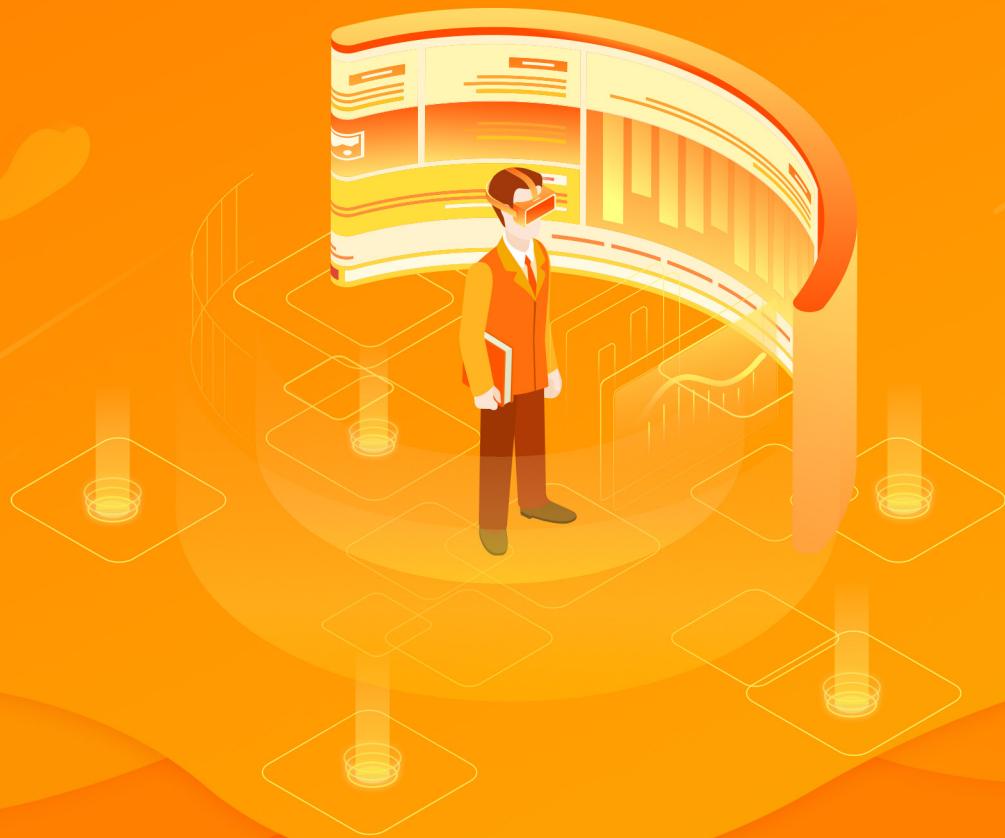


# 15分钟开发视觉AI应用

视觉AI应用开发从入门到精通



零门槛搞定人脸识别 / 口罩识别 / 图片内容安全应用  
三大实战案例教学快速入门视觉AI应用开发





阿里云开发者“藏经阁”  
海量免费电子书下载



加入钉群  
获取更多资讯

# 15分钟开发视觉AI应用

**1**

视觉AI应用开发从入门到精通  
**视觉开放平台简介**

**3** 简介

**4**

视觉AI应用开发从入门到精通  
**口罩识别实战教程**

**20** 前提条件

**21** 口罩识别

**2**

视觉AI应用开发从入门到精通  
**开发前准备**

**4** 步骤一：开通阿里云视觉智能开放平台  
**6** 步骤二：开通OSS服务  
**6** 步骤三：创建密钥并安装Java SDK

**5**

视觉AI应用开发从入门到精通  
**图片内容安全实战教程**

**24** 背景信息  
**24** 前提条件  
**25** 图片内容安全

**3**

视觉AI应用开发从入门到精通  
**人脸识别实战教程**

**8** 背景信息                   **12** 人体计数  
**8** 前提条件                   **14** 人脸搜索  
**9** 人脸属性识别

# 1

15分钟开发视觉AI应用 ◎

## 视觉开放平台简介

### 概述

据不完全统计，整个阿里巴巴集团有数千名开发人员围绕着视觉技术在电子商务、城市大脑、金融支付、交通物流、通信会议、新零售、文娱等多个行业的应用需求，不断贡献着各类技术创新与应用实践，形成了多个产品和解决方案。这其中沉淀了诸多视觉基础原子算法，如何将这些算法拿来服务更广泛的用户和开发者群体，发挥更大的价值呢？由阿里巴巴集团技术委员会视觉技术小组和战略合作部牵头，达摩院联合阿里云产品与解决方案管理部以及集团各个视觉技术团队一起创建了[阿里云视觉智能开放平台](#)。

阿里云视觉智能开放平台是基于阿里巴巴视觉智能技术实践经验，面向视觉智能技术企业和开发商（含开发者），为其提供高易用、普惠的视觉API服务，帮助企业快速建立视觉智能技术的应用能力的综合性视觉AI能力平台。

阿里云视觉智能开放平台将围绕多个视觉领域，例如：通用、图像、视频、目标识别以及3D、AR/VR等类目，不断的为您提供100多种视觉AI算法能力。

# 2 15分钟开发视觉AI应用 ◎ 开发前准备

## 概述

阿里云视觉智能开放平台的官网地址为 [vision.aliyun.com](https://vision.aliyun.com)。平台会为您提供普惠易用的 AI 能力。适用于城市大脑、安防、数字营销、泛金融身份认证、互联网娱乐、手机应用等行业，企业和开发商（含开发者）可以选择相应能力自行封装解决方案或者是产品、服务。

### 1 步骤一：开通阿里云视觉智能开放平台



1. 打开[阿里云视觉智能开放平台](https://vision.aliyun.com)，单击登录。

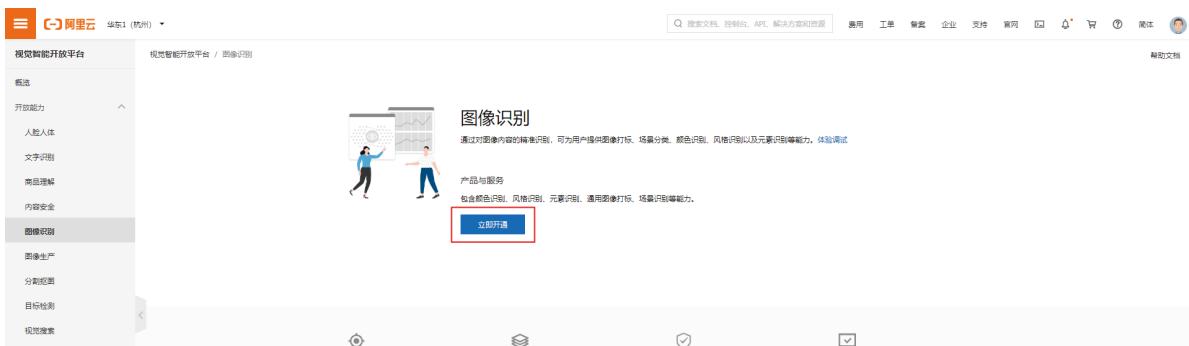


2. 选择相应能力类目，单击立即开通。例如：开通图像识别服务。有以下两个能力开通入口：

在首页能力广场下拉列表中单击选择图像识别 > 车型识别（任选某一个算法能力），进入能力详情页，然后单击立即开通。

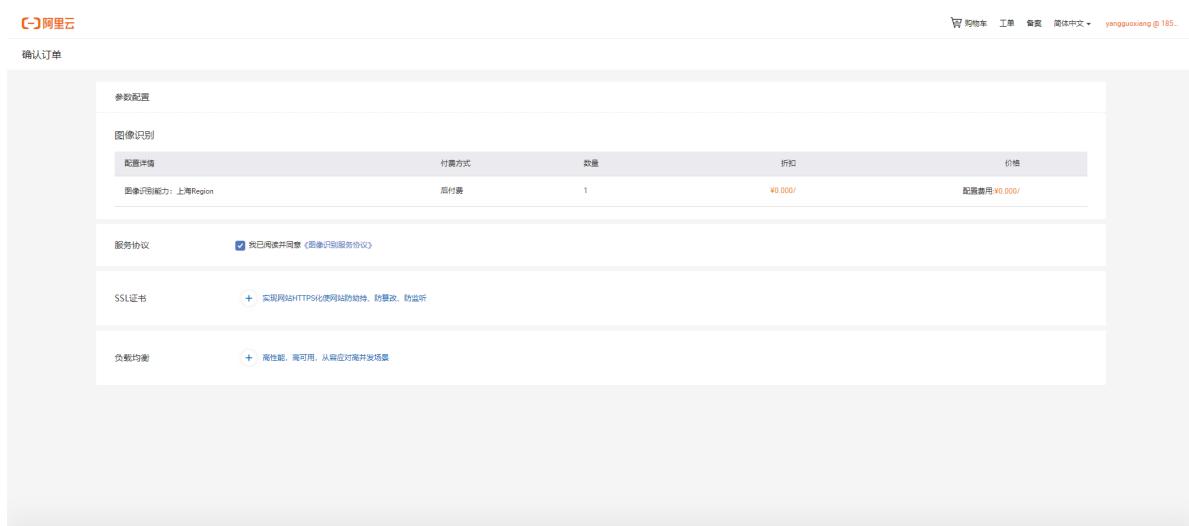


在[视觉智能开放平台](#)管理控制台的左侧边栏选择[图像识别](#)然后单击[立即开通](#)。



3. 确认开通服务的地域后单击[立即购买](#)。

4. 在确定订单步骤中, 勾选[我已阅读并同意《图像识别服务协议》](#)后, 单击去支付。



## 2 步骤二：开通 OSS 服务

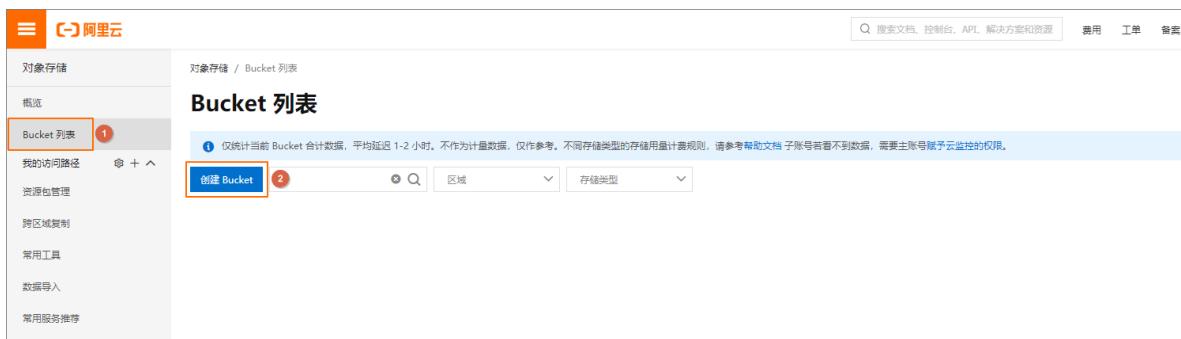


阿里云对象存储服务（Object Storage Service，简称 OSS）为您提供基于网络的数据存取服务。使用 OSS，您可以通过网络随时存储和调用包括文本、图片、音频和视频等在内的各种非结构化数据文件。

1. 登录[阿里云官网](#)。
2. 将鼠标移至产品，单击[对象存储 OSS](#)，打开 OSS 产品详情页面。
3. 在[OSS 产品详情页](#)，单击[立即开通](#)。



4. 登录[OSS 管理控制台](#)。
5. 单击[Bucket 列表](#)，之后单击[创建 Bucket](#)。



6. 在[创建 Bucket](#)对话框配置 Bucket 参数，配置完成后单击[确定](#)。

## 3 步骤三：创建密钥并安装Java SDK



1. 创建 AccessKey。
  - a. 登录阿里云管理控制台。
  - b. 访问[AccessKey 管理中心控制台](#)。
  - c. 创建和管理您的 AccessKey。

## 2. 安装 Java SDK。

在您的 Java 工程中添加以下 Pom 依赖。

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-core</artifactId>
  <version>4.4.8</version>
</dependency>
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>fastjson</artifactId>
  <version>1.2.52</version>
</dependency>
```

# 3 15分钟开发视觉AI应用 ◎ 人脸识别实战教程

## 概述

本教程介绍如何使用 Alibaba Cloud SDK for Java 进行人体属性识别、人体计数和人脸搜索任务。

## 1 背景信息



人脸人体识别技术是基于阿里云深度学习算法，结合图像或视频的人脸检测、分析、比对以及人体检测等技术，为您提供人脸人体的检测定位、人脸属性识别和人脸比对等能力。可以为开发者和企业提供高性能的在线API服务，应用于人脸AR、人脸识别和认证、大规模人脸检索、照片管理等各种场景。

## 2 前提条件



在开始之前，请确保完成以下步骤：

1. 开通人脸人体能力，请参见上述开发前准备。

The screenshot shows the Alibaba Cloud Visual Intelligence Open Platform console. On the left, there is a sidebar with a tree structure: '视觉智能开放平台' (Visual Intelligence Open Platform) is expanded, showing '开放能力' (Open Features) and '人脸人体' (Facial and Human Body). Other collapsed categories include '文字识别' (Text Recognition), '商品理解' (Product Understanding), '内容安全' (Content Safety), '图像识别' (Image Recognition), '图像生产' (Image Production), and '分割抠图' (Image Segmentation). The main content area is titled '人脸人体' (Facial and Human Body). It contains a brief description: '基于图像或视频中的人脸/人体检测、分析和比对技术，为用于提供人脸/人体的检测定位、人脸属性识别和人脸比对等应用能力。' followed by a '体验调试' (Experience and Debug) link. Below this is a section titled '产品与服务' (Products and Services) with a list of features: '包含人脸数畊检测、人脸特征点检测、人脸定位检测、人脸置信度检测、人脸姿势检测、人脸特征点定位检测、瞳孔检测、性别识别、年龄识别、简单表情识别、佩戴眼镜识别、人脸属性识别、人脸比对1:1等能力。' At the bottom of this section is a blue '立即开通' (Activate Now) button. The top navigation bar shows the current path: '视觉智能开放平台 / 人脸人体'.

2. 在您的 Java 工程中添加人脸人体能力的 pom 依赖：

```
<!-- https://mvnrepository.com/artifact/com.aliyun/aliyun-java-sdk-facebody -->
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-facebody</artifactId>
    <version>1.0.8</version>
</dependency>
```

### 3 人脸属性识别



[RecognizeFace](#) 可以识别检测人脸的性别、年龄、表情、眼镜四种属性。返回人脸的1024维深度学习特征，基于这个特征并按照特征比较规则，您可实现高性能的人脸识别。

例如需要识别以下图片中的人脸属性。



示例代码如下：

```
import com.aliyun.CommonConfig;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.facebody.model.v20191230.RecognizeFaceRequest;
import com.aliyuncs.facebody.model.v20191230.RecognizeFaceResponse;
import com.aliyuncs.profile.DefaultProfile;
import com.google.gson.Gson;

public class RecognizeFaceDemo {
    private static DefaultProfile profile = DefaultProfile.getProfile("cn-shanghai",
```

```
"<access key id>", "<access key secret>");  
private static IAcsClient client = new DefaultAcsClient(profile);  
  
public static void main(String[] args) {  
    String recognizeFaceURL = "https://visionapi-test.oss-cn-shanghai.aliyuncs.  
com/recognize_1.jpg";  
    recognizeFace(recognizeFaceURL);  
}  
  
/**  
 * 人脸属性识别  
 * @param imageUrl 图片 URL 地址  
 */  
  
private static void recognizeFace(String imageUrl)  
{  
    RecognizeFaceRequest recognizeFaceRequest = new RecognizeFaceRequest();  
    recognizeFaceRequest.setImageURL(imageUrl);  
  
    try {  
        RecognizeFaceResponse recognizeFaceResponse = client.getAcsResponse  
(recognizeFaceRequest);  
        System.out.println("人脸属性识别: ");  
        System.out.println(new Gson().toJson(recognizeFaceResponse));  
    } catch (ServerException e) {  
        e.printStackTrace();  
    } catch (ClientException e) {  
        System.out.println("ErrCode:" + e.getErrCode());  
        System.out.println("ErrMsg:" + e.getErrMsg());  
        System.out.println("RequestId:" + e.getRequestId());  
    }  
}  
}
```

代码返回结果类似如下：

```
{  
    "requestId": "9E461AFC-B125-4B04-8644-4D39C495B61A",  
    "data": {
```

```
"faceCount": 1,  
"landmarkCount": 105,  
"denseFeatureLength": 1024,  
"faceRectangles": [  
    604,  
    124,  
    128,  
    160  
,  
    "faceProbabilityList": [  
        55.0  
,  
        "poseList": [  
            -8.399615,  
            -10.90538,  
            14.451606  
,  
            "landmarks": [  
                614.1295,  
                161.77643,  
                647.84894,  
                ...  
,  
                "pupils": [  
                    637.25793,  
                    177.63129,  
                    5.2395487,  
                    689.71075,  
                    192.1905,  
                    5.2395487  
,  
                    "genderList": [  
                        0  
,  
                        "ageList": [  
                            24  
,  
                            ...  
                        ]  
                    ]  
                ]  
            ]  
        ]  
    ]  
]
```

```
  "expressions": [  
    1  
,  
    "glasses": [  
      1  
,  
      "denseFeatures": [  
        "-0.0080740926787257195",  
        "0.011421392671763897",  
        "-0.027754634618759155",  
        ...  
      ]  
    }  
  }
```

从返回结果中得到的该图片人脸属性识别结果如下：

人脸个数：1  
性别：女性  
年龄：24  
是否微笑：微笑  
是否戴眼镜：有眼镜

## 4 人体计数



[DetectBodyCount](#) 可以检测输入图片中人体的个数。

待检测图片如下：



示例代码如下：

```
import com.aliyun.CommonConfig;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.facebody.model.v20191230.DetectBodyCountRequest;
import com.aliyuncs.facebody.model.v20191230.DetectBodyCountResponse;
import com.aliyuncs.profile.DefaultProfile;
import com.google.gson.Gson;

/**
 * 人体计数
 */

public class DetectBodyCountDemo {

    private static DefaultProfile profile = DefaultProfile.getProfile("cn-shanghai",
    "<access key id>", "<access key secret>");

    private static IAcsClient client = new DefaultAcsClient(profile);

    public static void main(String[] args) {
        String detectBodyCountURL = "https://visionapi-test.oss-cn-shanghai.aliyuncs.com/
multiplayer.jpg";
        detectBodyCount(detectBodyCountURL);
    }

    /**
     * 人体计数
     * @param imageURL 图片URL地址
     */
    private static void detectBodyCount(String imageURL)
    {
        DetectBodyCountRequest detectBodyCountRequest = new DetectBodyCountRequest();
        detectBodyCountRequest.setImageURL(imageURL);
        try {
            DetectBodyCountResponse detectBodyCountResponse = client.getAcsResponse
```

```
(detectBodyCountRequest);

    System.out.println("人体计数: ");

    System.out.println(new Gson().toJson(detectBodyCountResponse));

} catch (ServerException e) {

    e.printStackTrace();

} catch (ClientException e) {

    System.out.println("ErrCode:" + e.getErrCode());

    System.out.println("ErrMsg:" + e.getErrMsg());

    System.out.println("RequestId:" + e.getRequestId());

}

}

}
```

从返回结果中得到该图片被检测到有3个人。

## 5 人脸搜索



人脸搜索可以根据输入图片，在数据库中搜索相似的人脸图片数据。

人脸搜索操作流程示意图如下：



说明: 公测阶段每个阿里云主账号只能创建一个数据库, 且开通服务后默认创建一个名称为 default 的数据库, 每个数据库最多可以添加5万张图片。

涉及的 API 如下：

## AddFaceEntity

## AddFace

## SearchFace

例如我们有以下人脸数据：



将上面图片样本加入人脸库，使用下面人脸数据搜索人脸库。



示例代码如下：

```
import com.aliyun.CommonConfig;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.facebody.model.v20191230.*;
import com.aliyuncs.profile.DefaultProfile;
import com.google.gson.Gson;

public class SearchFaceDemo {
    private static DefaultProfile profile = DefaultProfile.getProfile("cn-shanghai",
        CommonConfig.ACCESSKEY_ID, CommonConfig.ACCESSKEY_SECRET);
    private static IAcsClient client = new DefaultAcsClient(profile);
```

```
public static void main(String[] args) throws InterruptedException {
    String dbName = "default";
    String human1_1 = "https://visionapi-test.oss-cn-shanghai.aliyuncs.com/human_11.
jpg";
    String human1_2 = "https://visionapi-test.oss-cn-shanghai.aliyuncs.com/human_12.
jpg";
    String human2_1 = "https://visionapi-test.oss-cn-shanghai.aliyuncs.com/human_21.
jpg";
    String human2_2 = "https://visionapi-test.oss-cn-shanghai.aliyuncs.com/
human_22.jpg";
    String sample = "https://visionapi-test.oss-cn-shanghai.aliyuncs.com/
sample.jpg";
    String entityId1 = "human1";
    String entityId2 = "human2";

    // 1.创建人脸样本
    addFaceEntity(dbName, entityId1);
    addFaceEntity(dbName, entityId2);

    // 2.向人脸样本中加入人脸，每个样本人脸上限为5
    addFace(dbName, entityId1, human1_1);
    addFace(dbName, entityId1, human1_2);
    addFace(dbName, entityId2, human2_1);
    addFace(dbName, entityId2, human2_2);

    Thread.currentThread().sleep(3000);

    // 3.到人脸库中查找
    searchFace(dbName, sample, 1);
}

private static void addFaceEntity(String dbName, String entityId)
{
    AddFaceEntityRequest addFaceEntityRequest = new AddFaceEntityRequest();
    addFaceEntityRequest.setDbName(dbName);
    addFaceEntityRequest.setEntityId(entityId);
    try{
        AddFaceEntityResponse addFaceEntityResponse = client.getAcsResponse
(addFaceEntityRequest);
    }
}
```

```
        System.out.println("添加人脸样本: ");
        System.out.println(new Gson().toJson(addFaceEntityResponse));
    } catch (ServerException e) {
        e.printStackTrace();
    } catch (ClientException e) {
        System.out.println("ErrCode:" + e.getErrCode());
        System.out.println("ErrMsg:" + e.getErrMsg());
        System.out.println("RequestId:" + e.getRequestId());
    }
}

/**
 * 添加人脸数据
 * @param dbName 数据库名称
 * @param entityId 实体ID
 * @param imageUrl 人脸图片地址, 必须是同Region的OSS的图片地址。人脸必须是正面无
遮挡单人人脸。
*/
private static void addFace(String dbName, String entityId, String imageUrl)
{
    AddFaceRequest addFaceRequest = new AddFaceRequest();
    addFaceRequest.setDbName(dbName);
    addFaceRequest.setEntityId(entityId);
    addFaceRequest.setImageUrl(imageUrl);
    try{
        AddFaceResponse addFaceResponse = client.getAcsResponse(addFaceRequest);
        System.out.println("添加人脸数据: ");
        System.out.println(new Gson().toJson(addFaceResponse));
    } catch (ServerException e) {
        e.printStackTrace();
    } catch (ClientException e) {
        System.out.println("ErrCode:" + e.getErrCode());
        System.out.println("ErrMsg:" + e.getErrMsg());
        System.out.println("RequestId:" + e.getRequestId());
    }
}
```

```

    /**
     * 搜索人脸
     * @param dbName 数据库名称
     * @param imageUrl 图片URL地址。必须是同Region的OSS地址。
     */
    private static void searchFace(String dbName, String imageUrl, Integer limit)
    {
        SearchFaceRequest searchFaceRequest = new SearchFaceRequest();
        searchFaceRequest.setDbName(dbName);
        searchFaceRequest.setImageUrl(imageUrl);
        searchFaceRequest.setLimit(limit);
        try{
            SearchFaceResponse searchFaceResponse = client.getAcsResponse(searchFaceRequest);
            System.out.println("搜索人脸: ");
            System.out.println(new Gson().toJson(searchFaceResponse));
        } catch (ServerException e) {
            e.printStackTrace();
        } catch (ClientException e) {
            System.out.println("ErrCode:" + e.getErrCode());
            System.out.println("ErrMsg:" + e.getErrMsg());
            System.out.println("RequestId:" + e.getRequestId());
        }
    }
}

```

返回结果类似如下：

添加人脸样本：

```
{"requestId":"718AE1B9-F1B3-4720-98FA-69E2A41F4C31"}
```

添加人脸样本：

```
{"requestId":"9CE55AD1-1E21-49F0-BB2C-B3C07A015EAE"}
```

添加人脸数据：

```
{"requestId":"9551C7C7-8027-4E82-82FB-15FEC0EB8161","data":{"faceId":"1589791311965000"}}
```

添加人脸数据：

```
{"requestId":"C193A2E0-F22A-4418-AAD3-F28749BDDC17","data":{"faceId":"1589791312203000"}}
```

添加人脸数据:

```
{ "requestId": "687D9D8B-5EDE-4774-B377-C1CB4E0FB855", "data": { "faceId": "1589791312433000"}}
```

添加人脸数据:

```
{"requestId": "1E0948E3-7D75-4709-A624-073B4CB54874", "data": {"faceId": "1589791312679000"}}
```

搜索人脸:

```
{
  "requestId": "F0543F2A-216F-495A-8B4D-20CA06B605A7",
  "data": {
    "matchList": [
      {
        "faceItems": [
          {
            "faceId": "1589791312203000",
            "score": 0.86908734,
            "extraData": "",
            "entityId": "human1"
          }
        ],
        "location": {
          "x": 592,
          "y": 132,
          "width": 152,
          "height": 204
        }
      }
    ]
  }
}
```

从返回结果中得出输入图片:

被匹配到的样本数据faceId为1589791312203000。

匹配到的样本名称为human1。

与样本数据相似度为0.86908734。

# 4 15分钟开发视觉AI应用 ◎ 口罩识别实战教程

## 概述

人脸人体识别技术是基于阿里云深度学习算法，结合图像或视频的人脸检测、分析、比对以及人体检测等技术，为您提供人脸人体的检测定位、人脸属性识别和人脸比对等能力。本教程介绍如何使用 Alibaba Cloud SDK for Java 对图片中的人物进行口罩检测。

### 1 前提条件



在开始之前，请确保完成以下步骤：

1. 开通人脸人体能力，请参见上述开发前准备。

The screenshot shows the Alibaba Cloud Visual Intelligence Open Platform. On the left, there is a sidebar with a tree structure: '视觉智能开放平台' (Visual Intelligence Open Platform) at the top, followed by '概览' (Overview), '开放能力' (Open Features), and '人脸人体' (Facial and Body) which is highlighted. Below these are '文字识别' (Text Recognition), '商品理解' (Product Understanding), '内容安全' (Content Safety), '图像识别' (Image Recognition), '图像生产' (Image Production), and '分割抠图' (Segmentation Mask). The main content area is titled '人脸人体' (Facial and Body). It features a small icon of two people and a bar chart. Below the title, there is a brief description: '基于图像或视频中的人脸/人体检测、分析和比对技术，为用户提供人脸/人体的检测定位、人脸属性识别和人脸比对等应用能力。' followed by a '体验测试' (Experience Test) button. Further down, there is a '产品与服务' (Products and Services) section with a list of features: '包含人脸数量检测、人脸特征点检测、人脸定位检测、人脸识别度检测、人脸姿势检测、人脸特征点定位检测、瞳孔检测、性别识别、年龄识别、简单表情识别、佩戴眼镜识别、人脸属性识别、人脸比对1:1等能力。' At the bottom of this section is a '立即开通' (Activate Now) button.

2. 在您的 Java 工程中添加人脸人体能力的 pom 依赖：

```
<!-- https://mvnrepository.com/artifact/com.aliyun/aliyun-java-sdk-facebody -->
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-facebody</artifactId>
    <version>1.0.8</version>
</dependency>
```

## 2 人脸搜索



[DetectMask](#) 可以对输入图片中面积最大的人脸进行口罩检测。

例如要识别下面的图片中的人物是否戴了口罩。



示例代码如下：

```
import com.aliyun.CommonConfig;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.facebody.model.v20191230.DetectMaskRequest;
import com.aliyuncs.facebody.model.v20191230.DetectMaskResponse;
import com.aliyuncs.profile.DefaultProfile;
import com.google.gson.Gson;

public class DetectMaskDemo {
    private static DefaultProfile profile = DefaultProfile.getProfile("cn-shanghai",
            "<access key id>", "<access key secret>");
    private static IAcsClient client = new DefaultAcsClient(profile);
    public static void main(String[] args) {
        String wearMaskSampleImgURL = "https://visionapi-test.oss-cn-shanghai.aliyuncs.com/mask_1.jpg";
        detectMask(wearMaskSampleImgURL);
    }
}
```

```
/**  
 * 口罩识别  
 * @param wearMaskSampleImgURL 图片URL地址  
 */  
  
private static void detectMask(String wearMaskSampleImgURL) {  
    DetectMaskRequest detectMaskRequest = new DetectMaskRequest();  
    detectMaskRequest.setImageURL(wearMaskSampleImgURL);  
    try {  
        DetectMaskResponse detectMaskResponse = client.getAcsResponse(detectMas-  
kRequest);  
        System.out.println("口罩识别: ");  
        System.out.println(new Gson().toJson(detectMaskResponse));  
    } catch (ServerException e) {  
        e.printStackTrace();  
    } catch (ClientException e) {  
        System.out.println("ErrCode:" + e.getErrCode());  
        System.out.println("ErrMsg:" + e.getErrMsg());  
        System.out.println("RequestId:" + e.getRequestId());  
    }  
}  
}
```

代码返回结果类似如下：

```
{  
    "requestId": "3DFE230C-CB35-4F92-981F-F70D078E0C8D",  
    "data": {  
        "mask": 2,  
        "faceProbability": 0.57101476  
    }  
}
```

从返回结果中得到的该图片识别结果如下：

图片中的人物戴了口罩。

检测结果的可信度为0.57101476。

其中，返回结果中参数 mask 取值如下：

- 0: 没有检测出人脸或人脸清晰度不够。
- 1: 没有戴口罩。
- 2: 有戴口罩。
- 3: 口罩没有带好。

# 5 15分钟开发视觉AI应用 ◎ 图片内容安全实战教程

## 概述

内容安全技术是基于阿里云视觉分析技术和深度识别技术。本教程为您介绍如何通过智能视觉平台的图片检测能力保证内容安全。

## 1 背景信息



内容安全技术是基于阿里云视觉分析技术和深度识别技术，并经过在阿里经济体内和云上客户的多领域、多场景的广泛应用和不断优化，可提供风险和治理领域的图像识别、定位、检索等全面服务能力，不仅可以降低色情、涉恐、涉政、广告、垃圾信息等违规风险，而且能大幅度降低人工审核成本。

## 2 前提条件



在开始之前，请确保完成以下步骤：

1. 开通内容安全能力，请参见上述开发前准备。



The screenshot shows the Alibaba Cloud Visual Intelligence Platform interface. The left sidebar has a 'Content Safety' section highlighted. The main content area is titled 'Content Safety' with a sub-section 'Product & Service' containing text about identifying illegal content like色情 (adult content), 恐怖 (terrorism), 广告 (advertising), 垃圾信息 (spam), and text. A 'Start Now' button is visible.

2. 在您的 Java 工程中添加内容安全能力的 pom 依赖：

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/com.aliyun/aliyun-java-sdk-facebody -->
  <dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-imageaudit</artifactId>
    <version>1.0.6</version>
  </dependency>
</dependencies>
```

### 3 图片内容安全



[图片内容安全](#)支持检测的场景包括有图片智能鉴黄、图片涉恐涉政识别、图文违规识别、图片二维码识别、图片不良场景识别和图片logo识别等。

例如：识别以下图片是否涉嫌违规。



示例代码如下。

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.profile.DefaultProfile;
import com.google.gson.Gson;
import java.util.*;
import com.aliyuncs.imageaudit.model.v20191230.*;
```

```
public class ScanImage {  
    private static DefaultProfile profile = DefaultProfile.getProfile("cn-shanghai",  
    "<access key id>", "<access key secret>");  
    private static IAcsClient client = new DefaultAcsClient(profile);  
  
    public static void main(String[] args) {  
        ScanImageRequest request = new ScanImageRequest();  
        List<ScanImageRequest.Task> taskList = new ArrayList<ScanImageRequest.Task>();  
        ScanImageRequest.Task task1 = new ScanImageRequest.Task();  
        // 数据ID。需要保证在一次请求中所有的ID不重复。  
        task1.setDataId(UUID.randomUUID().toString());  
        // 待检测图像的URL。支持HTTP和HTTPS协议。当前仅支持上海地域的OSS链接。  
        task1.setImageURL("https://visionapi-test.oss-cn-shanghai.aliyuncs.com/  
        TB1k8mYCpY7gK0jSZKzXXaikpXa-692-440%5B1%5D.jpg");  
        taskList.add(task1);  
        request.setTasks(taskList);  
  
        // 指定图片检测的应用场景  
        List<String> sceneList = new ArrayList<String>();  
        // 图片智能鉴黄  
        sceneList.add("porn");  
        // 图片涉恐涉政识别  
        sceneList.add("terrorism");  
        // 图文违规识别  
        sceneList.add("ad");  
        // 图片不良场景识别  
        sceneList.add("live");  
        // 图片logo识别  
        sceneList.add("logo");  
        request.setScenes(sceneList);  
  
        try {  
            ScanImageResponse response = client.getAcsResponse(request);  
            System.out.println(new Gson().toJson(response));  
        } catch (ServerException e) {  
            e.printStackTrace();  
        } catch (ClientException e) {  
            System.out.println("ErrCode:" + e.getErrCode());  
        }  
    }  
}
```

```
        System.out.println("ErrMsg:" + e.getErrMsg());
        System.out.println("RequestId:" + e.getRequestId());
    }
}
}
```

代码返回结果类似如下：

```
{
  "RequestId": "B2B68CC0-62E8-4DE4-9657-1FDDB8718FDC",
  "Data": {
    "Results": [
      {
        "DataId": "3213132132131",
        "ImageURL": "https://visionapi-test.oss-cn-shanghai.aliyuncs.com/TB1k8mY-CpY7gK0jSZKzXXaikpXa-692-440%5B1%5D.jpg",
        "SubResults": [
          {
            "Suggestion": "pass",
            "Rate": 100,
            "Label": "normal",
            "Scene": "porn"
          },
          {
            "Suggestion": "block",
            "Rate": 99.88,
            "Label": "weapon",
            "Scene": "terrorism"
          },
          {
            "Suggestion": "pass",
            "Rate": 99.9,
            "Label": "normal",
            "Scene": "ad"
          },
          {
            "Suggestion": "pass",

```

```
        "Rate": 100,  
        "Label": "normal",  
        "Scene": "live"  
    },  
    {  
        "Suggestion": "pass",  
        "Rate": 99.9,  
        "Label": "normal",  
        "Scene": "logo"  
    }  
]  
}  
]  
}
```

从返回结果中得到的该图片识别结果如下：

智能鉴黄：通过  
涉恐涉政识别：不通过  
图文违规识别：通过  
图片不良场景识别：通过  
图片logo识别：通过



阿里云开发者“藏经阁”  
海量免费电子书下载



加入钉群  
获取更多资讯