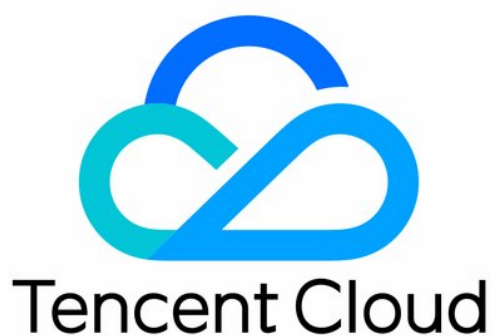


腾讯特效 SDK

SDK 集成指引（无 UI）

产品文档



【版权声明】

©2013–2025 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其他腾讯云服务相关的商标均为腾讯集团下的相关公司主体所有。另外，本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

文档目录

SDK 集成指引（无 UI）

通用集成腾讯特效

iOS

Android

原子能力集成指引

手势识别

人脸点位

人像分割

人脸属性

Android

人脸表情

iOS

Android

身体点位

iOS

Android

语音转表情

Android

iOS

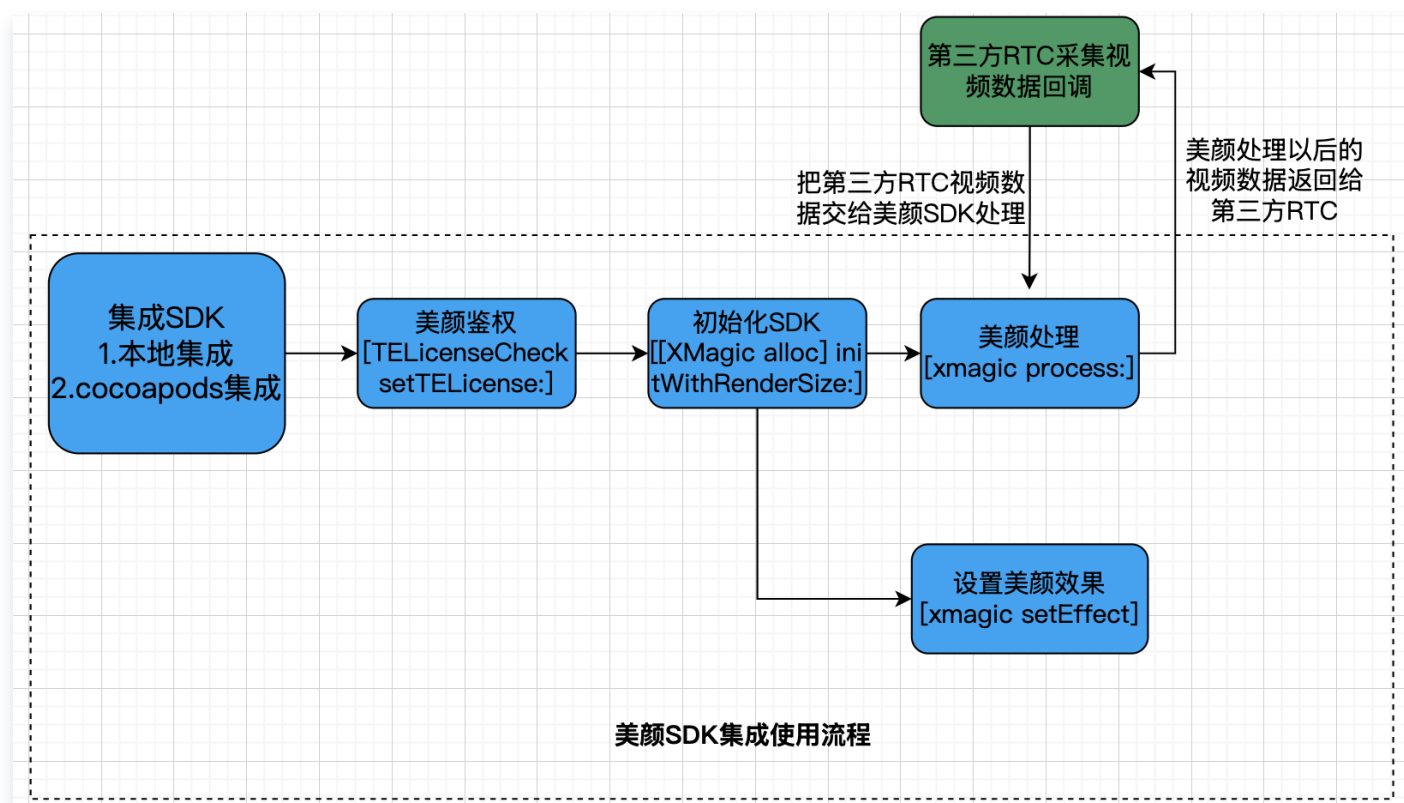
SDK 集成指引（无 UI）

通用集成腾讯特效

iOS

最近更新时间：2025-10-15 15:25:28

美颜特效 SDK 集成整体流程



集成准备

开发者环境要求

- 开发工具 Xcode 11 及以上：从 App Store 下载或单击查看 [下载地址](#)。
- 建议运行环境：
 - 设备要求：iPhone 5 及以上；iPhone 6 及以下前置摄像头最多支持到 720p，不支持 1080p。
 - 系统要求：iOS 10.0 及以上。

导入 SDK

您可以选择使用 CocoaPods 方案，或者先将 SDK 下载到本地，再将其手动导入到您当前的项目中。

使用 CocoaPods

1. 安装 CocoaPods。

在终端窗口中输入如下命令（需要提前在 Mac 中安装 Ruby 环境）：

```
sudo gem install cocoapods
```

2. 创建 Podfile 文件。

进入项目所在路径，输入以下命令之后项目路径下会出现一个 Podfile 文件。

```
pod init
```

3. 编辑 Podfile 文件。

- XMagic 版本在3.0.1以前

根据您的项目需要选择合适的版本，并编辑 Podfile 文件：

- XMagic 普通版。

请按如下方式编辑 Podfile文件：

```
platform :ios, '8.0'

target 'App' do
  pod 'XMagic'
end
```

- XMagic 精简版。

安装包体积比普通版小，但仅支持基础版 A1-00、基础版 A1-01、高级版 S1-00，请按如下方式编辑 Podfile 文件：

```
platform :ios, '8.0'

target 'App' do
  pod 'XMagic_Smart'
end
```

- XMagic 版本在3.0.1及以后：

根据您的项目套餐选择合适的版本，并编辑 Podfile 文件：

```
#请根据你的套餐pod install对应的库
#例如：如果你的套餐是all类型，那么只需要pod 'TencentEffect_All'
#例如：如果你的套餐是S1-04类型，那么只需要pod 'TencentEffect_S1-04'
pod 'TencentEffect_All'
#pod 'TencentEffect_A1-00'
#pod 'TencentEffect_A1-01'
#pod 'TencentEffect_A1-02'
#pod 'TencentEffect_A1-03'
#pod 'TencentEffect_A1-04'
#pod 'TencentEffect_A1-05'
#pod 'TencentEffect_A1-06'
#pod 'TencentEffect_S1-00'
#pod 'TencentEffect_S1-01'
#pod 'TencentEffect_S1-02'
#pod 'TencentEffect_S1-03'
#pod 'TencentEffect_S1-04'
#pod 'TencentEffect_S1-05'
#pod 'TencentEffect_S1-06'
#pod 'TencentEffect_S1-07'
#pod 'TencentEffect_X1-01'
#pod 'TencentEffect_X1-02'
```

4. 更新并安装 SDK。

在终端窗口中输入如下命令以更新本地库文件，并安装 SDK：

```
pod install
```

pod 命令执行完后，会生成集成了 SDK 的 `.xcworkspace` 后缀的工程文件，双击打开即可。

5. 添加美颜资源到实际项目工程中。

下载并解压对应套餐的 [SDK 和美颜资源](#)，将 `resources/motionRes` 文件夹下 `bundle` 资源添加到实际工程中。在 Build Settings 中的 Other Linker Flags 添加 `-ObjC`。

6. 将 Bundle Identifier 修改成与申请的测试授权一致。

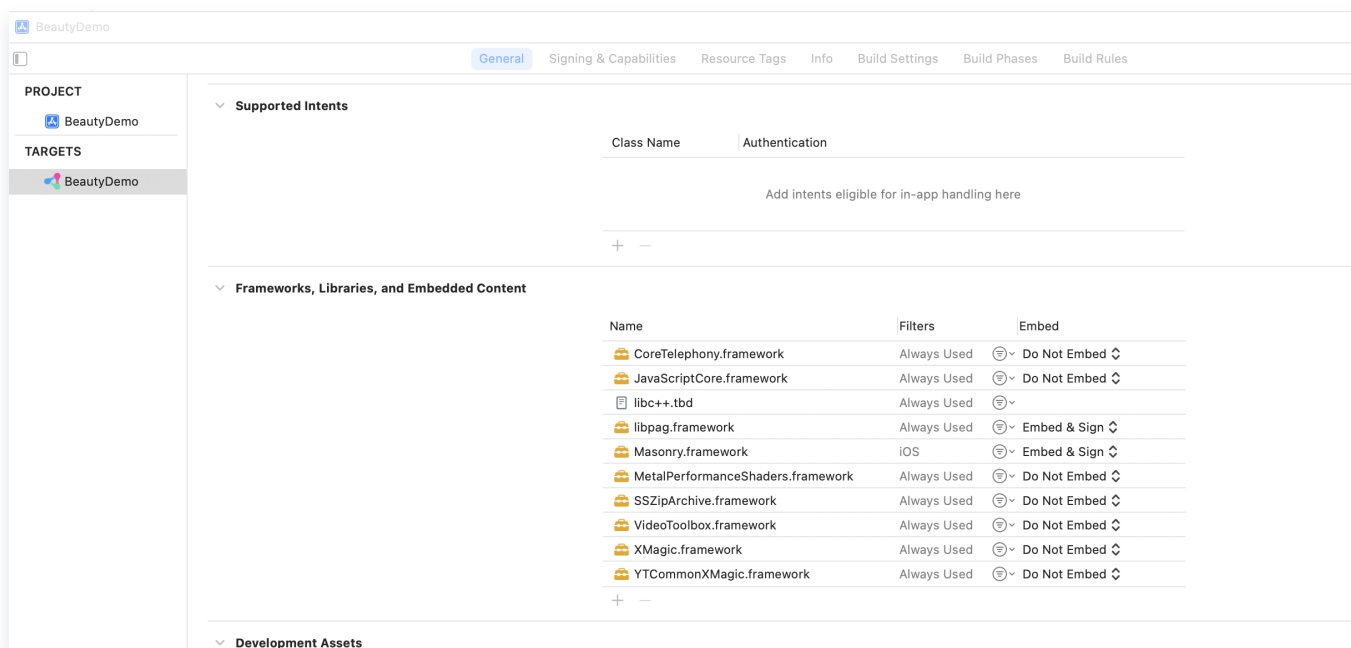
下载 SDK 并手动导入

1. 下载并解压 [SDK 和美颜资源](#)，frameworks 文件夹里面是 sdk、resources 文件夹里面是美颜的 bundle

资源。

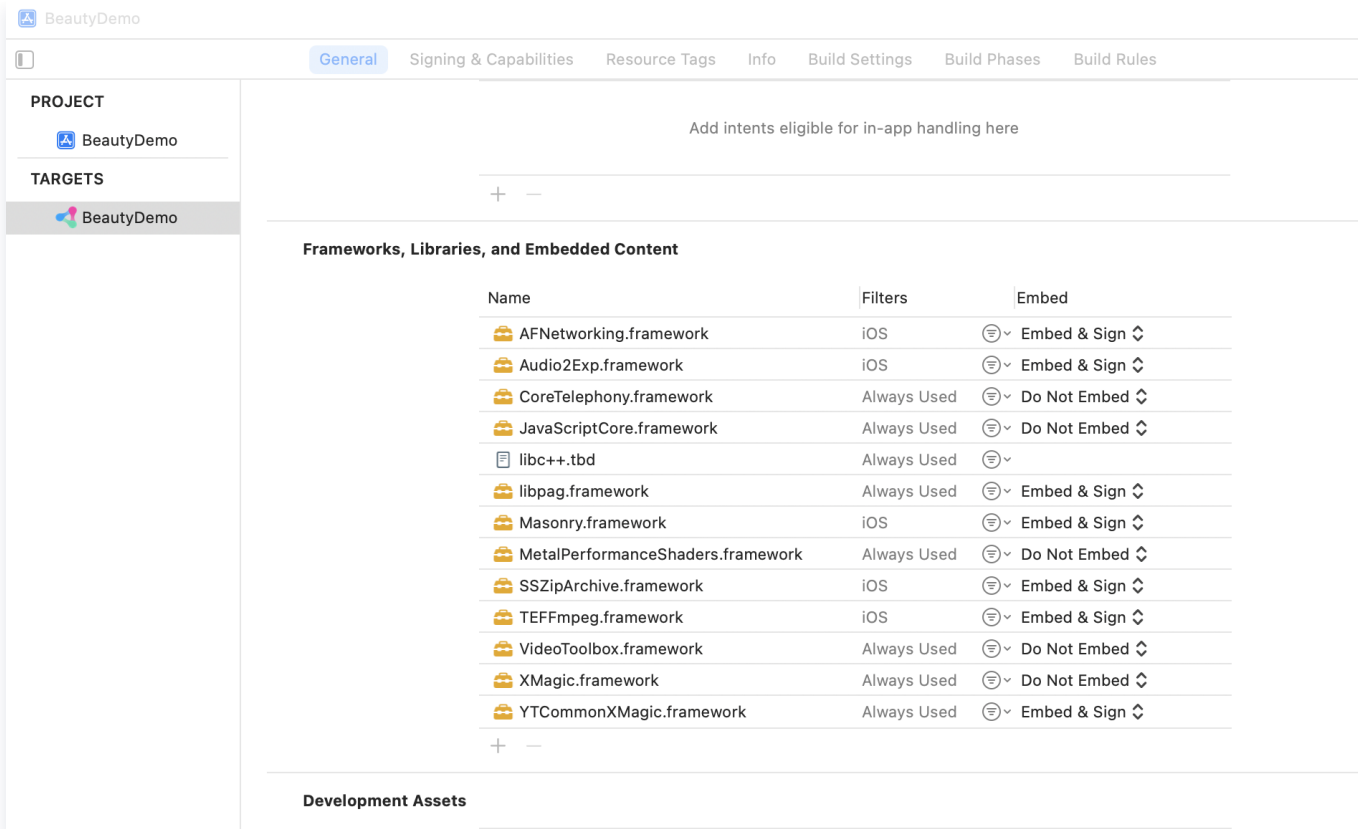
2. SDK 版本在2.5.1以前：

打开您的 Xcode 工程项目，把 frameworks 文件夹里面的 framework 添加到实际工程中，选择要运行的 target，选中 **General** 项，单击 **Frameworks,Libraries,and Embedded Content** 项展开，单击底下的“+”号图标去添加依赖库。依次添加下载的 `XMagic.framework`、`YTCommonXMagic.framework`、`libpag.framework` 及其所需依赖库 `MetalPerformanceShaders.framework`、`CoreTelephony.framework`、`JavaScriptCore.framework`、`VideoToolbox.framework`、`libc++.tbd`，根据需要添加其它工具库 `Masonry.framework`（控件布局库）、`SSZipArchive`（文件解压库）。



SDK 版本在2.5.1及以后：

打开您的 Xcode 工程项目，把 frameworks 文件夹里面的 framework 添加到实际工程中，选择要运行的 target，选中 **General** 项，单击 **Frameworks,Libraries,and Embedded Content** 项展开，单击底下的“+”号图标去添加依赖库。依次添加下载的 `XMagic.framework`、`YTCommonXMagic.framework`、`libpag.framework`、`Audio2Exp.framework`、`TEFFmpeg.framework` (version 3.0.0 以后，改名为：`TECodec.framework`) 及其所需依赖库 `MetalPerformanceShaders.framework`、`CoreTelephony.framework`、`JavaScriptCore.framework`、`VideoToolbox.framework`、`libc++.tbd`，根据需要添加其它工具库 `Masonry.framework`（控件布局库）、`SSZipArchive`（文件解压库）。



3. 把 resources 夹里面的美颜资源添加到实际工程中。
4. 在 Build Settings 中的 Other Linker Flags 添加 `-ObjC`。
5. 将 Bundle Identifier 修改成与申请的测试授权一致。

动态下载集成

为了减少包大小，您可以将 SDK 所需的模型资源和动效资源 MotionRes（部分基础版 SDK 无动效资源）改为联网下载。在下载成功后，将上述文件的路径设置给 SDK。

我们建议您复用 Demo 的下载逻辑，当然，也可以使用您已有的下载服务。动态下载的详细指引，请参见 [SDK 包体瘦身（iOS）](#)。

配置权限

在 Info.plist 文件中添加相应权限的说明，否则程序在 iOS 10 系统上会出现崩溃。请在 Privacy – Camera Usage Description 中开启相机权限，允许 App 使用相机。

集成步骤

步骤一：鉴权

1. 申请授权，得到 LicenseURL 和 LicenseKEY，请参见 [License 指引](#)。

2. 在相关业务模块的初始化代码中设置 URL 和 KEY，触发 license 下载，避免在使用前才临时去下载。也可以在 AppDelegate 的 didFinishLaunchingWithOptions 方法里触发下载。其中，LicenseURL 和 LicenseKey 是控制台绑定 License 时生成的授权信息。

SDK 版本在2.5.1以前， `TELicenseCheck.h` 在 `XMagic.framework` 里面；SDK 版本在2.5.1及以后，`TELicenseCheck.h` 在 `YTCommonXMagic.framework` 里面。

```
[TELicenseCheck setTELicense:LicenseURL key:LicenseKey
completion:^(NSInteger authresult, NSString * _Nonnull errorMsg) {
    if (authresult == TELicenseCheckOk) {
        NSLog(@"鉴权成功");
    } else {
        NSLog(@"鉴权失败");
    }
}];
```

鉴权 errorCode 说明：

错误码	说明
0	成功。Success
-1	输入参数无效，例如 URL 或 KEY 为空
-3	下载环节失败，请检查网络设置
-4	从本地读取的 TE 授权信息为空，可能是 IO 失败引起
-5	读取 VCUBE TEMP License文件内容为空，可能是 IO 失败引起
-6	v_cube.license 文件 JSON 字段不对。请联系腾讯云团队处理
-7	签名校验失败。请联系腾讯云团队处理
-8	解密失败。请联系腾讯云团队处理
-9	TELicense 字段里的 JSON 字段不对。请联系腾讯云团队处理
-10	从网络解析的TE授权信息为空。请联系腾讯云团队处理
-11	把 TE 授权信息写到本地文件时失败，可能是IO失败引起
-12	下载失败，解析本地 asset 也失败
-13	鉴权失败

[其他](#)[请联系腾讯云团队处理](#)

步骤二：加载 SDK (XMagic.framework)

使用腾讯特效 SDK 生命周期大致如下：

1. 加载美颜相关资源。

```
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                              @"root_path":[[NSBundle mainBundle] bundlePath] // LightCore.bundle所
                              在的目录。
};
```

2. 初始化腾讯特效 SDK。

```
/**
previewSize:视图的宽高
assetsDict: 上一步配置的LightCore.bundle及其路径
*/
self.xMagicApi = [[XMagic alloc] initWithRenderSize:previewSize
assetsDict:assetsDict];
```

3. 腾讯特效 SDK 处理每帧数据并返回相应处理结果。

```
/**
以设备摄像头数据输出为例
*/

//sampleBuffer: 设备摄像头输出的数据
-(CMSampleBufferRef)didProcessCPUData:
(CMSampleBufferRef) sampleBuffer{
    CVPixelBufferRef pixelBuffer =
    CMSampleBufferGetImageBuffer(sampleBuffer);
    YTPProcessInput *input = [[YTPProcessInput alloc] init];
    input.pixelData = [[YTImagePixelData alloc] init];
    input.pixelData.data = pixelBuffer;
    input.dataType = kYTImagePixelData;
    YTPProcessOutput *output = [self.xMagicKit process:input];
    if (output.pixelData.data != nil) { //output.pixelData.data: 美颜
SDK处理以后的数据
```

```
        CMSampleBufferRef outSampleBuffer = [self
sampleBufferFromPixelBuffer:output.pixelData.data];
        return outSampleBuffer;
    }
    return nil;
}

//PixelBuffer转sampleBuffer
- (CMSampleBufferRef) sampleBufferFromPixelBuffer:
(CVPixelBufferRef) pixelBuffer
{
    CFRetain(pixelBuffer);
    CMSampleBufferRef outputSampleBuffer = NULL;
    CMSampleTimingInfo timing = {kCMTIME_INVALID, kCMTIME_INVALID,
kCMTIME_INVALID};
    CMVideoFormatDescriptionRef videoInfo = NULL;
    OSStatus result =
CMVideoFormatDescriptionCreateForImageBuffer(NULL, pixelBuffer,
&videoInfo);
    result = CMSampleBufferCreateForImageBuffer(kCFAllocatorDefault,
pixelBuffer, true, NULL, NULL, videoInfo, &timing,
&outputSampleBuffer);
    CFArrayRef attachments =
CMSampleBufferGetSampleAttachmentsArray(outputSampleBuffer, YES);
    CFMutableDictionaryRef dict =
(CFMutableDictionaryRef) CFArrayGetValueAtIndex(attachments, 0);
    CFDictionarySetValue(dict,
kCMSampleAttachmentKey_DisplayImmediately, kCFBooleanTrue);
    CFRelease(videoInfo);
    CFRelease(pixelBuffer);
    return outputSampleBuffer;
}
```

4. 释放腾讯特效 SDK。

```
// 在需要释放SDK资源的地方调用
[self.xMagicApi deinit]
```

! 说明：

完成上述步骤后，用户即可根据自己的实际需求控制展示时机以及其他设备相关环境。

Android

最近更新时间：2025-05-30 17:17:40

开发者环境要求

- 建议使用 Android 7.0 （SDK API Level 24）及以上设备。
- Android Studio 3.5 及以上版本。

Demo 工程：TEBeauty_API_Example

从 github clone 出 [demo工程](#)，其中的 TEBeautyDemo 是含 UI 的 demo 工程，TEBeauty_API_Example 是不含 UI 的 demo 工程。

按照 TEBeauty_API_Example/README 文档中的指引将 TEBeauty_API_Example 运行起来，然后结合本文了解无 UI 集成 SDK 的详细步骤。

集成SDK

⚠ 注意：

Github 上的 demo 工程采用 Maven 方式集成 SDK。

Maven 集成

腾讯特效 SDK 已经发布到 mavenCentral 库，您可以通过配置 gradle 自动下载更新。

1. 在 dependencies 中添加腾讯特效 SDK 的依赖。

```
dependencies {  
    //例如：S1-04套餐如下：  
    implementation 'com.tencent.mediacloud:TencentEffect_S1-04:版本号'  
    //“版本号”可以在官网的“SDK下载/版本历史”页面看到，例如 3.0.0.13。“版本号”也可以使用"latest.release"，  
    //但请注意：这会让您使用的SDK始终保持最新版，在一些变化比较大的版本上可能不符合您的预期，请慎重使用"latest.release"  
}
```

2. 在 defaultConfig 中，指定 App 使用的 CPU 架构。

```
defaultConfig {
```

```
ndk {
    abiFilters "armeabi-v7a", "arm64-v8a"
}
```

说明：
目前特效 SDK 支持 armeabi-v7a 和 arm64-v8a。

3. 单击  **Sync Now**，自动下载 SDK 并集成到工程里。

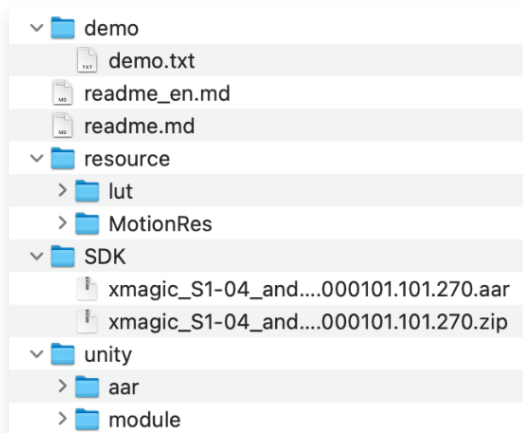
各套餐对应的 Maven 地址

版本	Maven 地址
A1 – 01	implementation 'com.tencent.mediacloud:TencentEffect_A1-01:版本号'
A1 – 02	implementation 'com.tencent.mediacloud:TencentEffect_A1-02:版本号'
A1 – 03	implementation 'com.tencent.mediacloud:TencentEffect_A1-03:版本号'
A1 – 04	implementation 'com.tencent.mediacloud:TencentEffect_A1-04:版本号'
A1 – 05	implementation 'com.tencent.mediacloud:TencentEffect_A1-05:版本号'
A1 – 06	implementation 'com.tencent.mediacloud:TencentEffect_A1-06:版本号'
S1 – 00	implementation 'com.tencent.mediacloud:TencentEffect_S1-00:版本号'
S1 – 01	implementation 'com.tencent.mediacloud:TencentEffect_S1-01:版本号'
S1 – 02	implementation 'com.tencent.mediacloud:TencentEffect_S1-02:版本号'
S1 – 03	implementation 'com.tencent.mediacloud:TencentEffect_S1-03:版本号'
S1 – 04	implementation 'com.tencent.mediacloud:TencentEffect_S1-04:版本号'
S1 – 05	implementation 'com.tencent.mediacloud:TencentEffect_S1-05:版本号'
S1 – 06	implementation 'com.tencent.mediacloud:TencentEffect_S1-06:版本号'
S1 – 07	implementation 'com.tencent.mediacloud:TencentEffect_S1-07:版本号'

手动集成（资源内置）

下载 SDK

下载 SDK，并解压。目录结构如下：



集成

将 SDK 文件夹下的 `xmagic-xxxx.aar` 文件拷贝到您工程 `libs` 目录下。

导入方法

打开 app 模块的 `build.gradle` 添加依赖引用：

```
android{
    ...
    defaultConfig {
        applicationId "修改成与授权license绑定的包名"
        ....
    }
    packagingOptions {
        pickFirst '**/libc++_shared.so'
    }
}

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

 注意：

项目中还需添加如下依赖：

```
dependencies{
    implementation 'com.google.code.gson:gson:2.8.2'
    //版本在2.6.0到3.1.0.2 需要添加
    implementation
    'androidx.exifinterface:exifinterface:1.3.3'
    //3.5.0 到3.9.1 版本需要添加
    implementation 'com.tencent.tav:libpag:4.3.33-noffavc'
    //3.9.2 版本及之后添加
    implementation 'com.tencent.tav:libpag:4.4.24-noffavc'
}
```

如果您想使用其他版本的 pag，请点击 [此处](#) 查看。

手动集成（资源动态下载）

动态下载 assets、so、动效资源指引

- 为了减少包大小，您可以将 SDK 所需的 assets 资源、so 库、以及动效资源 MotionRes（部分基础版 SDK 无动效资源）改为联网下载。在下载成功后，将上述文件的路径设置给 SDK。
- 我们建议您复用 Demo 的下载逻辑，当然，也可以使用您已有的下载服务。动态下载的详细指引，请参见 [SDK 包体瘦身（Android）](#)。

集成素材

如果您的套餐包含动效和滤镜功能，那么需要在 [SDK 下载页面](#) 下载对应的套餐包，解压之后，将 `resource` 目录下的 `lut` 和 `MotionRes` 放置到您工程的 `assets` 目录下：

- 动效： `../src/main/assets/MotionRes`
- 滤镜： `../src/main/assets/lut`

更多素材配置可参考 [素材使用指南](#)。

SDK使用流程

步骤一：鉴权

1. 申请授权，得到 License URL 和 License KEY，请参见 [License 指引](#)。
2. 在相关业务模块的初始化代码中设置 URL 和 KEY，触发 License 下载，避免在使用前才临时去下载。例如我们的 demo 工程是在 Application 的 onCreate 方法里触发下载，但在您的项目中不建议在这里触发，因为此时可能没有网络权限或联网失败率较高，请选择更合适的时机触发 license 下载。

```
//如果仅仅是为了触发下载或更新license，而不关心鉴权结果，则第4个参数传入null。  
TELICENSECHECK.getInstance().setXMagicLicense(context, URL, KEY,  
null);
```

3. 然后在真正要使用美颜功能前，再去做鉴权：

```
TELICENSECHECK.getInstance().setTELICENSE(context, URL, KEY, new  
TELICENSECHECK.Listener() {  
  
    @Override  
    public void onLicenseCheckFinish(int errorCode, String msg)  
    {  
  
        //注意：此回调不一定在调用线程  
        if (errorCode == TELICENSECHECK.ERROR_OK) {  
            //鉴权成功  
        } else {  
            //鉴权失败  
        }  
    }  
});
```

鉴权 errorCode 说明：

错误码	说明
0	成功。Success
-1	输入参数无效，例如 URL 或 KEY 为空
-3	下载环节失败，请检查网络设置
-4	从本地读取的 TE 授权信息为空，可能是 IO 失败引起
-5	读取 VCUBE TEMP License文件内容为空，可能是 IO 失败引起
-6	v_cube.license 文件 JSON 字段不对。请联系腾讯云团队处理

-7	签名校验失败。请联系腾讯云团队处理
-8	解密失败。请联系腾讯云团队处理
-9	TELicense 字段里的 JSON 字段不对。请联系腾讯云团队处理
-10	从网络解析的 TE 授权信息为空。请联系腾讯云团队处理
-11	把TE授权信息写到本地文件时失败，可能是 IO 失败引起
-12	下载失败，解析本地 asset 也失败
-13	鉴权失败，请检查 so 是否在包里，或者已正确设置 so 路径
3004/3005	无效授权。请联系腾讯云团队处理
3015	Bundle Id / Package Name 不匹配。检查您的 App 使用的 Bundle Id / Package Name 和申请的是否一致，检查是否使用了正确的授权文件
3018	授权文件已过期，需要向腾讯云申请续期
其他	请联系腾讯云团队处理

步骤二：资源拷贝

这里所指的资源文件包含两部分：

- SDK 的模型文件，位于 SDK 的 aar 包的 assets 目录。
- 滤镜和动效资源文件，位于 demo 工程的 assets 目录，命名分别是 lut 和 MotionRes。

使用美颜前需要将上述资源拷贝到 app 的私有目录。在未更新 SDK 版本的情况下，只需要拷贝一次。拷贝成功后，您可以在 App 的 SharedPreferences 中记录下来，下次就不用再拷贝了。具体可以参见 demo 工程的

TEMenuActivity.java

```
String resPath = new File(getFilesDir(),
AppConfig.getInstance().getBeautyFileDirName()).getAbsolutePath();
if (!resPath.endsWith(File.separator)) {
    resPath = resPath + File.separator;
}
AppConfig.resPathForSDK = resPath;
AppConfig.lutFilterPath = resPath + "light_material/lut";
AppConfig.motionResPath = resPath + "MotionRes";

new Thread(() -> {
```

```
Context context = getApplicationContext();
int addResult = XmagicApi.addAiModeFilesFromAssets(context,
AppConfig.resPathForSDK);
Log.d(TAG, "copyRes, add ai model files result = " + addResult);

String lutDirNameInAsset = "lut";
boolean result = FileUtil.copyAssets(context, lutDirNameInAsset,
AppConfig.lutFilterPath);
Log.d(TAG, "copyRes, copy lut, result = " + result);

String motionResDirNameInAsset = "MotionRes";
boolean result2 = FileUtil.copyAssets(context,
motionResDirNameInAsset, AppConfig.motionResPath);
Log.d(TAG, "copyRes, copy motion res, result = " + result2);
}).start();
```

步骤三：SDK 初始化及使用方法

1. 快速实现相机（可选）

我们假定您已经实现了相机应用，能正常启动相机，且能将相机的 SurfaceTexture 纹理信息回调到 Activity 用于美颜处理，如下所示：

```
@Override
public int onCustomProcessTexture(int textureId, int textureWidth,
int textureHeight) {
    //美颜SDK在这里处理textureId，为其添加美颜和特效，并返回处理后的新的
    textureID
}
```

如果您尚未实现相机应用，可以参见 demo 工程的 `TECameraBaseActivity.java`，使用 `GLCameraXView` 这个组件，将它添加到您的 Activity 的 layout 中，以快速实现相机预览：

```
<com.tencent.demo.camera.camerax.GLCameraXView
    android:id="@+id/te_camera_layout_camerax_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:back_camera="false"
    app:surface_view="false"
    app:transparent="true" />
```

2. 初始化美颜 SDK，可以在任意线程初始化，建议在子线程中进行。创建时机需要在 License 鉴权成功之后。

```
//AppConfig.resPathForSDK是资源拷贝环节确定的资源路径
mXmagicApi = new XmagicApi(this, AppConfig.resPathForSDK);
```

参数

参数	含义
Context context	上下文
String resDir	资源文件目录，详见请参见 步骤二
OnXmagicPropertyErrorListener errorListener	可选，回调函数实现类，用于回调 SDK 初始化和使用过程中的一些 错误码，错误码含义请参见 API 文档

3. （可选）添加素材提示语回调函数（方法回调有可能运行在子线程），部分素材会提示用户：点点头、伸出手掌、比心，这个回调就是用于展示类似的提示语。

```
mXmagicApi.setTipsListener(new XmagicTipsListener() {
    final XmagicToast mToast = new XmagicToast();
    @Override
    public void tipsNeedShow(String tips, String tipsIcon, int type, int
duration) {
        mToast.show(MainActivity.this, tips, duration);
    }

    @Override
    public void tipsNeedHide(String tips, String tipsIcon, int type) {
        mToast.dismiss();
    }
});
```

4. 美颜 SDK 处理每帧数据并返回相应处理结果。process 方法详细说明见 [API 文档](#)。

```
@Override
public int onCustomProcessTexture(int textureId, int textureWidth,
int textureHeight) {
    return mXmagicApi.process(textureId, textureWidth,
```

```
textureHeight);  
}
```

5. 设置美颜或特效。

- 3.5.0版本及以后使用 `setEffect` 方法 方法详细说明见 [API 文档](#)。
- 3.3.0版本及之前使用 `updateProperty` 方法详细说明见 [API 文档](#)。

```
//3.5.0版本及之后使用此方法  
mXmagicApi.setEffect(String effectName, int effectValue, String  
resourcePath, Map<String, String> extraInfo)  
//例如设置美白属性,强度50  
//mXmagicApi.setEffect(XmagicConstant.EffectName.BEAUTY_WHITEN, 50,  
null, null);  
  
// 可用的入参属性可以从 XmagicResParser.parseRes() 获得  
// 3.3.0版本及之前使用此方法  
@Deprecated  
mXmagicApi.updateProperty(XmagicProperty<?> p);
```

6. 生命周期方法 `onResume`，建议在 `Activity` 的 `onResume()` 方法中调用，调用后会恢复特效里的声音。

```
mXmagicApi.onResume();
```

7. 生命周期方法 `onPause`，建议在 `Activity` 的 `onPause()` 方法调用，调用后会暂停特效里的声音。

```
mXmagicApi.onPause();
```

8. 释放美颜 SDK，在OpenGL环境销毁时调用，需要在 GL 线程中调用，不能在主线程（Activity的onDestroy里）调用，否则可能造成资源泄露，多次进出后引起白屏、黑屏现象。

```
@Override  
public void onGLContextDestroy() {  
    mXmagicApi.onDestroy();  
}
```

步骤四：混淆配置

- 如果您在打 release 包时，启用了编译优化（把 minifyEnabled 设置为 true），会裁掉一些未在 java 层调用的代码，而这些代码有可能会被 native 层调用，从而引起 `no xxx method` 的异常。
- 如果您启用了这样的编译优化，那就要添加这些 keep 规则，防止 xmagic 的代码被裁掉：

```
-keep class com.tencent.xmagic.** { *;}
-keep class org.light.** { *;}
-keep class org.libpag.** { *;}
-keep class org.extra.** { *;}
-keep class com.gyailib.**{ *;}
-keep class com.tencent.cloud.iai.lib.** { *;}
-keep class com.tencent.beacon.** { *;}
-keep class com.tencent.qimei.** { *;}
-keep class androidx.exifinterface.** { *;}
```

附件（SDK 文件结构）：

 说明：

此表格列出了SDK用到的所有文件，可能您的套餐中没有某些文件，但并不影响该套餐功能的使用。

文件类型			说明
assets	audio2exp		avatar 虚拟人语音驱动模型，如果不使用该功能，则无需该模型
	benchmark		机型适配使用
	Light3DPlugin		3D 贴纸使用
	LightBody Plugin	LightBody3DModel.bundle	人体 3D 骨骼点位使用
		LightBodyModel.bundle	美体功能使用
	LightCore		SDK 核心模型资源
	LightHandPlugin		手势贴纸、手部点位能力需要
	LightSegmentPlugin		背景分割能力需要使用
	lut		免费的滤镜资源
demo_xxx_android	-		demo 工程

d_xxxx		
jniLibs	libace_zplan.so	3D 引擎库
	libaudio2exp.so	avatar 虚拟人语音驱动库，如果不使用该功能，则无需该库
	libc++_shared.so	libc++_shared.so 是一个 C++ 标准库的共享库，它提供了一组C++ 标准库函数和类，用于支持 C++ 程序的开发和运行。它在 Android 系统中被广泛使用，是 C++ 应用程序和库的重要组成部分。如果您的工程中已有 C++ 共享库，可以只保留一份
	liblight-sdk.so	light sdk 核心库
	libpag.so	light sdk 依赖的动画文件库
	libtecodec.so	light sdk 依赖的编解码库
	libv8jni.so	light sdk 依赖的用于解析 JavaScript 的库
	libYTCommonXMagic.so	license 鉴权使用
libs	xmagic-xxxx.aar	美颜 SDK 的 aar 文件
MotionRes	2dMotionRes	2D 贴纸
	3dMotionRes	3D 贴纸
	avatarRes	Avatar素材
	ganMotionRes	童趣贴纸
	handMotionRes	手势贴纸
	makeupRes	美妆贴纸
	segmentMotionRes	背景分割贴纸
unity	aar	unity 项目需要使用的桥接 aar
	module	桥接 aar 的原工程

原子能力集成指引

手势识别

最近更新时间：2023-08-03 14:27:07

功能说明

输入相机的 openGL 纹理，实时输出手势检测数据。您可以利用这些数据做一进步的开发。

Android 接口说明

Android 集成指引

Android 集成腾讯特效 SDK，具体请参见 [独立集成腾讯特效](#)。

Android 接口调用

1. 打开手势检测功能开关（XmagicApi.java）。

```
public void setFeatureEnableDisable(String featureName, boolean enable);
```

featureName 填 XmagicConstant.FeatureName.HAND_DETECT，enable 填 true。

2. 设置数据回调（XmagicApi.java）

```
void setAIDataListener(XmagicApi.OnAIDataListener aiDataListener)

public interface OnAIDataListener {
    void onFaceDataUpdated(List<TEFaceData> faceDataList);
    void onHandDataUpdated(List<TEHandData> handDataList);
    void onBodyDataUpdated(List<TEBodyData> bodyDataList);
    void onAIDataUpdated(String data);
}
```

onAIDataUpdated 返回 JSON 结构的 string 数据。

iOS 接口说明

iOS 集成指引

iOS 集成腾讯特效 SDK，具体请参见 [独立集成腾讯特效](#)。

iOS 接口调用

1. 打开手势检测功能开关（Xmagic.h）。

```
- (void)setFeatureEnableDisable:(NSString *_Nonnull)featureName
enable:(BOOL)enable;
```

featureName 填 HAND_DETECT（可在 TDefine.h 中引入），enable 填 true。

2. 设置数据回调（Xmagic.h）

```
- (void)registerSDKEventListener:(id<YTSDKEventListener>
_Nullable)listener;

- (void)onAIEvent:(id)event
{
    NSDictionary *eventDict = (NSDictionary *)event;
    if (eventDict[@"ai_info"] != nil) {
        NSLog(@"ai_info %@", eventDict[@"ai_info"]);
    }
}
```

eventDict[@"ai_info"] 即为返回的 JSON 结构的 string 数据。

回调 JSON 数据说明

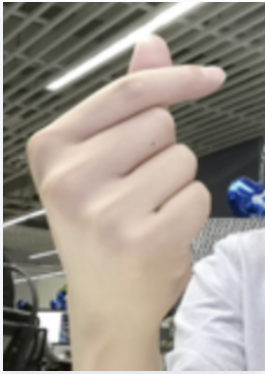
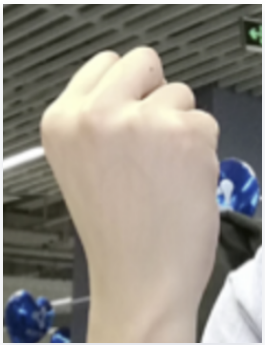
在回调的 JSON 数据中，"hand_info" 里是手势相关的数据，格式如下所示。

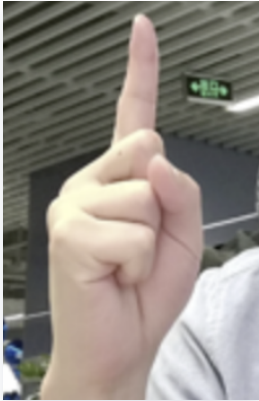
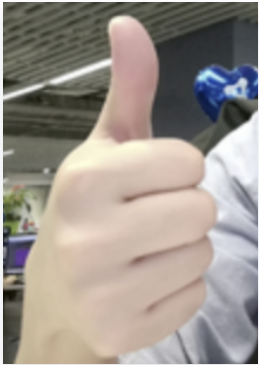
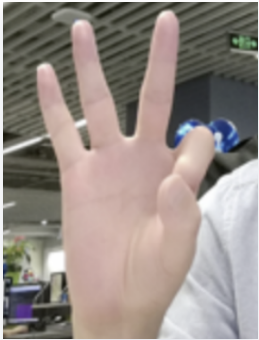
```
"hand_info":{
  "gesture": "PAPER",
  "hand_point_2d": [180.71888732910156, 569.2958984375, ... ,
353.8714294433594, 836.246826171875]
}
```

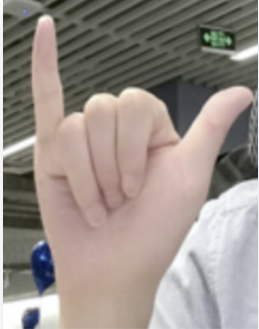
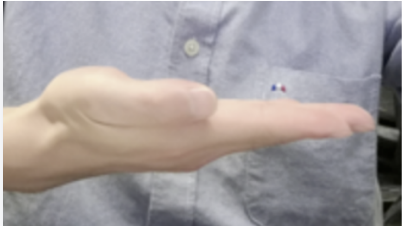
hand_info 中各字段说明如下：

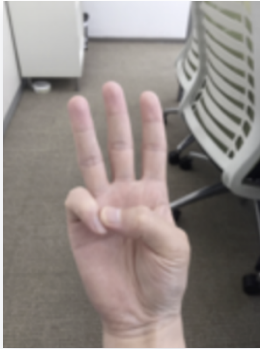
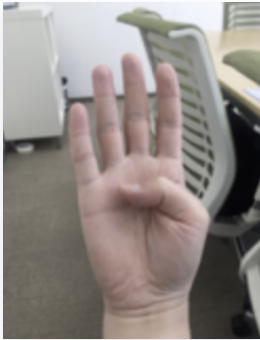
字段	含义
gesture	手势类型名称
hand_point_2d	捕捉到手势的数据信息

目前支持以下手势：

序号	手势	类型名称	示例图
1	比心	HEART	
2	手势5 (open)	PAPER	
3	手势2	SCISSOR	
4	拳头	FIST	

5	手势1	ONE	
6	我爱你	LOVE	
7	点赞	LIKE	
8	OK	OK	
9	摇滚手势	ROCK	

			
10	手势6	SIX	
11	手势8	EIGHT	
12	托	LIFT	
13	手势3	THREE	

			
14	手势4	FOUR	

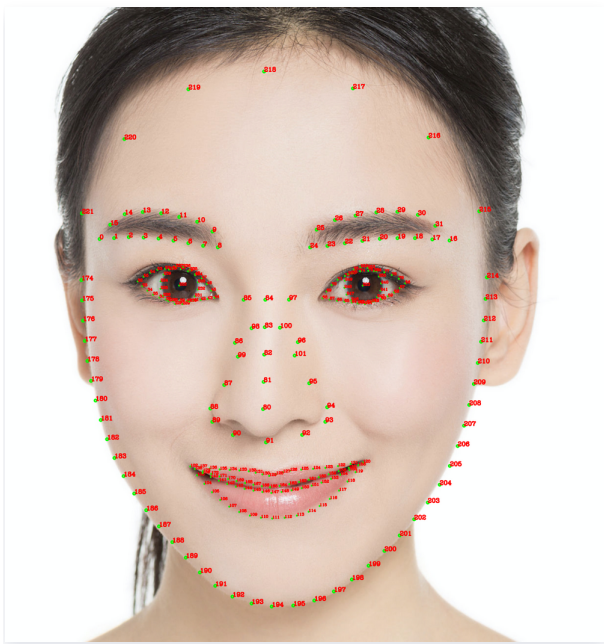
如果为不可识别的手势，则类型名称为 OTHER。

人脸点位

最近更新时间：2023-05-18 16:30:51

人脸检测（识别人脸出框、多人脸、面部遮挡），256个面部关键点位识别与输出。

人脸256点对应索引图



iOS 接口说明

iOS 集成指引

iOS 集成 SDK 指引，请参见 [独立集成腾讯特效](#)。

Xmagic 接口回调注册

```
/// @brief SDK事件监听接口
/// @param listener 事件监听器回调，主要分为AI事件，Tips提示事件，Asset事件
- (void)registerSDKEventListener:(id<YTSDKEventListener>
_Nullable)listener;
```

YTSDKEventListener 回调说明

```
#pragma mark - 事件回调接口
/// @brief SDK内部事件回调接口
@protocol YTSDKEventListener <NSObject>
```

```

/// @brief YTDataUpdate事件回调
/// @param event NSString*格式的回调
- (void)onYTDataEvent:(id _Nonnull)event;
/// @brief AI事件回调
/// @param event dict格式的回调
- (void)onAIEvent:(id _Nonnull)event;
/// @brief 提示事件回调
/// @param event dict格式的回调
- (void)onTipsEvent:(id _Nonnull)event;
/// @brief 资源包事件回调
/// @param event string格式的回调
- (void)onAssetEvent:(id _Nonnull)event;
@end

```

2.6.0及之前版本 设置回调成功后，每一帧人脸事件会回调：

```

- (void)onYTDataEvent:(id _Nonnull)event;

```

3.0.0版本 设置回调成功后，每一帧人脸事件会回调：

```

- (void)onAIEvent:(id _Nonnull)event;
//在onAIEvent方法中可通过下边方法可以获取到数据
NSDictionary *eventDict = (NSDictionary *)event;
if (eventDict[@"ai_info"] != nil) {
    NSLog(@"ai_info %@",eventDict[@"ai_info"]);
}

```

回调 data 是一个 JSON 格式数据，具体含义如下（256点对应上图的位置）：

```

/// @note 字段含义列表
/**
| 字段 | 类型 | 值域 | 说明 |
| :--- | :--- | :--- | :--- |
| trace_id | int | [1,INF) | 人脸id，连续取流过程中，id相同的可以认为是同一张人脸 |
| face_256_point | float | [0,screenWidth或screenHeight] | 共512个数，人脸256个关键点，屏幕左上角为(0,0) |
| face_256_visible | float | [0,1] | 人脸256关键点可见度 |
| out_of_screen | bool | true/false | 人脸是否出框 |

```

```
| left_eye_high_vis_ratio | float | [0,1] | 左眼高可见度点位占比 |  
| right_eye_high_vis_ratio | float | [0,1] | 右眼高可见度点位占比 |  
| left_eyebrow_high_vis_ratio | float | [0,1] | 左眉高可见度点位占比 |  
| right_eyebrow_high_vis_ratio | float | [0,1] | 右眉高可见度点位占比 |  
| mouth_high_vis_ratio | float | [0,1] | 嘴高可见度点位占比 |  
**/  
- (void)onYTDataEvent:(id _Nonnull)event;
```

Android 接口说明

Android 集成指引

Android 集成 SDK 指引，具体请参见 [独立集成腾讯特效](#)。

Xmagic 接口回调注册

设置人脸点位信息等数据回调。

2.6.0及之前版本使用如下方法

```
void setYTDataListener(XmagicApi.XmagicYTDataListener ytDataListener)
```

设置人脸信息等数据回调

```
public interface XmagicYTDataListener {  
    void onYTDataUpdate(String data)  
}
```

3.0.0版本使用如下方法

```
void setAIDataListener(XmagicApi.OnAIDataListener aiDataListener)
```

```
public interface OnAIDataListener {  
    void onFaceDataUpdated(List<TEFaceData> faceDataList);  
    void onHandDataUpdated(List<TEHandData> handDataList);  
    void onBodyDataUpdated(List<TEBodyData> bodyDataList);  
    void onAIDataUpdated(String data); //此方法是3.0.0版本新增方法，数据结构  
    和之前XmagicYTDataListener接口的数据保持一致  
}
```

onYTDataUpdate 和 onAIDataUpdated 返回 JSON string 结构，最多返回5个人脸信息：

```
{
  "face_info": [{
    "trace_id": 5,
    "face_256_point": [
      180.0,
      112.2,
      ...
    ],
    "face_256_visible": [
      0.85,
      ...
    ],
    "out_of_screen": true,
    "left_eye_high_vis_ratio": 1.0,
    "right_eye_high_vis_ratio": 1.0,
    "left_eyebrow_high_vis_ratio": 1.0,
    "right_eyebrow_high_vis_ratio": 1.0,
    "mouth_high_vis_ratio": 1.0
  ],
  ...
}
```

字段含义

字段	类型	值域	说明
trace_id	int	[1,INF)	人脸 ID，连续取流过程中，ID 相同的可以认为是同一张人脸。
face_256_point	float	[0,screenWidth] 或 [0,screenHeight]	共512个数，人脸256个关键点，屏幕左上角为(0,0)。
face_256_visible	float	[0,1]	人脸256关键点可见度。
out_of_screen	bool	true/false	人脸是否出框。

left_eye_high_vis_ratio	float	[0,1]	左眼高可见度点位占比。
right_eye_high_vis_ratio	float	[0,1]	右眼高可见度点位占比。
left_eyebrow_high_vis_ratio	float	[0,1]	左眉高可见度点位占比。
right_eyebrow_high_vis_ratio	float	[0,1]	右眉高可见度点位占比。
mouth_high_vis_ratio	float	[0,1]	嘴高可见度点位占比。

参数

参数	含义
XmagicApi.XmagicYTDataListener ytDataListener	回调函数实现类。

人像分割

最近更新时间：2025-10-09 11:46:20

在直播、会议等场景实现虚拟背景，实时精准分割，支持自定义背景：



Android 集成指引

Android 集成 SDK 指引，请参见 [独立集成美颜特效](#)。

方法

```
setEffect(String effectName, int effectValue, String resourcePath,
Map<String, String> extraInfo)
```

具体参数说明参见 [setEffect](#)。

虚拟背景设置

```
mXmagicApi.setEffect (EffectName.EFFECT_SEGMENTATION, 45,
AppConfig.motionResPath +
"/segmentMotionRes/video_segmentation_blur_45", null);
```

自定义背景设置

```
Map<String, String> extraInfo = new HashMap<>();
extraInfo.put("segType", "custom_background");
extraInfo.put("bgType", "0");
extraInfo.put("bgPath", imgPath);
mXmagicApi.setEffect(EffectName.EFFECT_SEGMENTATION, 0,
AppConfig.motionResPath + "/segmentMotionRes/video_empty_segmentation",
extraInfo);
```

iOS 集成指引

iOS 集成 SDK 指引，请参见 [独立集成美颜特效](#)。

方法

```
- (void)setEffect:(NSString * _Nullable)effectName effectValue:
(int)effectValue resourcePath:(NSString * _Nullable)resourcePath
extraInfo:(NSDictionary * _Nullable)extraInfo;
```

具体参数说明参见 [setEffect](#)。

虚拟背景设置

```
NSString *path = [[[NSBundle mainBundle]
pathForResource:@"segmentMotionRes" ofType:@"bundle"]
stringByAppendingPathComponent:@"video_segmentation_blur_75"];
[_xMagicKit setEffect:EFFECT_MOTION effectValue:0 resourcePath:path
extraInfo:nil];
```

自定义背景设置

```
NSString *resourcePath = [[[NSBundle mainBundle]
pathForResource:@"segmentMotionRes" ofType:@"bundle"]
stringByAppendingPathComponent:@"video_empty_segmentation"];
NSMutableDictionary *dic = [NSMutableDictionary dictionary];
dic[@"segType"] = @"custom_background";
dic[@"bgType"] = 0; //自定义背景的类型。 0表示图片，1表示视频
dic[@"bgPath"] = @" /var/mobile/Containers/Data/Application/06B00BBC-
9060-450F-8D3A-F6028D185682/Documents/MediaFile/image.png"; //自定义背景图
```

片的绝对路径。如果自定义背景选择的是视频，需要对视频进行压缩转码处理，使用压缩转码处理后的绝对路径

```
[_xMagicKit setEffect:EFFECT_SEGMENTATION effectValue:0  
resourcePath:resourcePath extraInfo:dic];
```

人脸属性

Android

最近更新时间：2023-06-01 09:52:59

功能说明

输入一张包含人脸的照片，输出人物面部特征信息，包括眼睛、眉毛、发型、肤色、性别、年龄等。该接口需要联网，SDK会把照片上传到Server端进行解析。

集成指引

首先需要集成腾讯特效SDK，具体请参见 [独立集成腾讯特效](#)。

接口说明

XMagicApi.java

```
public void getFaceFeatureFromPhoto(Bitmap bitmap, FaceFeatureListener listener);
```

- 参数 bitmap：请尽量让人脸位于画面中间，建议画面中只包含一个人脸，如果有多个人脸，SDK会随机选择一个。建议照片的短边大于等于500px，尺寸过小会影响识别效果。
- 参数 FaceFeatureListener，返回识别的结果

```
public interface FaceFeatureListener {  
    void onError(int errCode, String msg);  
    void onFinish(FaceDetailAttributesInfo faceInfo);  
}
```

- onError 回调：解析失败时会回调此接口，错误码如下。

```
public static final int ERROR_NO_AUTH = 1; // 没有权限  
public static final int ERROR_RES_INVALID = 5; // 传入的Avatar素材路径无效  
public static final int ERROR_PHOTO_INVALID = 10; // 读取照片失败  
public static final int ERROR_NETWORK_REQUEST_FAILED = 20; // 网络请求失败  
public static final int ERROR_DATA_PARSE_FAILED = 30; // 网络返回数据解析失败
```

```
public static final int ERROR_ANALYZE_FAILED = 40; //人脸分析失败
public static final int ERROR_AVATAR_SOURCE_DATA_EMPTY = 50; //加载
Avatar源数据失败
```

- onFinish 回调：解析成功时回调此接口，FaceDetailAttributesInfo 说明如下。

```
public static class FaceDetailAttributesInfo {
    public int age;    //[0,100]
    public int emotion;    //0: 自然, 1: 高兴, 2: 惊讶, 3: 生气, 4: 悲
    伤, 5: 厌恶, 6: 害怕
    public Eye eye;    // 眼睛信息
    public Eyebrow eyebrow; // 眉毛信息
    public int gender; // 性别信息。-1:没识别, 0: 男性, 1: 女性。
    public Hair hair;   // 发型信息
    public Hat hat;     // 帽子信息
    public int mask;    // 是否有口罩 -1:没识别, 0:无, 1有。
    public int moustache; // 胡子信息。-1:没识别, 0: 无胡子, 1: 有胡
    子。
    public int nose;    // 鼻子信息。-1:没识别, 0: 朝天鼻, 1: 鹰钩鼻,
    2: 普通, 3: 圆鼻头。
    public int shape;   // 脸型信息。 -1:没识别, 0: 方脸, 1: 三角脸, 2:
    鹅蛋脸, 3: 心形脸, 4: 圆脸。
    public int skin;    // 肤色信息。 -1:没识别, 0: 黄色皮肤, 1: 棕色皮
    肤, 2: 黑色皮肤, 3: 白色皮肤。
    public int smile;   //微笑程度, [0,100]。
}

public static class Eye {
    public int eyelidType; // 识别是否双眼皮。-1:没识别, 0: 无, 1:
    有。
    public int eyeSize;    // 眼睛大小。-1:没识别, 0: 小眼睛, 1: 普通眼
    睛, 2: 大眼睛。
    public int glass;      // 识别是否佩戴眼镜。-1:没识别, 0: 无眼镜,
    1: 普通眼镜, 2: 墨镜
    public int eyeOpen;    // 识别眼睛的睁开、闭合状态。-1:没识别, 0:
    睁开, 1: 闭眼
}

public static class Eyebrow {
```

```
public int eyebrowLength;    //眉毛长短。0：短眉毛，1：长眉毛。
public int eyebrowDensity;   //眉毛浓密。 0：淡眉，1：浓眉。
public int eyebrowCurve;     // 眉毛弯曲。0：不弯，1：弯眉。
}

public static class Hair {
    public int length; //头发长度信息。 0：光头，1：短发，2：中发，3：长
发，4：绑发。
    public int bang;     //刘海信息。 0：无刘海，1：有刘海。
    public int color;    //头发颜色信息。0：黑色，1：金色，2：棕色，3：
灰白色。
}

public static class Hat {
    public int style;    //帽子佩戴状态信息。0：不戴帽子，1：普通帽子，2：
头盔，3：保安帽。
    public int color;    //帽子颜色。0：不戴帽子，1：红色系，2：黄色系，
3：蓝色系，4：黑色系，5：灰白色系，6：混色系子。
}
```

人脸表情

iOS

最近更新时间：2025-09-16 16:49:52

功能说明

输入相机的 openGL 纹理，实时输出人脸52表情 BlendShape 数据，遵循苹果 ARKit 规范，详情请参见 [ARFaceAnchor](#)。您可以利用这些表情数据做一进步的开发，例如传到 Unity 中驱动您的模型。

iOS 集成指引

iOS 集成 SDK 指引，具体请参见 [独立集成腾讯特效](#)。

接口调用

1. 打开功能开关：

```
[self.beautyKit setFeatureEnableDisable:ANIMOJI_52_EXPRESSION
enable:YES];
```

2. 设置人脸点位信息数据回调。

2.6.0及之前版本使用如下方法：

```
//XMagic.h
- (void)registerSDKEventListener:(id<YTSDKEventListener>
_Nullable)listener;

@implementation listener
- (void)onYTDataEvent:(id)event
{
    NSLog(@"YTData %@", event);
}
```

3.0.0版本使用如下方法

```
//XMagic.h
- (void)registerSDKEventListener:(id<YTSDKEventListener>
_Nullable)listener;
```

```
- (void)onAIEvent:(id)event
{
    NSDictionary *eventDict = (NSDictionary *)event;
    if (eventDict[@"ai_info"] != nil) {
        NSLog(@"ai_info %@", eventDict[@"ai_info"]);
    }
}
```

onYTDataUpdate 返回 JSON string 结构，最多返回5个人脸信息：

onAIEvent 中通过eventDict[@"ai_info"] 获取到的 JSON string 结构数据，最多返回5个人脸信息：

```
"face_info": [{
    "trace_id": 5,
    "face_256_point": [
        180.0,
        112.2,
        ...
    ],
    "face_256_visible": [
        0.85,
        ...
    ],
    "out_of_screen": true,
    "left_eye_high_vis_ratio": 1.0,
    "right_eye_high_vis_ratio": 1.0,
    "left_eyebrow_high_vis_ratio": 1.0,
    "right_eyebrow_high_vis_ratio": 1.0,
    "mouth_high_vis_ratio": 1.0,
    "expression_weights": [
        0.12,
        -0.32
        ...
    ]
},
...
]
```

字段含义

- **trace_id**: 人脸 ID, 连续取流过程中, ID 相同的可以认为是同一张人脸。
- **expression_weights**: 实时表情blendshape数据, 数组长度为52, 每个数值取值范围为 0到1.0。

```
{"eyeBlinkLeft", "eyeLookDownLeft", "eyeLookInLeft", "eyeLookOutLeft", "eyeLookUpLeft", "eyeSquintLeft", "eyeWideLeft", "eyeBlinkRight", "eyeLookDownRight", "eyeLookInRight", "eyeLookOutRight", "eyeLookUpRight", "eyeSquintRight", "eyeWideRight", "jawForward", "jawLeft", "jawRight", "jawOpen", "mouthClose", "mouthFunnel", "mouthPucker", "mouthRight", "mouthLeft", "mouthSmileLeft", "mouthSmileRight", "mouthFrownRight", "mouthFrownLeft", "mouthDimpleLeft", "mouthDimpleRight", "mouthStretchLeft", "mouthStretchRight", "mouthRollLower", "mouthRollUpper", "mouthShrugLower", "mouthShrugUpper", "mouthPressLeft", "mouthPressRight", "mouthLowerDownLeft", "mouthLowerDownRight", "mouthUpperUpLeft", "mouthUpperUpRight", "browDownLeft", "browDownRight", "browInnerUp", "browOuterUpLeft", "browOuterUpRight", "cheekPuff", "cheekSquintLeft", "cheekSquintRight", "noseSneerLeft", "noseSneerRight", "tongueOut"}
```

- 其他字段是 [人脸信息](#), 只有当您购买了相关 License 才有那些字段。如果您只想获取表情数据, 请忽略那些字段。

Android

最近更新时间：2025-09-16 17:39:19

功能说明

输入相机的 OpenGL 纹理，实时输出人脸52表情 BlendShape 数据，遵循苹果 ARKit 规范，详情请参见 [ARFaceAnchor](#)。您可以利用这些表情数据做一进步的开发，例如传到 Unity 中驱动您的模型。

Android 集成指引

Android 集成 SDK 指引，具体请参见 [独立集成腾讯特效](#)。

接口调用

1. 打开功能开关：

```
//XmagicApi.java
//featureName = XmagicConstant.FeatureName.ANIMOJI_52_EXPRESSION
public void setFeatureEnableDisable(String featureName, boolean
enable);
```

2. 设置人脸点位信息数据回调。

2.6.0及之前版本使用如下方法：

```
//XmagicApi.java
void setYTDataListener(XmagicApi.XmagicYTDataListener ytDataListener)

public interface XmagicYTDataListener {
    void onYTDataUpdate(String data)
}
```

3.0.0版本使用如下方法：

```
void setAIDataListener(XmagicApi.OnAIDataListener aiDataListener)

public interface OnAIDataListener {
    void onFaceDataUpdated(List<TEFaceData> faceDataList);
    void onHandDataUpdated(List<TEHandData> handDataList);
}
```

```
void onBodyDataUpdated(List<TEBodyData> bodyDataList);  
void onAIDataUpdated(String data); //此方法是3.0.0版本新增方法，数据结  
构和之前XmagicYTDataListener接口的数据保持一致  
}
```

onYTDataUpdate 和 onAIDataUpdated 返回 JSON string 结构，最多返回5个人脸信息：

```
{  
  "face_info": [{  
    "trace_id": 5,  
    "face_256_point": [  
      180.0,  
      112.2,  
      ...  
    ],  
    "face_256_visible": [  
      0.85,  
      ...  
    ],  
    "out_of_screen": true,  
    "left_eye_high_vis_ratio": 1.0,  
    "right_eye_high_vis_ratio": 1.0,  
    "left_eyebrow_high_vis_ratio": 1.0,  
    "right_eyebrow_high_vis_ratio": 1.0,  
    "mouth_high_vis_ratio": 1.0,  
    "expression_weights": [  
      0.12,  
      -0.32  
      ...  
    ]  
  },  
  ...  
]
```

字段含义

- **trace_id**：人脸 ID，连续取流过程中，ID 相同的可以认为是同一张人脸。
- **expression_weights**：实时表情blendshape数据，数组长度为52，每个数值取值范围为 0到1.0。

```
{"eyeBlinkLeft", "eyeLookDownLeft", "eyeLookInLeft", "eyeLookOutLeft", "eyeLookUpLeft", "eyeSquintLeft", "eyeWideLeft", "eyeBlinkRight", "eyeLookDownRight", "eyeLookInRight", "eyeLookOutRight", "eyeLookUpRight", "eyeSquintRight", "eyeWideRight", "jawForward", "jawLeft", "jawRight", "jawOpen", "mouthClose", "mouthFunnel", "mouthPucker", "mouthRight", "mouthLeft", "mouthSmileLeft", "mouthSmileRight", "mouthFrownRight", "mouthFrownLeft", "mouthDimpleLeft", "mouthDimpleRight", "mouthStretchLeft", "mouthStretchRight", "mouthRollLower", "mouthRollUpper", "mouthShrugLower", "mouthShrugUpper", "mouthPressLeft", "mouthPressRight", "mouthLowerDownLeft", "mouthLowerDownRight", "mouthUpperUpLeft", "mouthUpperUpRight", "browDownLeft", "browDownRight", "browInnerUp", "browOuterUpLeft", "browOuterUpRight", "cheekPuff", "cheekSquintLeft", "cheekSquintRight", "noseSneerLeft", "noseSneerRight", "tongueOut"}
```

- 其他字段是 [人脸信息](#)，只有当您购买了相关 License 才有那些字段。如果您只想获取表情数据，请忽略那些字段。

身体点位

iOS

最近更新时间：2023-05-18 16:33:15

功能说明

输入相机的 openGL 纹理，实时输出身体3D数据。您可以利用这些3D数据做一进步的开发，例如传到 Unity 中驱动您的模型。

集成指引

首先需要集成腾讯特效SDK，具体请参见 [独立集成腾讯特效](#)。

接口调用

1. 打开功能开关（XMagic.h）。

```
- (void)setFeatureEnableDisable:(NSString * _Nonnull) featureName
enable:(BOOL) enable;
```

featureName 填 `XmagicConstant.FeatureName.BODY_3D_POINT`。

2. 设置数据回调（XMagic.h）。

2.6.0及之前版本使用如下方法

```
//XMagic.h
- (void)registerSDKEventListener:(id<YTSDKEventListener>
_Nullable)listener;

@implementation listener
- (void)onYTDataEvent:(id)event
{
    NSLog(@"YTData %@", event);
}
@end
```

onYTDataEvent 返回 JSON 结构的 string 数据，其示例如下：

- "face_info" 里是人脸相关的数据，与身体 3D 数据无关，可以忽略。

- "body_3d_info" 里各字段说明见下文

3.0.0版本使用如下方法

```
//XMagic.h
- (void)registerSDKEventListener:(id<YTSDKEventListener>
_Nullable)listener;

- (void)onAIEvent:(id)event
{
    NSDictionary *eventDict = (NSDictionary *)event;
    if (eventDict[@"ai_info"] != nil) {
        NSLog(@"ai_info %@",eventDict[@"ai_info"]);
    }
}
```

eventDict[@"ai_info"] 返回 JSON 结构的 string 数据，其示例如下：

- "face_info" 里是人脸相关的数据，与身体 3D 数据无关，可以忽略。
- "body_3d_info" 里各字段说明见下文

身体点位及点位数据说明

相关说明请参见 [身体点位及数据说明](#)。

Android

最近更新时间：2023-05-18 16:33:57

功能说明

输入相机的 openGL 纹理，实时输出身体3D数据。您可以利用这些3D数据做一进步的开发，例如传到 Unity 中驱动您的模型。

Android 集成指引

首先需要集成腾讯特效SDK，具体请参见 [独立集成腾讯特效](#)。

接口调用

1. 打开功能开关（XmagicApi.java）。

```
public void setFeatureEnableDisable(String featureName, boolean enable);
```

featureName 填 `XmagicConstant.FeatureName.BODY_3D_POINT`。

2. 设置数据回调（XmagicApi.java）。

2.6.0及之前版本使用如下方法

```
void setYTDataListener(XmagicApi.XmagicYTDataListener ytDataListener)

public interface XmagicYTDataListener {
    void onYTDataUpdate(String data)
}
```

onYTDataUpdate 返回 JSON 结构的 string 数据，其示例如下：

- "face_info" 里是人脸相关的数据，与身体 3D 数据无关，可以忽略。
- "body_3d_info" 里各字段说明见下文。

3.0.0版本使用如下方法

```
void setAIDataListener(XmagicApi.OnAIDataListener aiDataListener)

public interface OnAIDataListener {
    void onFaceDataUpdated(List<TEFaceData> faceDataList);
```

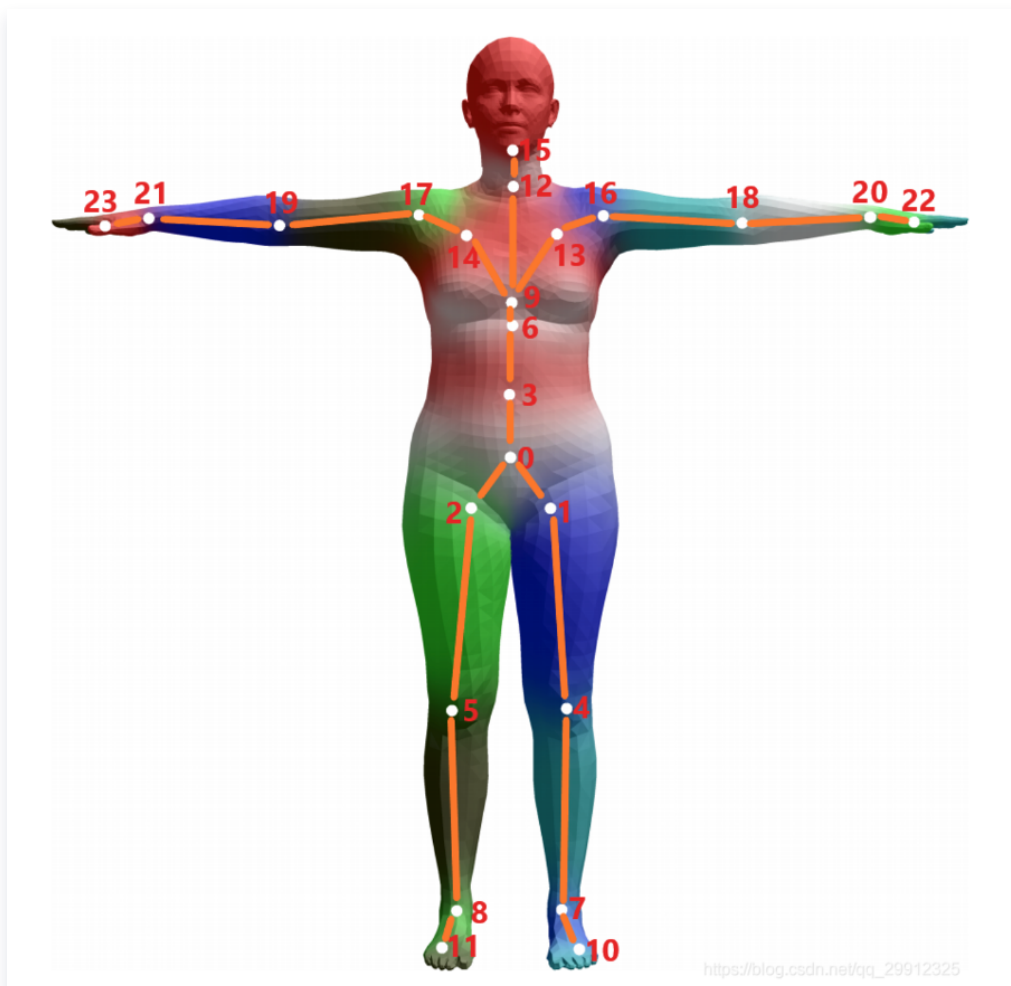
```
void onHandDataUpdated(List<TEHandData> handDataList);  
void onBodyDataUpdated(List<TEBodyData> bodyDataList);  
void onAIDataUpdated(String data); //此方法是3.0.0版本新增方法，数据结  
构和之前XmagicYTDataListener接口的数据保持一致  
}
```

onAIDataUpdated 返回 JSON 结构的 string 数据，其示例如下：

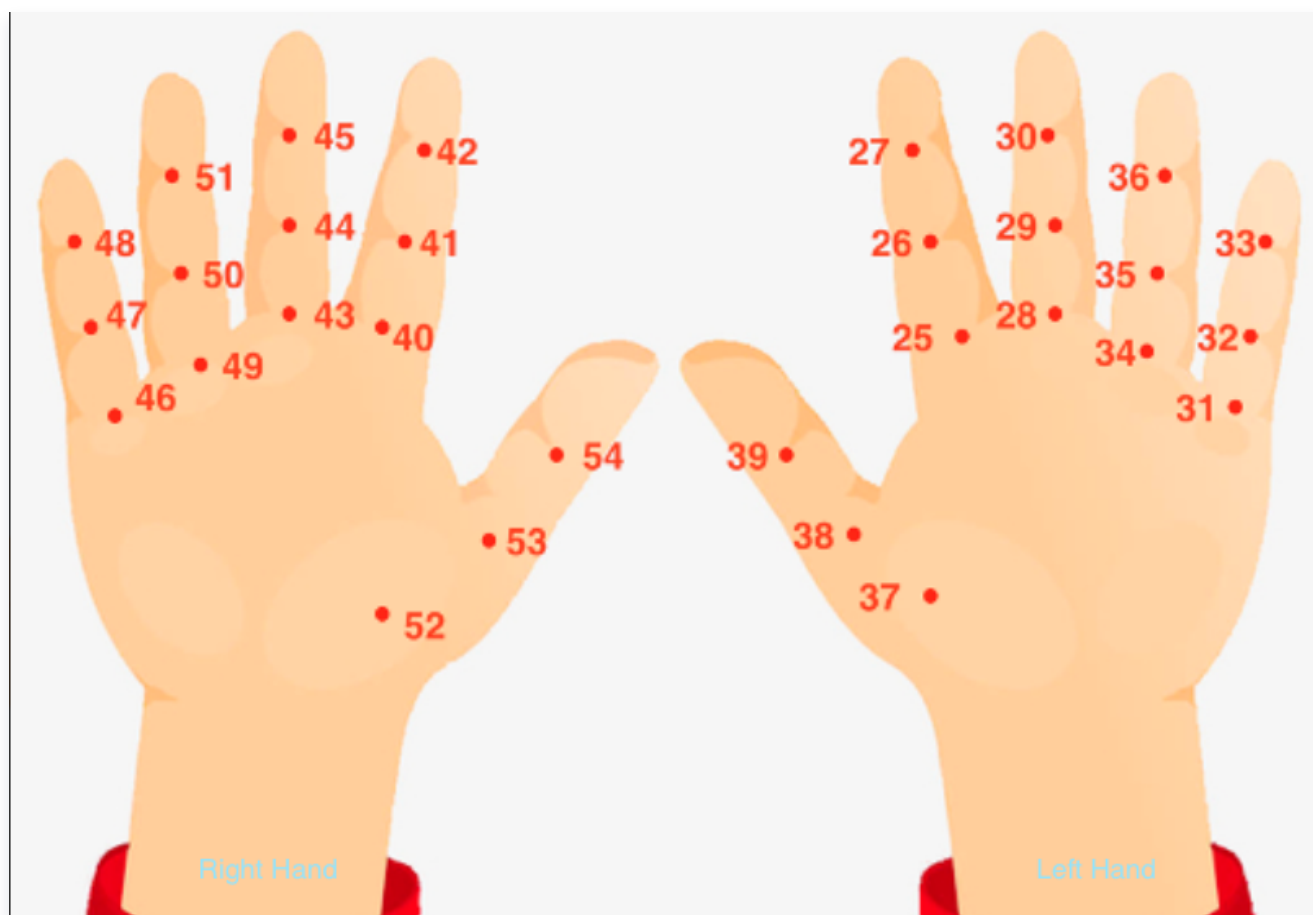
- "face_info" 里是人脸相关的数据，与身体 3D 数据无关，可以忽略。
- "body_3d_info" 里各字段说明见下文。

身体点位及点位数据说明

- 标准 SMPL 点位定义



- 标准 SMPLX 手部骨骼点位定义



SDK 输出的 JSON 数据示例如下：

```
{
  "face_info": [{ ... }],
  "body_3d_info": {
    "imageHeight": 652,
    "imageWidth": 320,
    "items": [{
      "index": 1,
      "pose": [-0.014862474985420704, 0.72017294, ...],
      "position_x": [150.62857055664062, 190.150, ...],
      "position_y": [605.454833984375, 655.21258, ...],
      "position_z": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      "rotation": [{
        "data": [0.9990578889846802, 0.0258054, ...],
        ...
      },
      {
        "data": [0.9546145796775818, 0.2637160, ...]
      }
    ]
  }
}
```

body_3d_info 里各个字段说明如下：

- imageWidth, imageHeight：输入给SDK的图像的宽高
- items：数组，目前只有一个元素
- index：保留位，目前可以忽略它
- pose：
 - (1) [0,2]位置，人体位置，以相机为中心，人体根骨骼的3D位置xyz
 - (2) [3,12]位置，人体形态，10个float数，以标准SMPL的10套不同mesh为基底，组合得到人体型的估计
 - (3) [13]位置，Focal_length，固定值为5000
 - (4) [14,29]位置，OpenGL投影矩阵，基于focal_length得到的在3D空间中渲染物体的投影矩阵。4X4投影矩阵在算法内部计算方式：

```
matrix={
  2 * focal_length / img_wid, 0, 0, 0,
  0, 2 * focal_length / img_hei, 0,0,
  0,0, (zf + zn) / (zn - zf), -1,
  0, 0, (2.0f * zf * zn) / (zn - zf), 0};
}
```

(5) [30,33]位置，接地数据，脚是否踩地，左脚跟、左脚尖、右脚跟、右脚尖

• position_x,position_y,position_z:

(1) [0,23]位置，人体2D点位，见上文的图1, 2D点的 position_z 都是0

(2) [24,47]位置，人体3D点位，见上文的图1

• rotation

(1) [0,23]位置，人体骨骼旋转四元数，每个四元数的属性顺序是 wxyz

(2) [25,54]位置，手部骨骼旋转四元数，左手15个，右手15个，每个四元数的属性顺序是 wxyz

骨骼的不同命名方式及对应关系

序号	Bone Names	Bone Names 2
0	"pelvis",	"Hips"
1	"left_hip",	"LeftUpLeg"
2	"right_hip",	"RightUpLeg"
3	"spine1",	"Spine"
4	"left_knee",	"LeftLeg"
5	"right_knee",	"RightLeg"
6	"spine2",	"Spine1"
7	"left_ankle",	"LeftFoot"
8	"right_ankle",	"RightFoot"
9	"spine3",	"Spine2"
10	"left_foot",	""
11	"right_foot",	""
12	"neck",	"Neck"
13	"left_collar",	"LeftShoulder"
14	"right_collar",	"RightShoulder"
15	"head",	"Head"
16	"left_shoulder",	"LeftArm"
17	"right_shoulder",	"RightArm"
18	"left_elbow",	"LeftForeArm"
19	"right_elbow",	"RightForeArm"
20	"left_wrist",	"LeftHand"
21	"right_wrist",	"RightHand"
22	"left_hand"	""
23	"right_hand"	""
25	"left_index1"	IndexFinger1_L
26	"left_index2"	IndexFinger2_L
27	"left_index3"	IndexFinger3_L
28	"left_middle1"	MiddleFinger1_L
29	"left_middle2"	MiddleFinger2_L
30	"left_middle3"	MiddleFinger3_L
31	"left_pinky1"	PinkyFinger1_L

32	"left_pinky2"	PinkyFinger2_L
33	"left_pinky3"	PinkyFinger3_L
34	"left_ring1"	RingFinger1_L
35	"left_ring2"	RingFinger2_L
36	"left_ring3"	RingFinger3_L
37	"left_thumb1"	ThumbFinger1_L
38	"left_thumb2"	ThumbFinger2_L
39	"left_thumb3"	ThumbFinger3_L
40	"right_index1"	IndexFinger1_R
41	"right_index2"	IndexFinger2_R
42	"right_index3"	IndexFinger3_R
43	"right_middle1"	MiddleFinger1_R
44	"right_middle2"	MiddleFinger2_R
45	"right_middle3"	MiddleFinger3_R
46	"right_pinky1"	PinkyFinger1_R
47	"right_pinky2"	PinkyFinger2_R
48	"right_pinky3"	PinkyFinger3_R
49	"right_ring1"	RingFinger1_R
50	"right_ring2"	RingFinger2_R
51	"right_ring3"	RingFinger3_R
52	"right_thumb1"	ThumbFinger1_R
53	"right_thumb2"	ThumbFinger2_R
54	"right_thumb3"	ThumbFinger3_R

语音转表情

Android

最近更新时间：2023-02-27 12:20:09

功能说明

输入音频数据，输出苹果 ARKit 标准的52表情数据，请参见 [ARFaceAnchor](#)。您可以利用这些表情数据做一进步的开发，例如传到 Unity 中驱动您的模型。

接入方式

方式1：通过腾讯特效 SDK 接入

语音转表情集成在腾讯特效SDK中，因此第一步需要按照腾讯特效文档进行接入。

- 1. 下载 [腾讯特效 SDK 完整版](#)。
- 2. 参考 [独立集成腾讯特效](#) 文档完成集成。

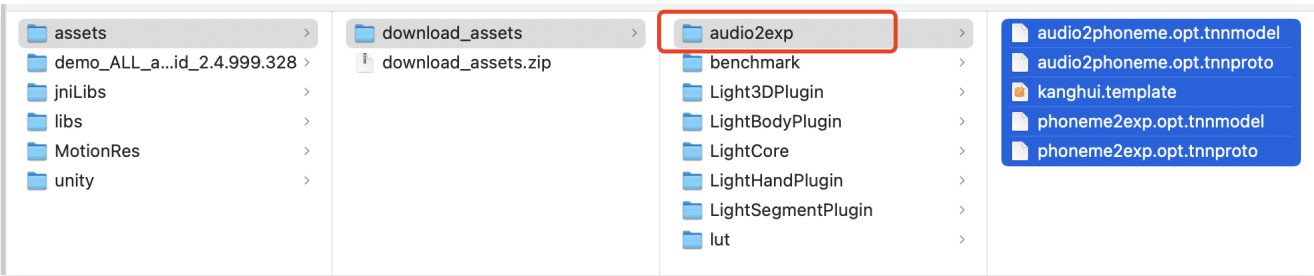
方式2：通过独立的语音转表情 SDK 接入

如果您只需要语音转表情，不需要用到腾讯特效 SDK 的任何能力，则可以考虑使用独立的语音转表情 SDK，aar 包约6MB左右。请联系我们的架构师或销售获取此 SDK。

接入步骤

- 1. 设置 License，请参见 [鉴权](#)。
- 2. 配置模型文件：请将必需的模型文件从 assets 拷贝到 app 的私有目录，例如：

`context.getFilesDir() + "/my_models_dir/audio2exp"`，然后在调用 Audio2ExpApi 的 `init(String modelPath)` 接口时，传入参数 `context.getFilesDir() + "/my_models_dir"` 模型文件在 SDK 包里，位置如下：



接口说明

接口	说明
<code>public int Audio2ExpApi.init(String</code>	初始化，传入模型路径，见上文说明。返回值为0表示成功

modelPath);	
<pre>public float[] Audio2ExpApi.parseAudio(float[] inputData);</pre>	输入的是音频数据，要求单通道，16K采样率，数组长度为267（即267个采样点），输出的数据是长度为52的float数组，表示52表情基，取值为0到1之间，顺序为 苹果标准顺序 <pre>{ "eyeBlinkLeft", "eyeLookDownLeft", "eyeLookInLeft", "eyeLookOutLeft", "eyeLookUpLeft", "eyeSquintLeft", "eyeWideLeft", "eyeBlinkRight", "eyeLookDownRight", "eyeLookInRight", "eyeLookOutRight", "eyeLookUpRight", "eyeSquintRight", "eyeWideRight", "jawForward", "jawLeft", "jawRight", "jawOpen", "mouthClose", "mouthFunnel", "mouthPucker", "mouthRight", "mouthLeft", "mouthSmileLeft", "mouthSmileRight", "mouthFrownRight", "mouthFrownLeft", "mouthDimpleLeft", "mouthDimpleRight", "mouthStretchLeft", "mouthStretchRight", "mouthRollLower", "mouthRollUpper", "mouthShrugLower", "mouthShrugUpper", "mouthPressLeft", "mouthPressRight", "mouthLowerDownLeft", "mouthLowerDownRight", "mouthUpperUpLeft", "mouthUpperUpRight", "browDownLeft", "browDownRight", "browInnerUp", "browOuterUpLeft", "browOuterUpRight", "cheekPuff", "cheekSquintLeft", "cheekSquintRight", "noseSneerLeft", "noseSneerRight", "tongueOut" }</pre>
<pre>public int Audio2ExpApi.release();</pre>	使用完毕后调用，释放资源

集成代码示例

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    findViewById(R.id.button).setOnClickListener(new
OnClickListener() {
        @Override
        public void onClick(View view) {

            TELicenseCheck.getInstance().setTELicense(MainActivity.this, licenseUrl,
licenseKey, new TELicenseCheckListener() {
                @Override
                public void onLicenseCheckFinish(int
errorCode, String s) {
```

```
                                Log.d(TAG,
"onLicenseCheckFinish: errorCode = "+errorCode+",msg="+s);
                                if (errorCode ==
TELicenseCheck.ERROR_OK) {
                                    //license check success
                                    Audio2ExpApi
audio2ExpApi = new Audio2ExpApi();
                                    int err =
audio2ExpApi.init(MainActivity.this.getFilesDir() +"/models");
                                    Log.d(TAG,
"onLicenseCheckFinish: err="+err);
                                    //TODO start record and
                                    parse audio data
                                } else {
                                    // license check failed
                                }
                            }
                        });
                    }
                });
            }
        }
```

❗ 说明

完整的示例代码请参考 [美颜特效 SDK demo 工程](#)。

- 录音可以参考 `com.tencent.demo.avatar.audio.AudioCapturer`。
- 接口使用可以参考：`com.tencent.demo.avatar.activity.Audio2ExpActivity` 及其相关类。

iOS

最近更新时间：2023-02-27 12:15:15

功能说明

输入音频数据，输出苹果 ARKit 标准的52表情数据，请参见 [ARFaceAnchor](#)。您可以利用这些表情数据做进一步的开发，例如传到 Unity 中驱动您的模型。

接入方式

方式1：与腾讯特效 SDK 一起使用

1. 语音转表情sdk结合腾讯特效SDK中，因此第一步需要按照腾讯特效文档进行接入。
2. 下载 [腾讯特效 SDK 完整版](#)。
3. 参考 [独立集成腾讯特效](#) 文档完成集成。
4. 将 [腾讯特效 SDK 完整版](#) 内的Audio2Exp.framework 拉入项目，并且在项目的target->General->Frameworks,Libraries,and Embedded Content处设置为Embed & Sign。

方式2：通过独立的语音转表情 SDK 接入

如果您只需要语音转表情，不需要用到腾讯特效 SDK 的任何能力，则可以考虑使用独立的语音转表情 SDK，Audio2Exp.framework 包约7MB左右。项目引入Audio2Exp.framework、YTCommonXMagic.framework两个动态库，并且在项目的target->General->Frameworks,Libraries,and Embedded Content处设置为Embed & Sign

接入步骤

1. 设置 License，请参见 [鉴权](#)。
2. 配置模型文件：请将必需的模型文件audio2exp.bundle拷贝到工程目录，然后在调用 Audio2ExpApi 的 initWithModelPath: 接口时，传入参数 audio2exp.bundle" 模型文件所在的路径。

接口说明

接口	说明
+ (int)initWithModelPath:(NSString*)modelPath;	初始化，传入模型路径，见上文说明。返回值为0表示成功
+ (NSArray*)parseAudio:	输入的是音频数据，要求单通道，16K采样率，数组长度为267（即267个采样点），输出的数据是长度为52的float数组，表示52表情基，取值为0到1之间，顺序

(NSArray *)inputData;	为 苹果标准顺序 { "eyeBlinkLeft", "eyeLookDownLeft", "eyeLookInLeft", "eyeLookOutLeft", "eyeLookUpLeft", "eyeSquintLeft", "eyeWideLeft", "eyeBlinkRight", "eyeLookDownRight", "eyeLookInRight", "eyeLookOutRight", "eyeLookUpRight", "eyeSquintRight", "eyeWideRight", "jawForward", "jawLeft", "jawRight", "jawOpen", "mouthClose", "mouthFunnel", "mouthPucker", "mouthRight", "mouthLeft", "mouthSmileLeft", "mouthSmileRight", "mouthFrownRight", "mouthFrownLeft", "mouthDimpleLeft", "mouthDimpleRight", "mouthStretchLeft", "mouthStretchRight", "mouthRollLower", "mouthRollUpper", "mouthShrugLower", "mouthShrugUpper", "mouthPressLeft", "mouthPressRight", "mouthLowerDownLeft", "mouthLowerDownRight", "mouthUpperUpLeft", "mouthUpperUpRight", "browDownLeft", "browDownRight", "browInnerUp", "browOuterUpLeft", "browOuterUpRight", "cheekPuff", "cheekSquintLeft", "cheekSquintRight", "noseSneerLeft", "noseSneerRight", "tongueOut" }
+ (int)releaseSdk	使用完毕后调用，释放资源

集成代码示例

```
// 初始化语音转表情sdk
NSString *path = [[NSBundle mainBundle] pathForResource:@"audio2exp"
ofType:@"bundle"];
int ret = [Audio2ExpApi initWithModelPath:path];
// 语音数据转52表情数据
NSArray *emotionArray = [Audio2ExpApi parseAudio:floatArr];
// 释放sdk
[Audio2ExpApi releaseSdk];

// 结合腾讯特效sdk xmgai使用
// 使用对应的资源初始化美颜sdk
self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize
assetsDict:assetsDict];
// 加载avatar素材
[self.beautyKit loadAvatar:bundlePath exportedAvatar:nil
completion:nil];
// 将52表情数据传入美颜sdk 就能看到效果
[self.beautyKit updateAvatarByExpression:emotionArray];
```

❗ 说明：

完整的示例代码请参考 [美颜特效 SDK demo 工程](#)。

- 录音可以参考 `TXCAudioRecorder`。

- 接口使用可以参考：`VoiceViewController` 及其相关类。