (detail.compareImageBase64) {
        this.imageBase64 = "" + detail.compareImageBa
      }
    }
  }
  }
  </script>

  <style>
```

```
      </style>
```

状态码

状态码

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

更新日志

2025-03-19 `v1.4.5`

- [优化] 新增相机配置组件内部padding

2024-09-20 `v1.4.3`

- [优化] 获取图片人脸信息新增面部数据

2024-09-15 `v1.4.2`

- [优化] 识别属性新增参数 `faseSize` ，人脸识别框尺寸
- [优化] 识别结果新增返回参数 `faseSize` ，人脸识别框尺寸

2024-09-02 `v1.4.1`

- [优化] 修复本地人脸图片对比异常的bug

2024-08-20 `v1.4.0`

- [新增] 增加照片对比功能
- [优化] 获取人脸新增返回图片的base64数据
- [优化] 新增返回识别成功后当前识别的人脸图片base64数据
- [优化] 事件回调兼容vue3

2024-06-07 `v1.3.0`

- [优化] 人脸识别结果新增返回识别成功的人脸图片base64格式
- [优化] 移除人脸识别上方小方块
- [优化] 更新虹软最新的sdk支持到安卓14
- [优化] 优化批量注册内存占用问题
- [优化] 其他部分细节优化

- [新增] 视频检测配置新增识别结果是否返回识别注册图片的base64数据参数 `resultRegisterBase64`
- [新增] 视频检测配置新增是否显示人脸上方识别状态提示参数： `showFaceResultNotice`

2024-03-18 v1.2.0

- [新增] 人脸识别组件 新增接口【删除人脸】

2024-03-13 v1.1.2

- [优化] 修复人脸识别组件下同一id可以重复注册问题

2024-03-08 v1.1.1

- [优化] 【组件属性】【摄像机配置】新增属性【是否锁定屏幕启动方向】
- [优化] 【组件属性】【视频检测配置】新增属性【活体检测阈值设置】
- [修复] 修复重载之后不能注册人脸和识别
- [修复] 修复删除人脸后再去录入，用id去查询人脸，结果查不到结果

2024-03-06 v1.1.0

- [新增] 新增接口【自定义人脸数据存储路径】

2023-12-13 v1.0.1

1. 视频检测配置新增识别后是否展示面部信息
2. 更新视频检测属性新增识别后是否展示面部信息

2023-12-08

首次发布

常见问题

为什么卸载之后需要重新注册人脸

> 人脸数据默认存储在软件安装目录下，软件卸载后会删除注册的人脸数据，重新安装需要重新激活和注册，您可以通过接口自定义人脸数据存储路径，不要把人脸数据存储在安装目录下即可

为什么注册成功之后人脸识别不返回识别结果

> 如果您用的vue版本是3的话有可能会出现这个问题，您可以将vue的版本改为2

在线激活

方法名

用法

- 用法如下：

```
```

- 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

```
```

- 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

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

自定义人脸数据存储路径 v1.1.0

方法名

`customFaceDataPath` `v1.1.0`

> 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

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

- 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：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 `v1.4.3`	Object	人脸面部信息

获取图片人脸信息

参数名	参数类型	参数描述
data.list.face.bottom `v1.4.3`	Integer	面部信息底部点位
data.list.face.top `v1.4.3`	Integer	面部信息顶部点位
data.list.face.left `v1.4.3`	Integer	面部信息左边点位
data.list.face.right `v1.4.3`	Integer	面部信息右边点位
data.list.face.width `v1.4.3`	Integer	面部信息宽度
data.list.face.height `v1.4.3`	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 `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

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 `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

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，registering：注册中，registerFinish：注册完成
data.userName	String	保存的名称
data.progress	Float	当前注册的进度
data.faceId	Float	SDK保存的id
data.successCount	Integer	当前已成功注册的数量
data.imagePath	String	当前注册的人脸图片路径
data.registerTime	String	注册时间
data.userId	String	保存的id
data.failedCount	Integer	当前注册失败的数量
data.featureData	Integer	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

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 v1.4.0

方法名

`faceCompare` `v1.4.0`

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

识别组件

组件名称

`leven-arcFacePro`

> 仅支持在nvue下使用

用法

```
<leven-arcFacePro ref="refLevenArcFacePro" style="flex:1; height: 500px;" :camera
  @onCameraOpened="onCameraOpened" @onCameraClosed="onCameraClosed" @onFaceResult
</leven-arcFacePro>
```

> 组件中具体的方法、属性以及事件请参考各自文档

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

更新摄像机属性

方法名

`setCamera`

> 相机属性设置后续调用一次重载组件方法后才能生效

用法

- 用法如下：

```
if (this.$refs.refLevenArcFacePro) {
  let params = JSON.parse(JSON.stringify(this.camera));
  params.rotation = 90;
  params.facing = 1;
  params.radius = 10;
  params.size = [800, 600];
  this.$refs.refLevenArcFacePro.setCamera(params, 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
rotation	Integer	否	无	相机旋转角度
facing	Integer	否	1	相机预览模式，0:后置，1:前置(默认)
radius	Integer	否	0	组件圆角值
size	Array	否	无	预览分辨率

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

更新视频检测属性

方法名

`setVideo`

> 属性设置后续调用一次重载组件方法后才能生效

用法

- 用法如下：

```
if (this.$refs.refLevenArcFacePro) {
  let params = JSON.parse(JSON.stringify(this.video));
  params.orient = 360;
  params.liveness = false;
  params.isContraryX = true;
  params.isContraryY = true;
  params.similar = 0.95;
  params.showFaceInfo = true;
  this.$refs.refLevenArcFacePro.setVideo(params, 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
orient	Integer	否	无	视频检测角度，可接收参数，0,90,180,270,360(全方位检测，默认)
liveness	Boolean	否	true	是否进行活体检测
isContraryX	Boolean	否	false	人脸框是否处于X反向状态
isContraryY	Boolean	否	false	人脸框是否处于Y反向状态

参数名	参数类型	是否必填	默认值	参数描述
similar	Float	否	0.8	识别阈值

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

重载组件

方法名

`reInit`

用法

- 用法如下：

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

删除人脸 v1.2.0

方法名

`deleteFace`

用法

- 用法如下：

```
this.$refs.refLevenArcFacePro.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

组件属性

相机属性需要在组件主题中使用，如：

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

属性配置可以放到data中，如：

```
data() {
  return {
    //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],
    },
    // 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true
      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true
      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
    }
  }
},
```

摄像机配置

属性名称 `camera`
具体属性参数请参考【摄像机配置】

视频检测配置

属性名称 `video`
具体属性参数请参考【视频检测配置】

摄像机配置

摄像机配置参数，所有的参数都可以不传，即：该属性可以不用设置，如果不设置则按照默认处理

相机旋转角度

参数名称： `rotation`

相机预览模式

参数名称： `facing`

> 0:后置，1:前置(默认)

预览分辨率

参数名称： `size`

> 使用示例：[800, 600]，默认[1280,720]

摄像机预览圆角

参数名称： `radius`

> 默认：0

是否锁定屏幕启动方向 `v1.1.1`

参数名称： `screenLocked`

> 默认：true

组件边距 `v1.4.5`

参数名称： `padding`

摄像机配置

> 组件内部padding，参数位置：[left, top, right, bottom]，默认：边距为0

视频检测配置

视频检测配置参数，所有的参数都可以不传，即：该属性可以不用设置，如果不设置则按照默认处理

视频检测角度

参数名称： `orient`

> 可接收参数，0,90,180,270,360(全方位检测，默认)

是否进行活体检测

参数名称： `liveness`

> 默认为true

人脸框是否处于X反向状态

参数名称： `isContraryX`

> 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

人脸框是否处于Y反向状态

参数名称： `isContraryY`

> 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true

识别阈值

参数名称： `similar`

> 默认：0.8

识别是否展示面部信息 `v1.0.1`

参数名称： `showFaceInfo`

> 默认：true，面部信息包括（是否戴眼镜，双眼是否闭合，是否张嘴等）

活体检测阈值设置 `v1.1.1`

参数名称： `livenessParams`
参数示例：

```
//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,
    // 活体 FQ 检测阈值，默认：0.65
    fq: 0.65
  }
```

> rgb：可见光活体检测阈值，默认：0.5
> ir：红外活体检测阈值，默认：0.7
> fq：活体 FQ 检测阈值，默认：0.65

识别结果是否返回识别注册图片的base64数据

参数名称： `resultRegisterBase64`

> 可能影响性能，默认：false

是否显示人脸上方识别状态提示

参数名称： `showFaceResultNotice`

> 默认：true

是否显示人脸上方识别状态提示 `v1.4.2`

参数名称： `faceSize`

> 默认：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

组件事件

错误事件

事件名称

`onError`

用法

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

返回示例

```
onError(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
code	Integer	虹软的错误码

相机打开事件

事件名称

`onCameraOpened`

用法

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

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

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-arcFacePro ref="refLevenArcFacePro" style="flex:1; height: 500px;"  @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

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
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 `v1.3.0`	String	识别后的头像图片base64格式
imageBase64 `v1.4.0`	String	人脸图片base64数据
faseSize `v1.4.2`	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 `v1.4.0`	String	当前识别的人脸图片
featureData	String	识别的人脸特征，base64加密