SenseInsight-Common SDK(Java)开发者文档V1.0.0

修订历史

文档版本	修订日期	修订说明
V1.0.0	2019-09-24	创建文档初稿

目录：

1. 概述

1.1 SDK说明

CommonSDK主要提供一套智能识别的基本API。包括人脸动态跟踪，人脸静态检测，人脸属性，人脸特征值提取，人脸特征值比对，人脸活体检测，人体的动态跟踪，人体属性等功能。本SDK为其Java接口文档。

1.2 接入前准备

- 用户请确认license(首次激活，需联网)、模型文件的位置放置正确。
- 请用户确认libstad_common.so等so动态库位置放置正确。

2. 数据结构定义

2.1 宏定义及枚举

2.1.1 错误码

```
public class StResult {
    public static final int ST_OK = 0;
    public static final int ST_E_INVALIDARG = -1;
    public static final int ST_E_HANDLE = -2;
    public static final int ST_E_OUTOFMEMORY = -3;
    public static final int ST_E_FAIL = -4;
    public static final int ST_E_DELNOTFOUND = -5;
    public static final int ST_E_INVALID_PIXEL_FORMAT = -6;
    public static final int ST_E_FILE_NOT_FOUND = -7;
    public static final int ST_E_INVALID_FILE_FORMAT = -8;
    public static final int ST_E_FILE_EXPIRE = -9;
    public static final int ST_E_INVALID_AUTH = -13;
    public static final int ST_E_INVALID_APPID = -14;
    public static final int ST_E_AUTH_EXPIRE = -15;
    public static final int ST_E_UUID_MISMATCH = -16;
```

```
    public static final int ST_E_ONLINE_AUTH_CONNECT_FAIL = -17;
    public static final int ST_E_ONLINE_AUTH_TIMEOUT = -18;
    public static final int ST_E_ONLINE_AUTH_INVALID = -19;
    public static final int ST_E_LICENSE_IS_NOT_ACTIVABLE = -20;
    public static final int ST_E_ACTIVE_FAIL = -21;
    public static final int ST_E_ACTIVE_CODE_INVALID = -22;
    public static final int ST_E_PRODUCT_VERSION_MISMATCH = -23;
    public static final int ST_E_PLATFORM_NOTSUPPORTED = -24;
    public static final int ST_E_NO_NETWORK = -25;
    public static final int ST_E_FILE_ACCESS_ERROR = -997;
    public static final int ST_E_SOCKET_ERROR = -998;
    public static final int ST_E_CANNOT_BE_ACCESSED = -999;
    public static final int ST_E_GET_UUID_FAILED = -1001;
    public static final int ST_E_ONLINEISSE_INVALID_LICENSE = -1002;
    public static final int ST_E_ONLINEISSE_INVALID_LICENSE_NOT_REDISTRIBUTABLE
= -1003;
    public static final int ST_E_ONLINEISSE_INVALID_REDISTRUBUTE_LIMIT_EXCEED =
-1004;
    public static final int ST_E_INVALID_CAPABILITY = -1005;
    public static final int ST_E_UNSUPPORTED_MODEL_VERSION = -1006;
    public static final int ST_E_OUT_OF_MAX_SEARCH_NUMBER = -1007;
    public static final int ST_E_WRAPPER_UNSUPPORTED_IMAGE_FORMAT = -1100;
    public static final int ST_E_ONLINEISSE_INVALID_UNKNOWN_ERROR = -2000;
}
```

2.1.2 图像格式

```
public class StImageFormat {
    public static final int ST_PIX_FMT_GRAY8 = 0;
    public static final int ST_PIX_FMT_YUV420P = 1;
    public static final int ST_PIX_FMT_NV12 = 2;
    public static final int ST_PIX_FMT_NV21 = 3;
    public static final int ST_PIX_FMT_BGRA8888 = 4;
    public static final int ST_PIX_FMT_BGR888 = 5;
}
```

2.1.3 人脸方向

```
public class StFaceOrientation {
    public static final int ST_FACE_UP = 1;
    public static final int ST_FACE_LEFT = 2;
    public static final int ST_FACE_DOWN = 4;
    public static final int ST_FACE_RIGHT = 8;
    public static final int ST_FACE_UNKNOWN = 15;
}
```

2.2 基础数据结构

2.2.1 Point

```
public class StPointF {
    public float x;
    public float y;
}
```

2.2.2 Rect

```
public class StRect {
    public int left;
    public int top;
    public int right;
    public int bottom;
}
```

2.2.3 图像

```
public class StImage {
    public byte[] getImageData();

    public void setImageData(byte[] imageData);

    public int getWidth();

    public void setWidth(int width);

    public int getHeight();

    public void setHeight(int height) ;
}
```

2.2.4 人脸

```
public class StFace {
    /**
     * @return 质量分
     */
    public float getQuality();

    /**
     * @return 结果标签
     */
    public int getLabel();

    /**
     * @return 人脸朝向
     */
    public int getFaceOrientation();

    /**
     * @return 人脸关键点
     */
    public StPointF[] getFacePoints();

    /**
     * @return 人脸框
     */
    public StRect getFaceRect();

    /**
     * @return 分值，目前和getQuality结果相同
     */
```

```
    public float getScore();

    /**
     * @return 水平转角，视频中人头的方向，左负右正
     */
    public float getYaw();

    /**
     * @return 俯仰角，上负下正
     */
    public float getPitch();

    /**
     * @return 旋转角，左负右正
     */
    public float getRoll();

    /**
     * @return 眼间距（未支持）
     */
    public float getEyeDist();

    /**
     * @return face id
     */
    public int getId();

    /**
     * 目前未支持
     */
    public float getGazeYaw();

    /**
     * 目前未支持
     */
    public float getGazePitch();
}
```

2.2.5 人脸属性

```
public class StAttributeResult {
    /**
     * 设置属性值
     * @param paramArrayOfInt 人脸属性值
     */
    public void setFeature(int[] paramArrayOfInt);

    /**
     * @return 预测的年龄值
     */
    public int getAge();

    /**
     * @return 0: 女性 1:男性
     */
    public int getGender();
```

```
    /**
     * @return  预测的魅力值(0 ~ 100)
     */
    public int getAttrActive() ;

    /**
     * @return 0: 愤怒 1: 平静 2: 厌恶 3: 愉快 4: 难过 5: 恐惧 6: 吃惊
     */
    public int getEmotion() ;

    /**
     * @return 0:无眼镜 1:普通眼镜 2:太阳镜
     */
    public int getGlass() ;

    /**
     * @return 0:未戴口罩 1:戴口罩
     */
    public int getMask();

    /**
     * @return 0:黄种人 1:黑种人 2:白种人
     */
    public int getRace();

    /**
     * @return 预测的微笑程度(0 ~ 100)
     */
    public int getSmileScore();
}
```

2.2.6 人脸特征值

```
public class StFaceFeature {
    public static StFaceFeature createStFaceFeature(byte[] feat);
    public byte[] getByteFeature() throws StFaceException;
    public void recycle();
    public boolean isRecycled();
}
```

2.2.7 人体

```
public class StBody {
    public float getQuality();

    public int getLabel();

    public int getBodyOrientation();

    public StRect getBodyRect();

    public StPointF[] getBodyPoints();

    public float getScore();

    public int getId();
```

```
}
```

2.2.8 人体属性

```
public class StBodyAttributeResult {
    public StBodyAttribute[] getBodyAttributes();
}

public class StBodyAttribute {
    public String getCategory();
    public String getLabel();
}
```

3. 接口说明

3.1 license校验

License校验分为两步：

1. 根据license文件内容获取授权激活码。（授权激活码成功成功后，最好缓存到文件中，后续校验时从缓存文件中读取。以免每次校验都需要联网生成授权激活码）
2. 激活授权激活码。

license校验的API在stLibrary.java文件中。

3.1.1 获取授权激活码

客户端需要获得授权激活码才能使用商汤SDK中的各项算法功能，所以客户端需要先读取license文件，并调用该接口获得激活码。

```
/**
 * 获取授权激活码。第一次获取成功后，可以将其缓存到文件，后续校验时从文件读取即可。
 * @param licName license授权的产品名
 * @param licContent license文件的内容
 * @param retCode 错误码。0表示成功
 * @return 授权激活码。
 */
public static native String onlineActivite(String licName,String licContent,
int[] retCode);
```

3.1.2 激活授权激活码

```
/**
 * 激活授权激活码
 * @param licName license授权的产品名
 * @param licContent license文件的内容
 * @param actCode 授权激活码，从onlineActivite API返回（可以缓存在文件中）
 * @return 校验结果 0 激活成功，其它值失败
 */
public static native int addLic(String licName,String licContent, String
actCode);
```

3.1.3 示例程序

```
```

```
int activate(String productName, String licensePath, String activateCodePath) {
    String license = readFileContent(licensePath);
    if (TextUtils.isEmpty(license)) {
        return -1;
    }

    String actCode = readFileContent(activateCodePath);
    if (TextUtils.isEmpty(actCode)) {
        int[] retCode = new int[1];
        actCode = StLibrary.onlineActivite(productName, license, retCode);
        if (!TextUtils.isEmpty(actCode)) {
            WriteToFile(actCode, activateCodePath);
        }
    }

    int ret = StLibrary.addLic(productName, license, TextUtils.isEmpty(actCode)
? "" : actCode);
    return ret;
}
```

3.2 人脸检测

3.2.1 创建人脸检测对象

```
/**
 * 创建人脸检测对象
 * @param detectModelPath detect模型路径
 * @param alignModelPath align模型路径
 * @param poseModelPath pose模型路径
 * @param cfg 始终传0
 * @throws StFaceException
 */
public StFaceTrack(String detectModelPath,String alignModelPath, String
poseModelPath, int cfg) throws StFaceException
```

3.2.2 动态人脸跟踪监测

```
/**
 * 对目标帧图像进行动态人脸跟踪检测，将返回图像中检测到的所有人脸信息。
 * 返回的人脸信息中包含人脸位置，人脸宽高，检测的质量分，可用来作为是否进行人脸属性识别和抽取特
征值的依据
 * （例如只对质量分大于等于0.6的人脸进行属性和特征值操作）。
 *
 * @param imageData 用于检测的目标帧图像字节数据
 * @param imageFormat 目标帧图像格式，详情请参考类StImageFormat的定义。
 * @param imageWidth 目标帧图像宽度
 * @param imageHeight 目标帧图像高度
 * @param orientation 目标帧图像方向，详情请参考类StFaceOrientation定义
 * @return 返回目标帧中所有的人脸信息，包括人脸位置，宽高，人脸关键点信息等，详情请参考class
StFace的定义。
 * 返回人脸信息可能为空，使用前需要判断一下是否为空。
 * @throws StFaceException -
 */
public StFace[] track(byte[] imageData, int imageFormat, int imageWidth, int
imageHeight, int orientation) throws StFaceException
```

注意：StFace中getEyeDist(), getLabel(), getFaceOrientation(), getGazePitch(), getGazeYaw()接口获取的值无效，没有开发。

- 示例:

```
StFace[] faces = mTrack.track(data, StImageFormat.ST_PIX_FMT_NV21, mWidth,
mHeight, StFaceOrientation.ST_FACE_UP);
```

3.2.3 静态人脸检测

```
/**
 * 对目标帧图像进行静态人脸检测，将返回图像中检测到的所有人脸信息。
 * @param imageData 用于检测的目标帧图像字节数据
 * @param imageFormat 目标帧图像格式，详情请参考类StImageFormat的定义。
 * @param imageWidth 目标帧图像宽度
 * @param imageHeight 目标帧图像高度。
 * @param stride 目标帧图像的行宽（图像每行占据的字节数）输入0即可。
 * @param orientation 目标帧图像方向，详情请参考类StFaceOrientation定义。
 * @return 返回目标帧中所有的人脸信息，包括人脸位置，宽高，人脸关键点信息等，详情请参考class
StFace的定义。
 * 返回人脸信息可能为空，使用前需要判断一下是否为空。
 * @throws StFaceException -
 */
public StFace[] detect(byte[] imageData, int imageFormat, int imageWidth, int
imageHeight, int stride, int orientation) throws StFaceException
```

- 示例:

```
StFace[] faces = mTrack.detect(data, StImageFormat.ST_PIX_FMT_NV21, mWidth,
mHeight, mWidth , StFaceOrientation.ST_FACE_UP);
```

3.2.4 静态检测人脸基本信息

```
/**
 * 对目标帧图像进行静态人脸检测，将返回图像中检测到的所有人脸基本信息，包括矩形框，人脸方向等。
 * @param imageData 用于检测的目标帧图像字节数据
 * @param imageFormat 目标帧图像格式，详情请参考类StImageFormat的定义。
 * @param imageWidth 目标帧图像宽度
 * @param imageHeight 目标帧图像高度
 * @param orientation 目标帧图像方向，详情请参考类StFaceOrientation定义
 * @return 返回目标帧中所有的人脸基本信息。只有人脸区域，方向和score会被填充。 详情请参考
class StFace的定义
 * @throws StFaceException -
 */
public StFace[] detectBasicInfo(byte[] imageData, int imageFormat, int
imageWidth, int imageHeight, int orientation) throws StFaceException
```

- 示例:

```
StFace[] faces = mTrack.detectBasicInfo(data, StImageFormat.ST_PIX_FMT_NV21,
mWidth, mHeight, mWidth , StFaceOrientation.ST_FACE_UP);
```

3.2.5 获取人脸详情

```
/**
 * 根据输入的人脸基本信息返回详细信息。返回人脸信息可能为空，使用前需要判断一下是否为空。
 * @param imageData 用于检测的目标帧图像字节数据
 * @param imageFormat 目标帧图像格式，详情请参考类StImageFormat的定义。
 * @param imageWidth 目标帧图像宽度
 * @param imageHeight 目标帧图像高度
 * @param basicInfo 人脸的基本信息，从detectBasicInfo方法得到。
 * @return 返回目标帧中人脸的详细信息，包括106个关键点。
 * @throws StFaceException
 */
public StFace getDetailedInfo(byte[] imageData, int imageFormat, int imageWidth,
int imageHeight, StFace basicInfo) throws StFaceException
```

- 示例：

```
StFace[] faces = mTrack.getDetailedInfo(data, StImageFormat.ST_PIX_FMT_NV21,
mWidth, mHeight, mWidth, basicInfo);
```

3.2.6 获取人脸质量

```
/**
 * 人脸检测获得人脸信息结果后，还可以进一步调用该接口获得人脸检测的质量分，可以作为是否进行    *
人脸属性识别和抽取特征值的依据。
 * @param data 用于检测的目标帧图像字节数据
 * @param imageFormat 目标帧图像格式，详情请参考类StImageFormat的定义。
 * @param face 用于获取质量分的目标人脸信息，通过人脸检测功能获得
 * @param width 目标帧图像宽度
 * @param height 目标帧图像高度
 * @return 目标帧图像中目标人脸的检测质量分，可以作为是否进行人脸属性识别和抽取特征值的依据。
 * @throws StFaceException -
 */
public float getDetectImageQuality(byte[] data, int imageFormat, StFace face,
int width, int height) throws StFaceException
```

- 示例：

```
StFace[] faces = mTrack.track(data, StImageFormat.ST_PIX_FMT_NV21, mWidth,
mHeight, StFaceOrientation.ST_FACE_UP);
float quality = mTrack.getDetectImageQuality(data, faces[0], mWidth, mWidth);
```

3.2.7 释放人脸检测对象

```
public void release()
```

3.3 人脸活体检测

3.3.1 获得人脸活体检测对象

```
/**
 * 创建活体检测对象
 * @param liveModelPath liveness 模型路径
 * @param alignModelPath align 模型路径
 * @throws StFaceException -
 */
public StFaceLiveness(String liveModelPath, String alignModelPath) throws
StFaceException
```

3.3.2 检测人脸活体检测特征

```
/**
 * 对目标帧图像中指定人脸检测人脸活体特征，将返回人脸的活体特征置信度分值(人脸为假脸的概率值)。
 * @param imageData 用于检测的目标帧图像字节数据
 * @param imageFormat 目标帧图像格式，详情请参考类StImageFormat的定义。
 * @param imageWidth 目标帧图像宽度
 * @param imageHeight 目标帧图像高度。
 * @param stFace 用于获取人脸属性的目标人脸信息，通过人脸检测功能获得
 * @return 人脸的活体特性置信度(人脸为假脸的概率值，取值范围0 ～1).
 * @throws StFaceException -
 */
public float liveness(byte[] imageData, int imageFormat, int imageWidth, int
imageHeight, StFace stFace)
        throws StFaceException
```

- 示例：

```
StFace[] faces = mTrack.detect(data, StImageFormat.ST_PIX_FMT_NV21, mWidth,
mHeight, mWidth , StFaceOrientation.ST_FACE_UP);
float score = faceLiveness.liveness(data, StImageFormat.ST_PIX_FMT_NV21, mWidth,
mHeight, faces[0]);
```

3.3.3 释放人脸活体检测对象

```
public void release()
```

3.4 人脸属性

3.4.1 获取人脸属性算法对象

```
/**
 * 创建人脸属性算法对象
 * @param attrModelPath 属性模型路径
 * @param alignModelPath align模型路径
 * @throws StFaceException-
 */
public StFaceAttribute(String attrModelPath, String alignModelPath)
```

3.4.2 人脸属性识别

```
/**
 * 获取人脸属性，返回图像中指定人脸的属性信息（包括性别，年龄，微笑，魅力等）
 * @param imageData 用于检测的目标帧图像字节数据
 * @param imageFormat 目标帧图像格式，详情请参考类StImageFormat的定义。
 * @param imageWidth 目标帧图像宽度
 * @param imageHeight 目标帧图像高度
 * @param stFace 用于获取人脸属性的目标人脸信息，通过人脸检测功能获得
 * @return 人脸属性信息（包括年龄，性别等等）,详情请参考class StAttributeResult的定义
 * @throws StFaceException -
 */
public StAttributeResult attribute(byte[] imageData, int imageFormat, int
imageWidth, int imageHeight, StFace stFace)
        throws StFaceException
```

- 示例：

```
StFace[] faces = mTrack.detect(data, StImageFormat.ST_PIX_FMT_NV21, mWidth,
mHeight, mWidth , StFaceOrientation.ST_FACE_UP);
StAttributeResult attrRes = faceAttribute. Attribute(data,
StImageFormat.ST_PIX_FMT_NV21, mWidth, mHeight, faces[0]);
int age = attrRes.getAge();
int gender = attrRes.getGender(); //0 : 女性 ；  1 ：  男性
```

3.4.3 释放人脸属性算法对象

```
public void release()
```

3.5 人脸特征值提取和比对

3.5.1 获得人脸特征值算法对象

```
/**
 * 创建特征值算法对象
 * @param alignModelPath align模型路径
 * @param verifyModelPath verify模型路径
 * @throws StFaceException -
 */
public StFaceVerify(String alignModelPath, String verifyModelPath)
```

3.5.2 抽取人脸特征值

```
/**
 * 抽取目标帧图像中指定人脸的特征值数据
 * @param imageData 用于检测的目标帧图像字节数据
 * @param imageFormat 目标帧图像格式，详情请参考类StImageFormat的定义。
 * @param imageWidth 目标帧图像宽度
 * @param imageHeight 目标帧图像高度
 * @param stFace 用于获取人脸特征值的目标人脸信息，通过人脸检测功能获得
 * @return 人脸特征值数据，详情请参考类StFaceFeature定义。
 * @throws StFaceException -
 */
public StFaceFeature getFeature(byte[] imageData, int imageFormat, int
imageWidth, int imageHeight, StFace stFace)
        throws StFaceException
```

3.5.3 人脸特征值比较

```
/**
 * 比较2个人脸的特征值数据，返回2张人脸是同一个人的置信度分值。
 * @param stFaceFeature1 特征值1，由接口getFeature返回
 * @param stFaceFeature2 特征值2，由接口getFeature返回
 * @return 参数对应的2张人脸特征值数据是同一个人的置信度分值。
 * @throws StFaceException -
 */
public float compareFeature(StFaceFeature stFaceFeature1, StFaceFeature
stFaceFeature2)
```

- 示例：

```
StFaceFeature featRes1 = faceFeat.getFeature (data1,
StImageFormat.ST_PIX_FMT_NV21, mWidth, mHeight, mWidth,  faces[0]);
StFaceFeature featRes2 = faceFeat.getFeature (data2,
StImageFormat.ST_PIX_FMT_NV21, mWidth, mHeight, mWidth,  faces[0]);
float score = faceFeat.compareFeature(featRes1, featRes2);
```

3.5.4 释放人脸特征值算法对象

```
public void release()
```

3.6 图像工具类

3.6.1 获得图像工具类对象

```
/**
 * 创建图片工具类对象
 * @param alignModelPath align模型路径
 * @throws StFaceException -
 */
public StFaceImage(String alignModelPath) throws StFaceException
```

3.6.2 裁剪图片

```
/**
 * 裁剪图片。客户端可调用该接口对目标帧图像进行裁剪，获得一张新的图像，新图像有固定长宽(178 *
218)并自动将目标人脸居中。
 * @param imageData 用于图片裁剪的目标帧图像字节数据
 * @param face 用于图片裁剪的目标人脸信息，通过人脸检测功能获得。
 * @param imageFormat 目标帧图像格式，详情请参考类StImageFormat的定义。
 * @param width 目标帧图像宽度
 * @param height 目标帧图像高度
 * @return 裁剪后的图片，该图片固定长宽(178 * 218)并将目标人脸居中
 * @throws StFaceException -
 */
public StImage captureAvat(byte[] imageData, StFace face, int imageFormat, int
width, int height) throws StFaceException
```

- 示例：

```
StFace[] faces = mTrack.detect(data, StImageFormat.ST_PIX_FMT_NV21, mWidth,
mHeight, mWidth , StFaceOrientation.ST_FACE_UP);
faceImg.captureAvat(data, faces[0], StImageFormat.ST_PIX_FMT_NV21, mWidth,
mHeight)
```

3.6.3 释放图片工具类对象

```
public void release()
```

3.7 人体检测

3.7.1 创建人体检测对象

```
/**
 * 创建人体检测对象
 * @param detectModelPath 人体检测对象模型路径
 * @param alignModelPath align模型路径
 * @throws StFaceException -
 */
public StBodyTrack(String detectModelPath, String alignModelPath)
        throws StFaceException
```

3.7.2 动态人体跟踪检测

```
/**
 * 动态人体跟踪检测
 * 对目标帧图像进行动态人体跟踪检测，将返回图像中检测到的所有人体信息，返回的人脸信息中包含
 * 人体区域，检测的质量分 等。返回人体信息可能为空，使用前需要判断一下是否为空。
 * @param imageData 用于检测的目标帧图像字节数据
 * @param imageFormat 目标帧图像格式，详情请参考类StImageFormat的定义。
 * @param imageWidth 目标帧图像宽度
 * @param imageHeight 目标帧图像高度
 * @param orientation 目标帧图像方向，详情请参考类StFaceOrientation定义
 * @return 返回目标帧中所有的人体信息，包括人体位区域，人体关键点信息等，详情请参考class
StBody的定义。
 * @throws StFaceException -
 */
public StBody[] track(byte[] imageData, int imageFormat, int imageWidth, int
imageHeight, int orientation)
        throws StFaceException
```

3.7.3 释放人体检测对象

```
public void release();
```

3.8 人体属性

3.8.1 获得人体属性算法对象

```
/**
 * 创建人体属性算法对象
 * @param attrModelPath 属性对象模型
 * @throws StFaceException -
 */
public StBodyAttributes(String attrModelPath) throws StFaceException
```

3.8.2 人体属性识别

```
/**
 * 对目标帧图像中指定人体进行人体属性识别，将返回图像中指定人体的属性信息
 * @param imageData 用于检测的目标帧图像字节数据
 * @param imageFormat 目标帧图像格式，详情请参考类StImageFormat的定义
 * @param imageWidth 目标帧图像宽度
 * @param imageHeight 目标帧图像高度
 * @param stBody 用于获取人体属性的目标人体信息，通过人体检测功能获得
 * @return 人体属性信息,详情请参考class StBodyAttributeResult的定义。含有label和
category两个字段。
 * @throws StFaceException -
 */
public StBodyAttributeResult attribute(byte[] imageData, int imageFormat, int
imageWidth, int imageHeight, StBody stBody)
        throws StFaceException
```

- 示例

```
StBodyAttributeResult attribute = bodyAttributes.attribute(data
        , StImageFormat.ST_PIX_FMT_NV21, mWidth, mHeight, body);
StBodyAttribute[] attrs = attribute.getBodyAttributes();
for (StBodyAttribute item : attrs) {
    builder.append('\n').append("Label: ").append(item.getLabel())
            .append('\n').append("Category: ").append(item.getCategory());
}
builder.append("\n\n");
```

3.8.3 释放人体属性算法对象

```
public void release();
```

4. 示例程序

- 请参看SDK_test.java. 注意各种算法对象不用后需要释放(调用release方法).