**人脸识别能力API**

## 人脸检测

### 1.1接口功能

①人脸位置检测：检测图像中人脸的位置并输出人脸位置矩形框及其置信度

②人脸关键点：检测人脸的72个关键点信息并输出点的坐标及其置信度

③人脸属性：检测并输出人脸属性（年龄、性别、眼镜、帽子）并输出每一项属性的置信度

④人脸质量：输出人脸质量信息，目前人脸质量，包括整体人脸、整体正脸、局部人脸、pitch、roll、yaw

⑥输出人脸数量

⑦输出人脸唯一标识faceToken

### 1**.2请求说明**

①HTTP方法:POST

②请求体格式化：Content-Type为application/json，通过json格式化请求体

③图片格式：目前只支持jpg格式图片，图像需经过Base64编码，编码后的数据不包含图片头（data:image/jpeg;base64）

④请求URL：<http://ip:port/v1/Detect>

⑤请求Body

{

"images":["asdasajshajkskashdjkqiweuijkadfmn"], //编码后的图像数据

"facefield": {

"landmark": 0, //是否输出72个关键点信息，默认为0，不输出

"quality": 0, //是否输出人脸质量信息，默认为0，不输出

"attributes": 0 //是否输出人脸属性信息，默认为0，不输出

}

}

备注：人脸位置为默认输出项，不支持选择。人脸检测算法检测人脸属性、人脸质量、的时候需要依赖关键点信息，因此如果要检测人脸属性、人脸质量中的一项或多项，必须将关键点检测项置位。

### 1.3返回说明

返回值为json字符串，具体说明如下

{

"errCode": 0, //请求错错误码

"errMsg": "正确", //请求返回信息

"detectInfo": {

"faceNum": 1, //人脸数量

"faces": [{ //人脸详细信息

"box": { //人脸位置矩形框

"center": { //矩形框中心点

"coordinate\_x": 125.96384, //中心点x坐标

"coordinate\_y": 114.86412 //中心点y坐标

},

"rect": {

"width": 87.47117, //矩形框宽度

"height": 80.88723 //矩形框高度

},

"score": 0.9999219 //人脸位置置信度

},

"landmark": { //72个关键点信息

"points": [

{

"coordinate\_x": 227.15843, //每个关键点的x坐标

"coordinate\_y": 243.57726 //每个关键点的y坐标

}......

],

"score": 71.796104 //关键点置信度

},

"attributes": {

"age": 36, //年龄

"age\_score": 100, //年龄置信度

"gender": 0, //性别，0男，1女

"gender\_score": 98, //性别置信度

"glass": 0, //是否戴眼镜

"glass\_score": 99, //置信度

"hat": 0, //是否戴帽子

"hat\_score": 99, //置信度

},

"pitch": 0, //俯仰角

"yaw": 0, //左右旋转角

"roll": 0, //平面内旋角

"global\_is\_face": 0, //整体人脸

"global\_front\_face": 0, //整体正脸

"local\_is\_face": 0, //局部人脸

"faceToken": "dac24db0438e42f694a823273eb01514" //人脸唯一标识

}]}

}

### 1.4请求及返回示例

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 服务或资源URI | URI | /v1/Detect | | | | | | | | | | |
| 参数 |  | | | | | | | | | | |
| HTTP Method | POST | | | | | | | | | | | |
| Header |  | | | | | | | | | | | |
| Body | Type | String body（Json） | | | | | | | | | | |
| 参数 | 键 | | | | | | | 类型 | | 描述 | |
| images | | | | | | | STRING | | 图像Base64编码后的字符串数组，不包含数据头（data:image/jpeg;） | |
| face\_field | | | | | | | OBJ | | 人脸检测项开关0:是，1:否,默认为0，默认输出人脸框位置 | |
|  | landmark | | | | | | INT | | 是否输出72个关键点信息， | |
| attribute | | | | | | INT | | 是否输出属性信息，属性包括年龄、性别、人种、眼镜、帽子等 | |
| quality | | | | | | INT | | 是否输出人脸质量，包括遮挡、模糊、完整度、置信度等 | |
| Example | {  "images:["asdasdasdasd”],  "facefield": {  "landmark": 1,  "quality": 0,  "attributes": 0  }  } | | | | | | | | | | |
| Return | Type | String body（Json） | | | | | | | | | | |
| 数据 | 键 | | | | | | | | 类型 | | 描述 |
| faces | | | | | | | | OBJ | | 识别到的人脸 |
|  | | Box | | | | | | OBJ | | 人脸框 |
|  | center | | | | | OBJ | | 人脸框中心点 |
|  | | | coordinate\_x | | FLOAT | | 中心点x坐标 |
| coordinate\_y | | FLOAT | | 中心点y坐标 |
| rect | | | | | OBJ | | 矩形框 |
|  | | | width | | FLOAT | | 宽 |
| height | | FLOAT | | 高 |
| score | | | | | FLOAT | | 人脸框置信度 |
| landmark | | | | | | OBJ | | 72个关键点 |
|  | | points | | | | OBJ | | 关键点信息 |
|  | | coordinate\_x | | FLOAT | | 关键点x坐标 |
|  | | coordinate\_y | | FLOAT | | 关键点y坐标 |
| score | | | | FLOAT | | 关键点置信度 |
| feature | | | | | | OBJ | | 192维特征点 |
|  | | values | | | | FLOAT | | 每一维特征点的值 |
| attributes | | | | | | OBJ | | 人脸属性 |
|  | | | age | | | FLOAT | | 年龄 |
| age\_score | | | FLOAT | | 年龄置信度 |
| gender | | | FLAOT | | 性别 |
| gender\_score | | | FLOAT | | 性别置信度 |
| glass | | | FLOAT | | 眼镜 |
| glass\_score | | | FLOAT | | 眼镜置信度 |
| hat | | | FLOAT | | 帽子 |
| hat\_score | | | FLOAT | | 帽子置信度 |
| pitch | | | | | | FLOAT | | 俯仰角 |
| yaw | | | | | | FLOAT | | 左右旋转角 |
| roll | | | | | | FLOAT | | 平面内旋角 |
| global\_is\_face | | | | | | FLOAT | | 是否是完整人脸 |
| global\_front\_face | | | | | | FLOAT | | 是否是正脸 |
| local\_is\_face | | | | | | FLOAT | | 局部人脸 |
| faceToken | | | | | | STRING | | 人脸唯一标识 |
| Example | {"faces": [{  "box": {  "center": {  "coordinate\_x": 356.06494,  "coordinate\_y": 314.4172  },  "rect": {  "width": 277.54434,  "height": 236.06032  },  "score": 0.99955004  },  "landmark": {  "points": [  {  "coordinate\_x": 227.15843,  "coordinate\_y": 243.57726  }......  ],  "score": 71.796104  },  "feature": {"values": [  0.04743944,.....  ]},      "attributes": {  "age": 36,  "age\_score": 100,  "gender": 0,  "gender\_score": 98,  "glass": 0,  "glass\_score": 99,  "hat": 0,  "hat\_score": 99  },  "pitch": -3.3769498,  "yaw": 10.513793,  "roll": -1.5962598,  "global\_is\_face": 0.99973077,  "global\_front\_face": 0.999106,  "local\_is\_face": 0.9916507，  "faceToken": " dac24db0438e42f694a823273eb01514"  }]} | | | | | | | | | | |

## 人脸对比

### 接口功能

①接口能力：比对两张图片中人脸的相似度并返回比分

②如果图像中有多张人脸，报错

### 2.2 请求说明

①HTTP方法:POST

②请求体格式化：Content-Type为application/json，通过json格式化请求体

③图片格式：目前只支持jpg格式图片，图像需经过Base64编码，编码后的数据不包含图片头（data:image/jpeg;base64）

④请求URL：http://ip:port/v1/Match

⑤请求Body

{

"image1":"qaweqawfewfawfawweqweasdfasdfwfqe", // 第一张图像数据

"image2":"qweqweajksdjhdsadfsadfsdfasdfasdjk" //第二张图像数据

}

### 2.3 返回说明

正确请求后会返回json字符串，具体说明如下：

{

"errCode": 0, //请求返回的错误码

"errMsg": “正确", //请求返回的信息

"score": 1 //两张人脸比分

}

### 2.4 请求及返回示例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 服务或资源URI | URI | /v1/Match | | |
| 参数 |  | | |
| HTTP Method | POST | | | |
| Header |  | | | |
| Body | Type | String body（Json） | | |
| 参数 | 键 | 类型 | 描述 |
| image1 | STRING | 第一张图像数据 |
| image2 | STRING | 第二张图像数据 |
| Example | {  "image1":"qaweqawfewfawfawweqweasdfasdfwfqe", "image2":"qweqweajksdjhdsadfsadfsdfasdfasdjk"  } | | |
| Return | Type | String body（Json） | | |
| 数据 | 键 | 类型 | 描述 |
| errCode | INT | [请求返回错误码](#_5.错误码) |
| errMsg | STRING | 请求返回错误信息 |
| score | FLOAT | 返回的比分 |
| Example | {  errCode": 0, //请求返回的错误码  "errMsg": “正确", //请求返回的信息  "score": 1 //两张人脸比分  } | | |

## 3.人脸搜索

### 3.1 1:N 人脸搜索

#### 3.1.1 接口功能

①在指定人脸库中进行检索，找到最相似的人脸

②返回最相似人脸的唯一标识和人脸图像的唯一标识

#### 3.1.2请求说明

①HTTP方法:POST

②请求体格式化：Content-Type为application/json，通过json格式化请求体

③图片格式：目前只支持jpg格式图片，图像需经过Base64编码，编码后的数据不包含图片头（data:image/jpeg;base64）

④请求URL：http://ip:port/v1/Scan

⑤请求Body

{

"data":"qaweqawfewfawfawweqweasdfasdfwfqe", // 图像数据

}

备注：目前实现的功能不包含用户组、用户的信息，请求只需传入图像数据

#### 3.1.3 返回说明

请求成功返回json字符串，详细信息如下

{

"errCode": 0, //错误码

"errMsg": "正确", //错误信息

"score": 0.97848886, //相似度

"faceToken": "2e7b2cac4948a2353d3360f0" //人脸唯一标示

"imgToken": "2e7b2cac4948a2353d3360f0" //人脸图像唯一标识

"userID":"zhangsan" //用户ID

"userInfo": "zhangsan" //用户信息

}

#### 3.1.4 请求及返回示例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 服务或资源URI | URI | /v1/Scan | | |
| 参数 |  | | |
| HTTP Method | POST | | | |
| Header |  | | | |
| Body | Type | String body（Json） | | |
| 参数 | 键 | 类型 | 描述 |
| data | STRING | 图像数据 |
|
| Example | {  "data":"qaweqawfewfawfawweqweasdfasdfwfqe"  } | | |
| Return | Type | String body（Json） | | |
| 数据 | 键 | 类型 | 描述 |
| errCode | INT | [请求返回错误码](#_5.错误码) |
| errMsg | STRING | 请求返回错误信息 |
| score | FLOAT | 返回的比分 |
| imgToken | STRING | 图像唯一标识 |
| faceToken | STRING | 人脸唯一标识 |
| userID | STRING | 用户ID |
| userInfo | STRING | 用户信息 |
| Example | {  errCode": 0, //请求返回的错误码  "errMsg": “正确", //请求返回的信息  "score": 1, //相似度  "imgToken":"2353d3360f6258c3c9116325a1791" //图像唯一标识  “faceToken”:” 2e7b2cac4948a2353d3360f6258c3c9116325a179143c”  } | | |

### 3.2 M:N 人脸搜索

M：N识别的原理，相当于在多个人脸的图片中，先分别找出所有人脸，然后分别在待查找的人脸集合中，分别做1：N识别，最后将识别结果汇总在一起进行返回。

#### 3.2.1接口功能

①待识别的人脸图像中含有多张人脸，通过一次请求，同时返回图片中每张人脸最相似的人脸图像唯一标识及人脸唯一标识

#### 3.2.2 请求说明

①HTTP方法:POST

②请求体格式化：Content-Type为application/json，通过json格式化请求体

③图片格式：目前只支持jpg格式图片，图像需经过Base64编码，编码后的数据不包含图片头（data:image/jpeg;base64）

④请求URL：http://ip:port/v1/ScanEx

⑤请求Body

{

"data":"qaweqawfewfawfawweqweasdfasdfwfqe", // 图像数据

}

#### 3.2.3 返回说明

请求成功后返回json字符串，详细信息如下

{

"errCode": 0, //请求返回错误码

"errMsg": "正确", //请求返回的错误信息

"scanResult": { // 人脸搜索结果

"faceNum": 2, //人脸数量

"faceList": [

{

"faceToken": "268b9b3a12cca32ff48bc7", //人脸唯一标识

"score": 0.7712203, //人脸比分

"imgToken": "268b9b3a12cca32ff48bc7" //人脸图像唯一标识

},

{

"faceToken": "4079022794289ecce1716cc", //人脸唯一标识

"score": 0.871755, //人脸比分

"imgToken": "268b9b3a12cca32ff48bc7" //人脸图像唯一标识

}

]

}

}

#### 3.2.4 请求及返回示例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 服务或资源URI | URI | /v1/ScanEx | | | | | | |
| 参数 |  | | | | | | |
| HTTP Method | POST | | | | | | | |
| Header |  | | | | | | | |
| Body | Type | String body（Json） | | | | | | |
| 参数 | 键 | | | 类型 | | 描述 | |
| data | | | STRING | | 第一张图像数据 | |
| Example | {  "data":"qaweqawfewfawfawweqweasdfasdfwfqe"  } | | | | | | |
| Return | Type | String body（Json） | | | | | | |
| 数据 | 键 | | | | 类型 | | 描述 |
| errCode | | | | INT | | [请求返回错误码](#_5.错误码) |
| errMsg | | | | STRING | | 请求返回错误信息 |
| scanResult | | | | OBJ | | 返回搜索结果 |
|  | faceNum | | | INT | | 图像中人脸数量 |
| faceList | | | OBJ | | 每张人脸的比对结果（数组） |
|  |  | faceToken | | STRING | | 人脸唯一标识 |
| score | | FLOAT | | 比分 |
| imgToken | | STRING | | 图像唯一标识 |
| Example | {  "errCode": 0, //请求返回错误码  "errMsg": "正确", //请求返回的错误信息  "scanResult": { // 人脸搜索结果  "faceNum": 2, //人脸数量  "faceList": [  {  "faceToken":"268b9b3a12cca32ff48bc7", "score": 0.7712203,  "imgToken": "268b9b3a12cca32ff48bc7"  },  {  "faceToken": "4079022794289ecce1716cc",  "score": 0.871755,  "imgToken": "268b9b3a12cca32ff48bc7"  }  ]  }  } | | | | | | |

## 4.人脸库管理

### 4.1人脸注册

#### 4.1.1 接口功能

①向人脸库中添加一张人脸

②返回图像唯一标识和人脸唯一标识

#### 4.1.2 请求说明

①HTTP方法:POST

②请求体格式化：Content-Type为application/json，通过json格式化请求体

③图片格式：目前只支持jpg格式图片，图像需经过Base64编码，编码后的数据不包含图片头（data:image/jpeg;base64）

④请求URL：http://ip:port/v1/AddFace

⑤请求Body

{

"data":"qaweqawfewfawfawweqweasdfasdfwfqe", // 图像数据

"userID":"", //用户ID

"userInfo":””, //用户信息

}

#### 4.1.3 返回说明

请求成功时返回json字符串，详细信息如下

{

"errCode": 0, //请求错误码

"errMsg": "正确", //请求错误信息

"location": { //人脸位置

"center": { //人脸位置矩形框中心点

"coordinate\_x": 213.6981, //中心点x坐标

"coordinate\_y": 142.13022 //中心点y坐标

},

"rect": { //人脸位置矩形框

"width": 109.8787, //矩形框的宽

"height": 119.309494 //矩形框的高

},

"score": 0.9999491 //人脸位置置信度

},

"faceToken":"4031fe05ebb4bfc000c157e9e6db" //人脸唯一标示

"imgToken":"4031fe05ebb4bfc000c157e9e6db " //图像唯一标识

}

#### 4.1.4 请求及返回示例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 服务或资源URI | URI | /v1/AddFace | | | | | | |
| 参数 |  | | | | | | |
| HTTP Method | POST | | | | | | | |
| Header |  | | | | | | | |
| Body | Type | String body（Json） | | | | | | |
| 参数 | 键 | | | 类型 | | 描述 | |
| data | | | STRING | | base64编码后的图像数据 | |
| userID | | | STRING | | 用户唯一标示 | |
| userInfo | | | STRING | | 用户信息 | |
| Example | {  "data":"qaweqawfewfawfawweqweasdfasdfwfqe", // 图像数据  "userID":"", //用户ID  "userInfo":””, //用户信息  } | | | | | | |
| Return | Type | String body（Json） | | | | | | |
| 数据 | 键 | | | | 类型 | | 描述 |
| errCode | | | | INT | | [错误码](#_5.错误码) |
| errMsg | | | | STRING | | 错误信息 |
| faceToken | | | | STRING | | 人脸唯一标示 |
| imgToken | | | | STRING | | 图像唯一标识 |
| location | | | | OBJ | | 人脸位置 |
|  | center | | | OBJ | | 人脸位置中心点 |
|  | coordinate\_x | | FLOAT | | 中心点x坐标 |
| coordinate\_y | | FLOAT | | 中心点y坐标 |
|  | rect | | | OBJ | | 人脸矩形框 |
|  |  | width | | FLOAT | | 矩形的高 |
|  | height | | FLOAT | | 矩形的宽 |
| Example | {  "errCode": 0, //请求错误码  "errMsg": "正确", //请求错误信息  "location": { //人脸位置  "center": { //人脸位置矩形框中心点  "coordinate\_x": 213.6981, //中心点x坐标  "coordinate\_y": 142.13022 //中心点y坐标  },  "rect": { //人脸位置矩形框  "width": 109.8787, //矩形框的宽  "height": 119.309494 //矩形框的高  },  "score": 0.9999491 //人脸位置置信度  },  "faceToken":"4031fe05ebb4bfc000c157e9e6db",  "imgToken":"4031fe05ebb4bfc00e6db "  } | | | | | | |

### 4.2人脸更新

#### 4.2.1 接口功能

①更新人脸库中指定用户的人脸图像，新上传的人脸将覆盖此用户下的所有原有图像

#### 4.2.2 请求说明

①HTTP方法:POST

②请求体格式化：Content-Type为application/json，通过json格式化请求体

③请求URL：http://ip:port/v1/UpdateFace

⑤请求Body

{

"data":"ajkahsower92354lkja2q834hjkasnxc89" //图像数据

"userID"："zhangsan" //用户ID

"userInfo":"" //用户信息

}

#### 4.2.3 返回说明

请求成功时返回json字符串，详细信息如下

{

"errCode": 0, //请求错误码

"errMsg": "正确", //请求错误信息

"location": { //人脸位置

"center": { //人脸位置矩形框中心点

"coordinate\_x": 213.6981, //中心点x坐标

"coordinate\_y": 142.13022 //中心点y坐标

},

"rect": { //人脸位置矩形框

"width": 109.8787, //矩形框的宽

"height": 119.309494 //矩形框的高

},

"score": 0.9999491 //人脸位置置信度

},

"faceToken":"4031fe05ebb4bfc000c157e9e6db" //人脸唯一标示

"imgToken":"4031fe05ebb4bfc000c157e9e6db " //图像唯一标识

}

#### 4.2.4 请求及返回示例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 服务或资源URI | URI | /v1/UpdateFace | | | | | | |
| 参数 |  | | | | | | |
| HTTP Method | POST | | | | | | | |
| Header |  | | | | | | | |
| Body | Type | String body（Json） | | | | | | |
| 参数 | 键 | | | 类型 | | 描述 | |
| data | | | STRING | | base64编码后的图像数据 | |
| userID | | | STRING | | 用户唯一标示 | |
| userInfo | | | STRING | | 用户信息 | |
| Example | {  "data":"qaweqawfewfawfawweqweasdfasdfwfqe", // 图像数据  "userID":"zahngsan", //用户ID  "userInfo":”zhangsan”, //用户信息  } | | | | | | |
| Return | Type | String body（Json） | | | | | | |
| 数据 | 键 | | | | 类型 | | 描述 |
| errCode | | | | INT | | [错误码](#_5.错误码) |
| errMsg | | | | STRING | | 错误信息 |
| faceToken | | | | STRING | | 人脸唯一标示 |
| imgToken | | | | STRING | | 图像唯一标识 |
| location | | | | OBJ | | 人脸位置 |
|  | center | | | OBJ | | 人脸位置中心点 |
|  | coordinate\_x | | FLOAT | | 中心点x坐标 |
| coordinate\_y | | FLOAT | | 中心点y坐标 |
|  | rect | | | OBJ | | 人脸矩形框 |
|  |  | width | | FLOAT | | 矩形的高 |
|  | height | | FLOAT | | 矩形的宽 |
| Example | {  "errCode": 0, //请求错误码  "errMsg": "正确", //请求错误信息  "location": { //人脸位置  "center": { //人脸位置矩形框中心点  "coordinate\_x": 213.6981, //中心点x坐标  "coordinate\_y": 142.13022 //中心点y坐标  },  "rect": { //人脸位置矩形框  "width": 109.8787, //矩形框的宽  "height": 119.309494 //矩形框的高  },  "score": 0.9999491 //人脸位置置信度  },  "faceToken":"4031fe05ebb4bfc000c157e9e6db",  "imgToken":"4031fe05ebb4bfc00e6db "  } | | | | | | |

### 4.3.获取用户人脸列表

#### 4.3.1 接口功能

①获取人脸库中指定用户的人脸列表

#### 4.3.2 请求说明

①HTTP方法:POST

②请求体格式化：Content-Type为application/json，通过json格式化请求体

③请求URL：http://ip:port/v1/ListFace

⑤请求Body

{

"userID"："zhangsan" //用户ID

}

#### 4.3.3 返回说明

请求成功返回json字符串，详细信息如下：

{

"errCode": 0, //请求返回错误码

"errMsg": "正确", //请求返回错误信息

"faceRecord": [ //返回查找记录

{

"createTime": "1541389749", //创建时间

"updateTime": "1541389749", //修改时间

"faceToken": "a1e2157487054ee489bc72cc4e0cbbac" //人脸唯一标识

"imgToken": " a1e2157487054ee489bc72cc4e0cbbac " //图像唯一标识

}

]

}

#### 4.3.4 请求及返回示例

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 服务或资源URI | URI | /v1/ListFace | | | | | |
| 参数 |  | | | | | |
| HTTP Method | POST | | | | | | |
| Header |  | | | | | | |
| Body | Type | String body（Json） | | | | | |
| 参数 | 键 | | 类型 | | 描述 | |
| userID | | STRING | | 用户ID | |
| Example | {  "userID"："zhangsan" //用户ID  } | | | | | |
| Return | Type | String body（Json） | | | | | |
| 数据 | 键 | | | 类型 | | 描述 |
| errCode | | | INT | | [请求返回错误码](#_5.错误码) |
| errMsg | | | STRING | | 请求返回错误信息 |
| faceList | | | OBJ | | 返回的人脸列表 |
|  | imgToken | | STRING | | 图像唯一标识 |
| faceToken | | STRING | | 人脸唯一标识 |
| createTime | | INT64 | | 创建时间 |
| updateTime | | INT64 | | 修改时间 |
| Example | {  "errCode": 0, //请求返回错误码  "errMsg": "正确", //请求返回错误信息  "faceRecord": [ //返回查找记录  {  "createTime": "1541389749", //创建时间  "updateTime": "1541389749", //修改时间  "imgToken": " a1e2157487054ee489bc72cc4e0cbbac ", //图像唯一标识  "faceToken": "a1e2157487054ee489bc72cc4e0cbbac" //人脸唯一标识  }  ]  } | | | | | |

### 4.4 删除人脸

#### 4.4.1 接口功能

删除指定用户的指定的脸照片及数据库记录

#### 4.4.2 请求说明

①HTTP方法:POST

②请求体格式化：Content-Type为application/json，通过json格式化请求体

③请求URL：http://ip:port/v1/DelFace

⑤请求Body

{

"faceToken"："487054ee489bc72cc4e ", //人脸的唯一标识

"userID"："zhangshan" //用户ID

}

#### 4.4.3 返回说明

请求成功返回json字符串，详细信息如下

{

"errCode"：0, //请求错误码

"errMsg": "正确" //请求错误信息

}

#### 4.4.4 请求及返回示例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 服务或资源URI | URI | /v1/DelFace | | |
| 参数 |  | | |
| HTTP Method | POST | | | |
| Header |  | | | |
| Body | Type | String body（Json） | | |
| 参数 | 键 | 类型 | 描述 |
| userID | STRING | 用户ID |
| faceToken | STRING | 人脸唯一标识 |
| Example | {  "faceToken":"qaweqawfewfawfawweqweasdfasdfwfqe"，  "userID":"zhangshan"  } | | |
| Return | Type | String body（Json） | | |
| 数据 | 键 | 类型 | 描述 |
| errCode | INT | [请求返回错误码](#_5.错误码) |
| errMsg | STRING | 请求返回错误信息 |
| Example | {  errCode": 0, //请求返回的错误码  "errMsg": “正确", //请求返回的信息  } | | |

### 4.5 获取用户列表

#### 4.5.1 接口功能

①获取人脸库中用户列表

#### 4.5.2 请求说明

①HTTP方法:GET

②请求URL：http://ip:port/v1/ListUser

#### 4.5.3 返回说明

请求成功返回json字符串，详细信息如下：

{

"errCode": 0, //错误码

"errMsg": "正确", //错误信息

"userList": [ // 用户列表

"zhangsan",

"lisi" ]

}

#### 4.5.4 请求及返回示例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 服务或资源URI | URI | /v1/ListUser | | |
| 参数 |  | | |
| HTTP Method | GET | | | |
| Header |  | | | |
| Return | Type | String body（Json） | | |
| 数据 | 键 | 类型 | 描述 |
| errCode | INT | [请求返回错误码](#_5.错误码) |
| errMsg | STRING | 请求返回错误信息 |
| faceList | STRING | 返回的人脸列表 |
| Example | {  "errCode": 0, //错误码  "errMsg": "正确", //错误信息  "userList": [ // 用户列表  "zhangsan",  "lisi" ]  } | | |

### 4.6 查询指定用户信息

#### 4.6.1 接口功能

①查询人脸库中指定用户的用户信息

#### 4.6.2 请求说明

①HTTP方法:POST

②请求体格式化：Content-Type为application/json，通过json格式化请求体

③请求URL：http://ip:port/v1/GetUserInfo

⑤请求Body

{

"userID"："zhangshan" //用户ID

}

#### 4.6.3 返回说明

请求成功返回json字符串，详细信息如下

{

"errCode"：0, //请求错误码

"errMsg": "正确" //请求错误信息

"userInfo": "zhangsan" //用户信息

}

#### 4.6.4 请求及返回示例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 服务或资源URI | URI | /v1/ GetUserInfo | | |
| 参数 |  | | |
| HTTP Method | POST | | | |
| Header |  | | | |
| Body | Type | String body（Json） | | |
| 参数 | 键 | 类型 | 描述 |
| userID | STRING | 用户ID |
| Example | {  "userID":"zhangshan" //userID  } | | |
| Return | Type | String body（Json） | | |
| 数据 | 键 | 类型 | 描述 |
| errCode | INT | [请求返回错误码](#_5.错误码) |
| errMsg | STRING | 请求返回错误信息 |
| userInfo | STRING | 用户信息 |
| Example | {  "errCode": 0, //请求返回的错误码  "errMsg": “正确", //请求返回的信息  "userInfo":"zhangsan" //用户信息  } | | |

### 4.7 删除用户

#### 4.7.1 接口功能

①删除指定用户

#### 4.7.2 请求说明

①HTTP方法:POST

②请求体格式化：Content-Type为application/json，通过json格式化请求体

③请求URL：http://ip:port/v1/DelUser

⑤请求Body

{

"userID"："zhangsan" //用户ID

}

#### 4.7.3 返回说明

请求成功返回json字符串，详细信息如下

{

"errCode"：0, //请求错误码

"errMsg": "正确" //请求错误信息

}

#### 4.7.4 请求及返回示例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 服务或资源URI | URI | /v1/DelUser | | |
| 参数 |  | | |
| HTTP Method | POST | | | |
| Header |  | | | |
| Body | Type | String body（Json） | | |
| 参数 | 键 | 类型 | 描述 |
| userID | STRING | 用户ID |
| Example | {  "userID":"zhangshan"  } | | |
| Return | Type | String body（Json） | | |
| 数据 | 键 | 类型 | 描述 |
| errCode | INT | [请求返回错误码](#_5.错误码) |
| errMsg | STRING | 请求返回错误信息 |
| Example | {  errCode": 0, //请求返回的错误码  "errMsg": “正确", //请求返回的信息  } | | |

## 5.错误码

//xgface相关

  ERR\_OK = 0 //正确

    ERR\_INVALID\_PARAM = -10001 //参数错误

    ERR\_CONNECT\_FAILED = -10002 //连接错误

    ERR\_ALG\_FAILED = -10003 //算法返回出错

    ERR\_DETECT\_FAILED = -10004 //人脸检测失败

    ERR\_DETECT\_NO\_FACE = -10005 //没有检测到人脸

    ERR\_MULTI\_FACE = -10006 //检测到多张人脸

    //人脸库(mysql)相关

    ERR\_CONN\_MYSQL\_FAILED = -11001 //连接数据库失败

    ERR\_EXIST = -11002 //该人脸已存在

    ERR\_QUERY\_FAILED = -11003 //检索出错

    ERR\_GET\_PATH\_FAILED = -11004 //获取路径时出错

    ERR\_CREATE\_PATH\_FAILED = -11005 //创建路径时出错

    ERR\_CREATE\_FILE\_FAILED = -11006 //创建文件失败

    ERR\_INSERT\_FAILED = -11007 //插入记录失败

    ERR\_ENCODE\_FAILED = -11008 //base64解码失败

    ERR\_DEL\_FILE\_FAILED = -11009 //删除文件失败

    ERR\_DEL\_RECORD\_FAILED = -11010 //删除记录失败

    //xgindex相关

    ERR\_CONN\_INDEX\_FAILED = -12001 //连接xgindex失败

    ERR\_SIMIARITY\_FAILED = -12002 //获取相似度失败

    ERR\_INSERT\_INDEX\_FAILED = -12003 //插入index获取clusterID

    ERR\_DEL\_CLUSTER\_FAILED = -12004 //删除cluster失败