key	value
版本	v3.1.0.0
时间	2023-06-27

1. 接口说明

请参考 Demo app 中的 `OfflineOCR` 类的用法，该类中的主要方法列表如下：

```
public class OfflineOCR {
    private Context appContext;

    public int init_sdk_and_env(Context mContext, AssetManager assetM, String
appKey);

    public native int init(Context context, AssetManager assetM, String appID, int
numThread);

    public native String inferGeneralStatic(Bitmap bitmap, boolean do_para_det);

    public native String inferGeneralYUV(byte[] data, int width, int height, int
rowStride, Bitmap img_to_show_bmp);

    public native boolean feedHighResolutionFrameYUV(byte[] data, int width, int
height, int rowStride);
```

```
    public native boolean release();

    private OfflineOCR() {
    }

    public static OfflineOCR getInstance() {
        return OfflineOCR.Inner.instance;
    }

}
```

1.1 初始化接口

```
public int init_sdk_and_env(Context mContext, AssetManager assetM, String appKey);
```

该方法内部会依次做如下事情：1.将运行时依赖的部分动态库拷贝至DCIM目录下；2.设置环境变量`ADSP_LIBRARY_PATH`指向如下目录：`${DCIM目录}/adsp_runtime_libs;/system/lib/rfsa/adsp;/system/vendor/lib/rfsa/adsp;/dsp`；3.调用native方法`init`。

params:

- `mContext`: 当前上下文，可通过`this.getApplicationContext()`获取。
- `assetM`: AssetManager对象，用于将assets目录入口传入native层，以便sdk加载模型。
- `appKey`：有道智云平台上的"应用ID"，用于鉴权。

return:

- `0`: 表示初始化成功；
- `-1000`: 表示初始化失败，一般是模型初始化失败。
- 小于`-8000`：表示初始化失败，原因是鉴权未通过，请联系有道客服或工程师。

1.2 单张图片OCR方法

```
public native String inferGeneralStatic(Bitmap bitmap, boolean do_para_det);
```

params:

- `bitmap`: 待识别的图片，Bitmap形式。
- `do_para_det`: 无效参数，置为false即可。

return:

- 返回JSON字符串。包括如下字段：
    - `Status`：可能取的值包括：`SUCCESS`、`ERROR`。
    - `Result`：
        - 当`Status`为`ERROR`时，`Result`内容为错误信息；

    - 当`Status`为`SUCCESS`时，`Result`内容为OCR结果，该结果也是JSON字符串，其详细结构见第3节说明。

1.3 实况OCR方法（低清路）

```
public native String inferGeneralYUV(byte[] data, int width, int height, int
rowStride, Bitmap img_to_show_bmp);
```

【注意】：实况OCR需要同时依赖高清、低清两路camera数据，详细介绍见第4.6节说明。

params:

- `data`: 低清路的一帧YUV数据，格式为NV21。
- `width`: 该帧的宽度。
- `height`: 该帧的高度。
- `rowStride`：一行的像素数，由于camera返回的数据可能存在row padding，所以该值可能大于`width`的值。该值的获取方式，可参考`java/com/youdao/YDOfflineOCRDemo/CameraXActivity.java`。
- `img_to_show_bmp`：用于绘制高亮帧的Bitmap对象，算法会将目标段落高亮显示，并将该高亮图绘制在这个bitmap上。请在应用层提前初始化足够大的Bitmap对象。

return:

- 返回JSON字符串。解析为JSON对象后，包括如下字段：
    - `want_HR`：bool值。默认为`false`，当需要应用层提供高清路输入时，会被置为`true`，详情见第4.6节。
    - `general_OCR_res`：JSON字符串。解析为JSON对象后，包括如下字段：
        - `Status`：可能取的值包括：`SUCCESS`、`ERROR`。
        - `Result`：
            - 当`Status`为`ERROR`时，`Result`内容为错误信息；
            - 当`Status`为`SUCCESS`时，`Result`内容为OCR结果，该结果也是JSON字符串，其详细结构见第3节说明。

1.4 实况OCR方法（高清路）

```
 public native boolean feedHighResolutionFrameYUV(byte[] data, int width, int
height, int rowStride);
```

【注意】：实况OCR需要同时依赖高清、低清两路camera数据，详细介绍见第4.6节说明。

params:

- `data`: 高清路的一帧YUV数据，格式为NV21。
- `width`: 该帧的宽度。
- `height`: 该帧的高度。

- `rowStride`：一行的像素数，由于camera返回的数据可能存在row padding，所以该值可能大于`width`的值。该值的获取方式，可参考`java/com/youdao/YDOfflineOCRDemo/CameraXActivity.java`。

return:

- bool值。为`true`表明调用成功，为`false`表明调用失败。

1.5 释放方法

```
public native boolean release();
```

不再需要算法时，调用该方法进行释放。但建议慎重调用，因为初始化比较耗时，所以尽量不要频繁初始化/释放。

return:

- bool值。为`true`表明调用成功，为`false`表明调用失败。

1.6 实例化方法

```
public static OfflineOCR getInstance();
```

OfflineOCR 被实现为单例，可通过调用 `getInstance` 方法获取单例实例。

2. OCR结果JSON结构

下表所示结构为：

1. `inferGeneralYUV` 方法返回结果中的 `general_OCR_res`-`Result`字段中的JSON字符串解析出来的JSON对象结构；
2. `inferGeneralStatic`方法返回结果中的`Result`字段中的JSON字符串解析出来的JSON对象结构。

字段	类型	说明
Status	text	一定存在，识别成功返回success否则返回error
Reason	text	当识别错误时存在，返回错误原因。目前只有一种可能：“offline OCR no text detected.”表示未检测到文字。
Result	text	当识别成功时存在。注意它的结果是个字符串，因此还要再做解一次json串
+regions	text	包含各个文本区域regions。结果可能为空
++boundingBox	text	region的位置，四个顶点(顺序为左上、右上、右下、左下)的x、y坐标，由逗号分隔
++lines	text	region包含各个文本行lines

字段	类型	说明
+++boundingBox	text	line的位置，四个顶点(顺序为左上、右上、右下、左下)的x、y坐标，由逗号分隔
+++text	text	line的文本内容字符串
+++words	text	line包含各个words
++++boundingBox	text	word的位置，四个顶点(顺序为左上、右上、右下、左下)的x、y坐标，由逗号分隔
++++word	text	word的内容

3. 接入流程

3.1 开通权限

【注意】当前SDK版本不是正式版本，故临时使用“通用OCR”功能的按端激活方式授权，需要联系我们在后台配合操作授权。

1. 登录有道智云平台，申请通用OCR权限（https://ai.youdao.com/console/#/service-singleton/universal-character-recognition）。
2. 点击“创建应用”，“选择服务”中勾选“光学字符识别服务”-“通用文字识别”，“接入方式”选择“Android SDK”。其他条目按网页提示填写即可。
3. 联系我们，提供“应用ID”，我们在后台开通按端激活权限。

3.2 引用sdk

1. 把 offlineocr-release.aar 和 zhiyun_offline_common.jar 都放在 app/libs 下；
2. 修改 app 的 build.gradle，在其中的 dependencies 部分加入如下一行内容：

```
dependencies {
    ...
    implementation fileTree(dir: 'libs', include: ['*.jar','*.aar'])
    ...
}
```

3. 修改 app 的 build.gradle，在其中的 android 部分加入如下 repositories 段落：

```
android {
    ...

    repositories {
        flatDir {
            dir 'libs'
        }
    }
    ...
}
```

3.3 声明依赖的系统库

在安卓12及以上的系统版本中，需要显式声明对部分系统库的依赖。
在 `AndroidManifest.xml` 中， `manifest` 标签下的 `application` 标签下，加入如下内容：

```
<uses-native-library
    android:name="libcdsprpc.so"
    android:required="true" />

<uses-native-library
    android:name="libOpenCL.so"
    android:required="true" />
```

3.4 初始化SDK

参考 Demo 源码中的 `java/com/youdao/YDOfflineOCRDemo/MyApplication.java`。核心代码如下：

```
SecurityUtil.setContext(this);
tipOCR = OfflineOCR.getInstance();
int init_ret = tipOCR.init_sdk_and_env(this.getApplicationContext(), getAssets(),
MY_APP_KEY);
```

其中`MY_APP_KEY`是有道智云平台上的"应用ID"。

3.5 调用相册单张图片OCR功能

参考 Demo 源码中的 `java/com/youdao/YDOfflineOCRDemo/TouchSelectTextActivity.kt`。核心是调用 `OfflineOCR` 的 `inferGeneralStatic` 方法。详见第2.2节的方法说明。

3.6 调用实况OCR功能

参考 Demo 源码中的 `java/com/youdao/YDOfflineOCRDemo/CameraXActivity.java`。

实况OCR依赖两路camera输入，一路低清、一路高清。 两路的宽高比，需保持一致。不同机型的camera支持分辨率不同，如需自动选取最优分辨率，请参考`CameraXActivity.java`中`startCamera`方法的源码。

低清路以camera默认的帧率取帧，并将每帧都通过`OfflineOCR`类的`inferGeneralYUV`方法输入算法。算法将高亮显示每帧中的合法目标段落，并将该“高亮帧”写入该方法的最后一个参数传入的Bitmap。应用层可逐帧绘制该Bitmap，以将“高亮帧”显示给用户。

算法通过对**低清路**的逐帧检测，将在内部判定当前画面中的目标段落是否稳定，如果已经稳定，则会在最新一帧低清帧对应的`inferGeneralYUV`方法返回值中通知应用层。（具体来说，`inferGeneralYUV`方法返回的是 JSON 字符串，对该字符串的解析，请参考 `CameraXActivity.java` 中的 `processResult`方法。）该 JSON 中的 `want_HR` 字段默认是 `false`，而在算法判定画面稳定时，`want_HR`字段将被赋值为 `true`。此时需要应用层主动从**高清路**取一帧，并异步调用`OfflineOCR`类的`feedHighResolutionFrameYUV`方法，将高清帧输入算法，进行OCR。OCR的结果会写入算法内部的缓冲区，并被尽快写入低清路对应的`inferGeneralYUV`方法的返回值中。（举个例子：高清帧输入算法后，立即开始了异步的OCR，但OCR是比较耗时的，例如可能低清帧又连续输入了10帧后，上次高清路的OCR才做完，则第11帧的低清路数据对应的`inferGeneralYUV`方法的返回值JSON中，就会包含本次高清路的OCR结果）。

【注意】应用层在收到`want_HR`信号后，应只送一帧**高清路**数据。即，每次收到`want_HR`为`true`的信号时，送一帧**高清路**数据，其他时候不要调用`OfflineOCR`类的`feedHighResolutionFrameYUV`方法。