实时音视频

客户端高级功能

产品文档

腾讯云

文档目录

客户端高级功能
显示房间指定人的视频

最近更新时间：2018-10-19 16:40:06

本文将指导您的客户端有选择的看到房间内用户画面。

Android

效果图

下图中，当前除老师外，共有四名学生开启了视频，老师可以选择只看到其中的两名学生的视频画面。

源码下载

在此我们提供以下所讲到的完整 Demo 代码，如有需要请您自行下载。

Demo 代码下载

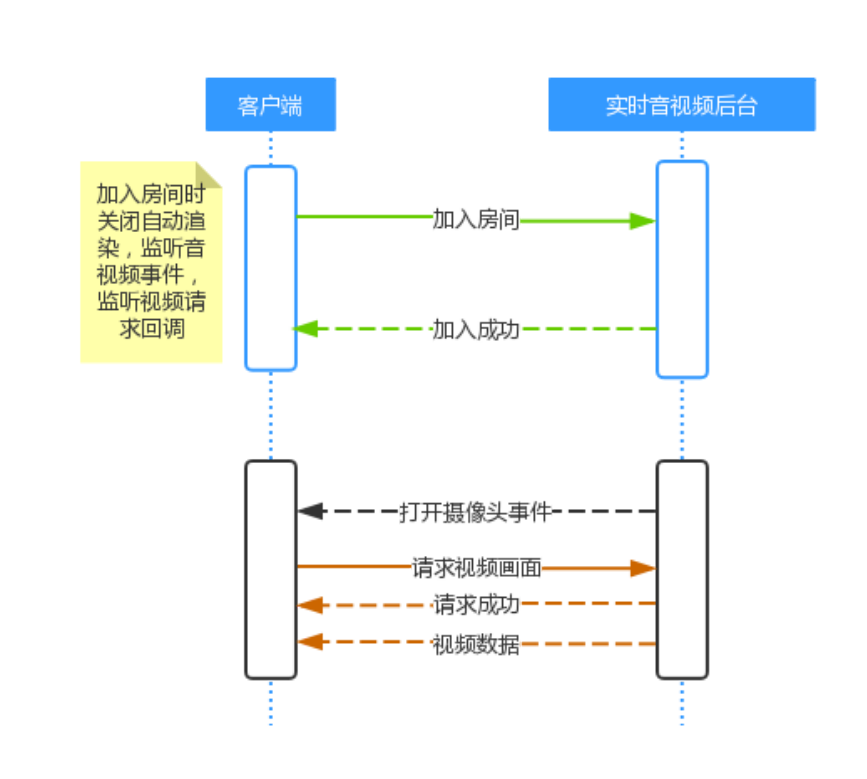
相关概念

视频数据类型

腾讯云支持一个帐号同时上行一路主流和一路辅流，这里用于区分视频流的来源称为视频类型，目前支持的视频类型有： 摄像头，屏幕分享，播片（PC 端产生）。

常量	视频类型	流类型	描述
CommonConstants.Const_VideoType_Camera	摄像头	主流	通过摄像头采集数据产生
CommonConstants.Const_VideoType_Screen	屏幕分享	辅流	通过分享屏幕产生
CommonConstants.Const_VideoType_File	播片	辅流	通过播放视频文件产生

流程图



具体实现

修改进房配置

要实现手动选择用户视频画面，需要在进房间中配置关闭自动渲染，并监听音视频回调(可参考 音视频事件与成员状态),以及视频请求回调:

```
ILiveRoomOption option = new ILiveRoomOption()
.autoRender(false)
.exceptionListener(this)
.roomDisconnectListener(this)
.setRoomMemberStatusLisenter(this) // 监听房间内音视频事件
.setRequestViewLisenter(this) // 监听视频请求回调
.autoCamera(true)
.autoMic(true);
```

请求视频画面

在收到用户的 has camera 或 has screen 事件(如果有播片，还有处理 has file 事件)时，调用接口去请求用户视频画面:

```
public boolean onEndpointsUpdateInfo(int eventid, String[] updateList) {
switch (eventid){
case ILiveConstants.TYPE_MEMBER_CHANGE_HAS_CAMERA_VIDEO:
for (String userId : updateList){
// 请求视频画面
ILiveSDK.getInstance().getContextEngine().requestUserVideoData(userId,
CommonConstants.Const_VideoType_Camera);
}
break;
case ILiveConstants.TYPE_MEMBER_CHANGE_HAS_SCREEN_VIDEO:
for (String userId : updateList){
// 请求视频画面
ILiveSDK.getInstance().getContextEngine().requestUserVideoData(userId,
CommonConstants.Const_VideoType_Screen);
}
break;
}

// 这里需要返回false
return false;
}
```

渲染视频画面

在视频请求成功后，调用渲染控件的接口来进行渲染：

```
public void onComplete(String[] userIdList, AVView[] viewList, int count, int result, String errMsg) {
if (ILiveConstants.NO_ERR == result){
for (int i=0; i<userIdList.length; i++){
avRootView.renderVideoView(true, userIdList[i], viewList[i].videoSrcType, true);
}
}else {
// Todo 失败处理
}
}
```

API说明

requestUserVideoData

属于 ContextEngine 的方法，用于请求视频画面，参数说明如下:

名称	类型	描述
userId	String	需要请求的用户 userId
srcType	int	视频类型

removeUserVideoData

属于 ContextEngine 的方法，用于取消请求视频画面，参数说明如下:

名称	类型	描述
userId	String	需要请求的用户userId
srcType	int	视频类型

renderVideoView

属于渲染控件 AVRootView 的方法，用于渲染一路视频画面，参数说明如下:

名称	类型	描述
bHasVideo	boolean	当前是否已有画面，建议填 true
userId	String	需要请求的用户 userId
srcType	int	视频类型
bAutoRender	boolean	是否自动渲染(自动空闲 AVVideoView 渲染)，建议填 true

iOS

iOS 中视频渲染可以直接在音视频事件回调中选择处理：
具体操作参考 *iOS 创建房间*

PC

PC 中视频渲染可以直接在音视频事件回调中选择处理：
具体操作参考 *PC 创建房间*

常见问题

- 请求视频画面是否需要取消?

> sdk内部是在音视频事件的has camera和no camera中成对使用，即has camera时请求，no camera时取消请求，推荐用户也这样实现，避免该用户断开后重新上行，画面会自动渲染出来。

定制视频画面展示

最近更新时间：2018-06-25 15:44:04

本文将指导您的客户端定制自己的视频布局。

Android

效果图

以下分别为二分、四分布局，用户可以根据自己的需求定制自己的布局。

二分布局	四分布局
	

源码下载

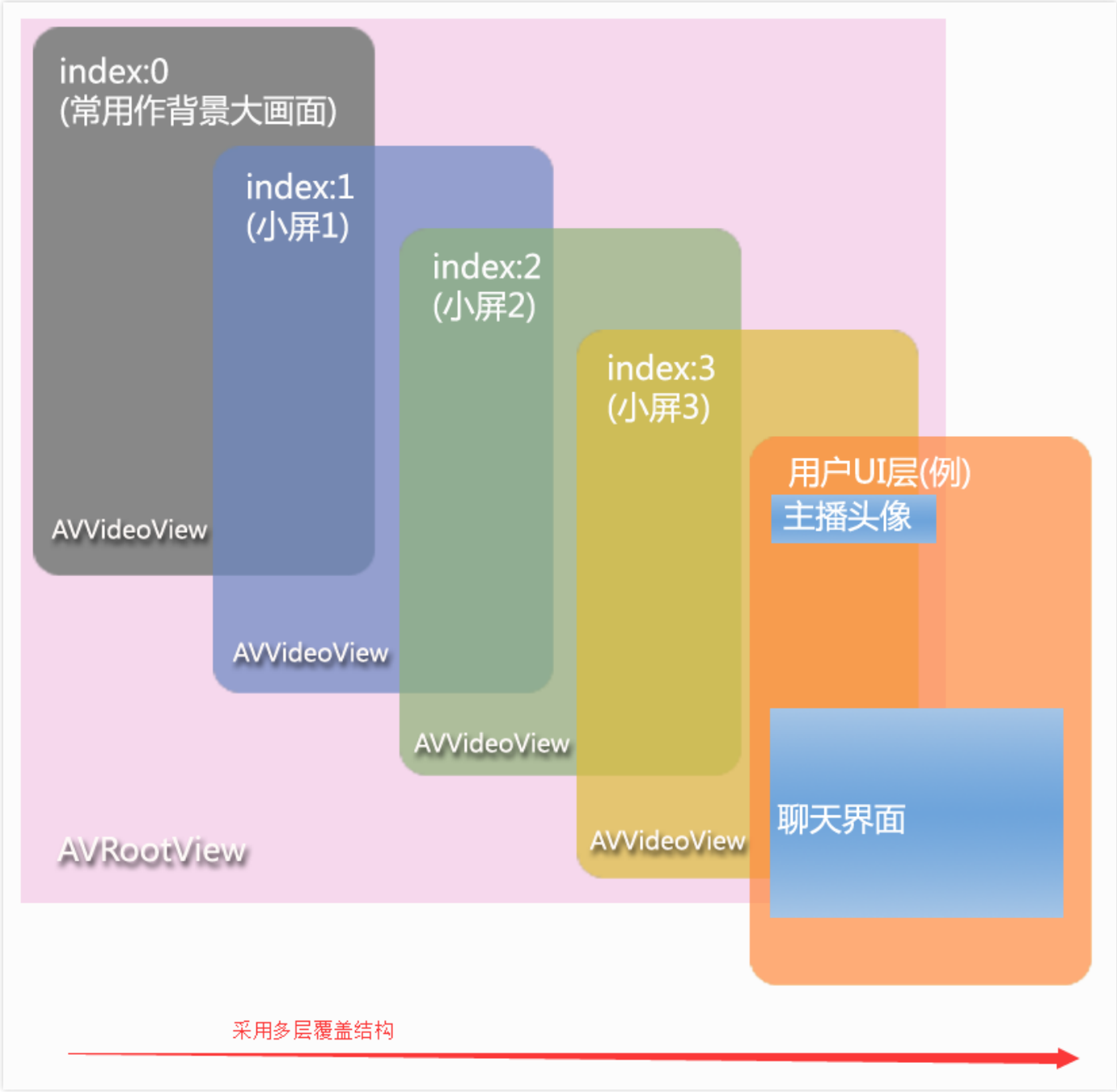
在此我们提供以下所讲到的完整 Demo 代码，如有需要请您自行下载。

Demo 代码下载

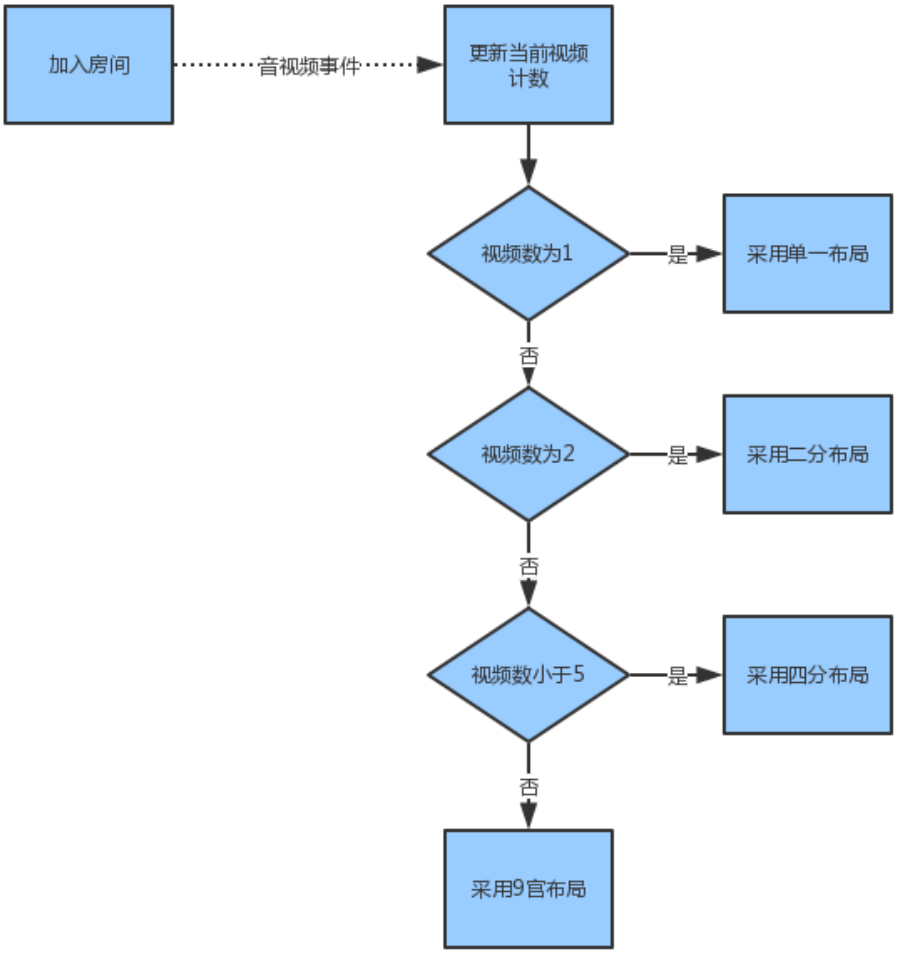
相关概念

- **AVRootView**
  AVRootView 继承自 Android 原生的 SurfaceView，可以理解为一个容器，用于放置所有的视频画面。

- **AVVideoView**
  逻辑上在的渲染控件，用于渲染一路视频画面，一个 **AVRootView** 可以包含多个 **AVVideoView**。



流程图



具体实现

可以通过**AVRootView**的宽高，动态设置**AVVideoView**的宽高。

二分布局

```
public void trViewDouble(){
avRootView.getViewByIndex(0).setPosLeft(0);
avRootView.getViewByIndex(0).setPosTop(0);
avRootView.getViewByIndex(0).setPosWidth(avRootView.getWidth());
avRootView.getViewByIndex(0).setPosHeight(avRootView.getHeight()/2);
avRootView.getViewByIndex(0).autoLayout();

avRootView.getViewByIndex(1).setPosLeft(0);
avRootView.getViewByIndex(1).setPosTop(avRootView.getHeight()/2);
avRootView.getViewByIndex(1).setPosWidth(avRootView.getWidth());
avRootView.getViewByIndex(1).setPosHeight(avRootView.getHeight()/2);
avRootView.getViewByIndex(1).autoLayout();
}
```

四分布局

```
public void trViewQuarter(){
// 计算视频画面的宽高
int subWidth = avRootView.getWidth()/2;
int subHeight = avRootView.getHeight()/2;

// 设置视频画面左上角位置
avRootView.getViewByIndex(0).setPosLeft(0);
avRootView.getViewByIndex(0).setPosTop(0);

avRootView.getViewByIndex(1).setPosLeft(subWidth);
avRootView.getViewByIndex(1).setPosTop(0);

avRootView.getViewByIndex(2).setPosLeft(0);
avRootView.getViewByIndex(2).setPosTop(subHeight);

avRootView.getViewByIndex(3).setPosLeft(subWidth);
avRootView.getViewByIndex(3).setPosTop(subHeight);

for (int i=0; i<4; i++){
avRootView.getViewByIndex(i).setPosWidth(subWidth);
avRootView.getViewByIndex(i).setPosHeight(subHeight);
avRootView.getViewByIndex(i).autoLayout();
}
}
```

API说明

setPosLeft

属于 **AVVideoView** 的方法，设置 **AVVideoView** 左上角x轴坐标，参数如下:

名称	类型	描述
posLeft	int	左上角x轴坐标，单位为像素

setPosTop

属于 **AVVideoView** 的方法，设置 **AVVideoView** 左上角y轴坐标，参数如下:

名称	类型	描述
posTop	int	左上角y轴坐标，单位为像素

setPosWidth

属于 **AVVideoView** 的方法，设置 **AVVideoView** 宽度，参数如下:

名称	类型	描述
posWidth	int	宽度，单位为像素

setPosHeight

属于 **AVVideoView** 的方法，设置 **AVVideoView** 高度，参数如下:

名称	类型	描述
posHeight	int	高度，单位为像素

autoLayout

属于 **AVVideoView** 的方法，刷新 **AVVideoView** 显示(使用设置的位置生效)

iOS

iOS 中视频渲染可以直接在视频中定制：
具体操作参考 *iOS 创建房间*。

PC

PC 中视频渲染可以直接在视频中定制：
具体操作参考 *PC 创建房间*。

常见问题

使用独立渲染如何定制布局?
独立渲染时，一个渲染控件 ILiveRootView 只显示一路视频画面，所以可以直接在布局文件布局 ILiveRootView 来实现视频布局。

联系邮箱

如果对上述文档有不明白的地方，请反馈到trtcfb@qq.com

屏幕分享
屏幕分享（android）

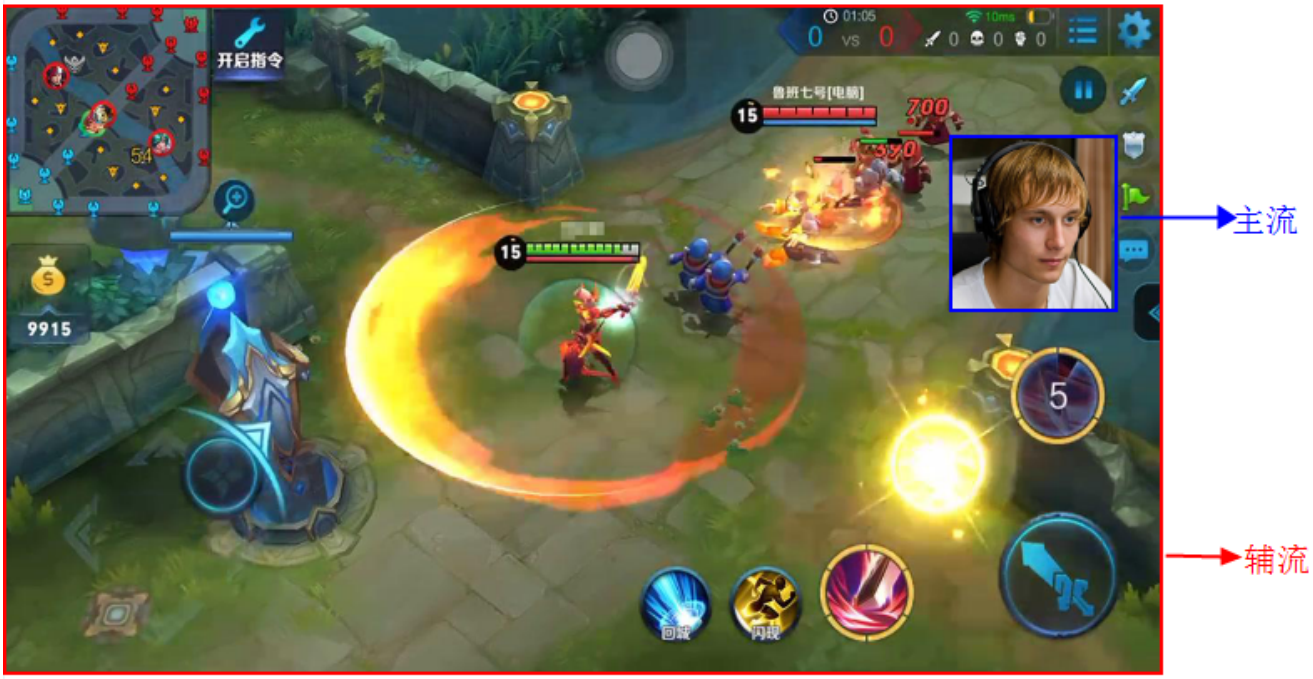
最近更新时间：2018-06-25 15:42:34

本文将指导您如何将主播的屏幕内容以视频的方式分享给观众观看

效果图

下图中，主播开启屏幕分享，观众端将看到主播的摄像头画面（主流）和屏幕画面（辅流，PS:主播在玩王者）。

主播



观众

源码下载

在此我们提供以下所讲到的完整 Demo 代码，如有需要请您自行下载。

Demo 代码下载

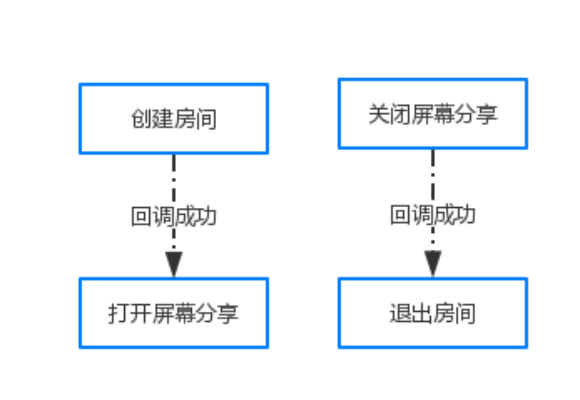
相关概念

视频数据类型

腾讯云支持一个帐号同时上行一路主流和一路辅流，这里用于区分视频流的来源称为视频类型，目前支持的视频类型有： 摄像头，屏幕分享，播片（PC端产生）。

常量	视频类型	流类型	描述
CommonConstants.Const_VideoType_Camera	摄像头	主流	通过摄像头采集数据产生
CommonConstants.Const_VideoType_Screen	屏幕分享	辅流	通过分享屏幕产生
CommonConstants.Const_VideoType_File	播片	辅流	通过播放视频文件产生

流程图



具体实现

开启直播

创建房间，开启直播可参考（创建房间）

开启屏幕录制，分享屏幕

用户在创建房间成功后，调用屏幕录制接口，开启屏幕分享:

```
/**
* @param mode 质量模式
* @param vertical 是否竖屏
* @param callBack 回调
*/
ILiveRoomManager.getInstance().enableScreen(int mode, boolean vertical, ILiveCallBack callBack)
```

参数说明：

int mode：屏幕录制时的分辨率。目前屏幕分享有三种质量模式，可参考 `com.tencent.ilivesdk.adapter.CommonConstants` 中常量：

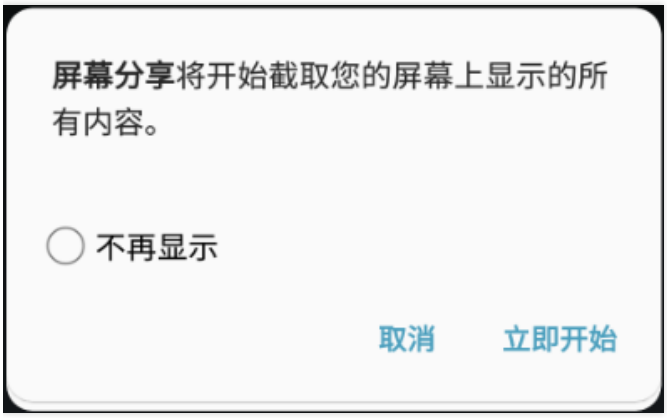
```
/** 超清（1280*720） */
public static int Const_Screen_Super_HD = AVVideoCtrl.SCREEN_SUPER_DEFINITION;
/** 高清(960*540) */
public static int Const_Screen_HD = AVVideoCtrl.SCREEN_HIGH_DEFINITION;
/** 标清(864*480) */
public static int Const_Screen_SD = AVVideoCtrl.SCREEN_STANDARD_DEFINITION;
```

boolean vertical：屏幕录制的旋转模式，true 为竖屏，false 为横屏

ILiveCallBack callBack：开启屏幕分享的结果回调，可处理成功或失败操作

录屏权限获取

开启屏幕录制时需动态获取 Android 系统的屏幕录取权限，如下图所示。单击【立即开始】即可（勾选"不再显示"可永久授权，后续开启屏幕分享将无需再申请权限）。



至此开启屏幕分享相关操作完成了，如果开启成功，此时主播用户的上行有两路流，一路摄像头采集的主流，另一路为屏幕分享的辅流。观众端看到的直播画面将如此前的效果图所示，有两个画面

> 注：上面相关代码及操作都为主播端的行为，观众端无需做任何处理。

停止屏幕分享

如主播用户退出房间或想主动关闭屏幕分享，调用停止屏幕分享接口即可

```
/**
 * @param callBack 回调
 */
ILiveRoomManager.getInstance().disableScreen(ILiveCallBack callBack);
```

常见问题

调用屏幕分享接口失败?

> 屏幕录制是 Android 5.0 之后提供的新接口，故需用户的 Android 系统高于 5.0，开发人员需在代码逻辑中对此做判断。

联系邮箱

如果对上述文档有不明白的地方，请反馈到trtcfb@qq.com

加入房间前预览
加入房间前预览（android）

最近更新时间：2018-06-25 15:43:17

本文将指导您的客户端在创建或加入房间前进行预览。

效果图

源码下载

在此我们提供以下所讲到的完整 Demo 代码，如有需要请您自行下载。
Demo 代码下载

相关概念

- **本地采集模式**
  指 SDK 定义的 Camera 的预览模式。我们知道，出于安全考虑，在设计 Android 的时候规定，如果需要调用 Camera，必须为其制显示在屏幕上的 SurfaceView 预览，否则将无法使用 Camera。设置参考 setCaptureMode

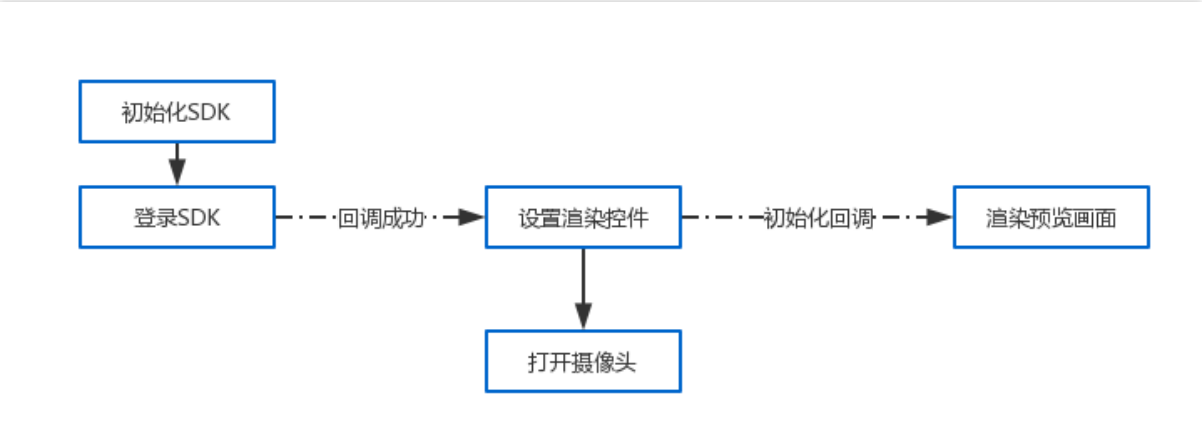
常量	模式名称	描述
ILiveConstants.CAPTURE_MODE_AUTO	自适应模式	使用自适应(Android 7.0 以上采用 SurfaceTexture 预览)
ILiveConstants.CAPTURE_MODE_SURFACETEXTURE	无悬浮窗模式	采用 SurfaceTexture 预览(无需悬浮窗权限)
ILiveConstants.CAPTURE_MODE_SURFACEVIEW	悬浮窗模式	采用 SurfaceView(需要悬浮窗权限)

示例:

```
// 设置无悬浮窗模式(登录前调用有效)
ILiveSDK.getInstance().setCaptureMode(ILiveConstants.CAPTURE_MODE_SURFACETEXTURE);
```

悬浮窗模式为官方方案，但鉴于添加悬浮窗权限申请在高版本系统中较为麻烦，推荐是使用自动模式。

流程图



具体实现

设置渲染控件

```
ILiveRoomManager.getInstance().initAvRootView(arvRoot);
```

打开摄像头

- 如果是使用 **无悬浮窗模式**，可以直接打开:

```
ILiveRoomManager.getInstance().enableCamera(ILiveConstants.FRONT_CAMERA, true);
```

- 如果是使用 **悬浮窗模式**，需确认悬浮窗已创建成功:

```
if (ILiveRoomManager.getInstance().isSurfaceViewCreated()) {
// 悬浮窗已创建，直接打开
ILiveRoomManager.getInstance().enableCamera(ILiveConstants.FRONT_CAMERA, true);
```

```
}else{
// 悬浮窗尚未创建成功，等悬浮窗创建成功后再打开
arvRoot.setSurfaceCreateListener(new AVRootView.onSurfaceCreatedListener() {
@Override
public void onSurfaceCreated() {
ILiveRoomManager.getInstance().enableCamera(ILiveConstants.FRONT_CAMERA, true);
}
});
}
```

渲染本地视频画面

由于渲染视频的 AVVideoView 并不是创建时直接创建的，需要等创建成功再开始渲染:

```
arvRoot.setSubCreatedListener(new AVRootView.onSubViewCreatedListener() {
@Override
public void onSubViewCreated() {
arvRoot.renderVideoView(true, ILiveLoginManager.getInstance().getMyUserId(), CommonConstants.Const_VideoType_Camera, true);
}
});
```

API说明

setCaptureMode

属于 ILiveSDK 的方法，用于设置 Camera 的预览模式，参数说明:

名称	类型	描述
CaptureMode	int	Camera的预览模式,参考预览模式

联系邮箱

如果对上述文档有不明白的地方，请反馈到trtcfb@qq.com

加入房间前预览（iOS）

最近更新时间：2018-06-25 15:43:30

本文讲解加入房间前，如何开启本地画面的预览。

效果图

源码下载

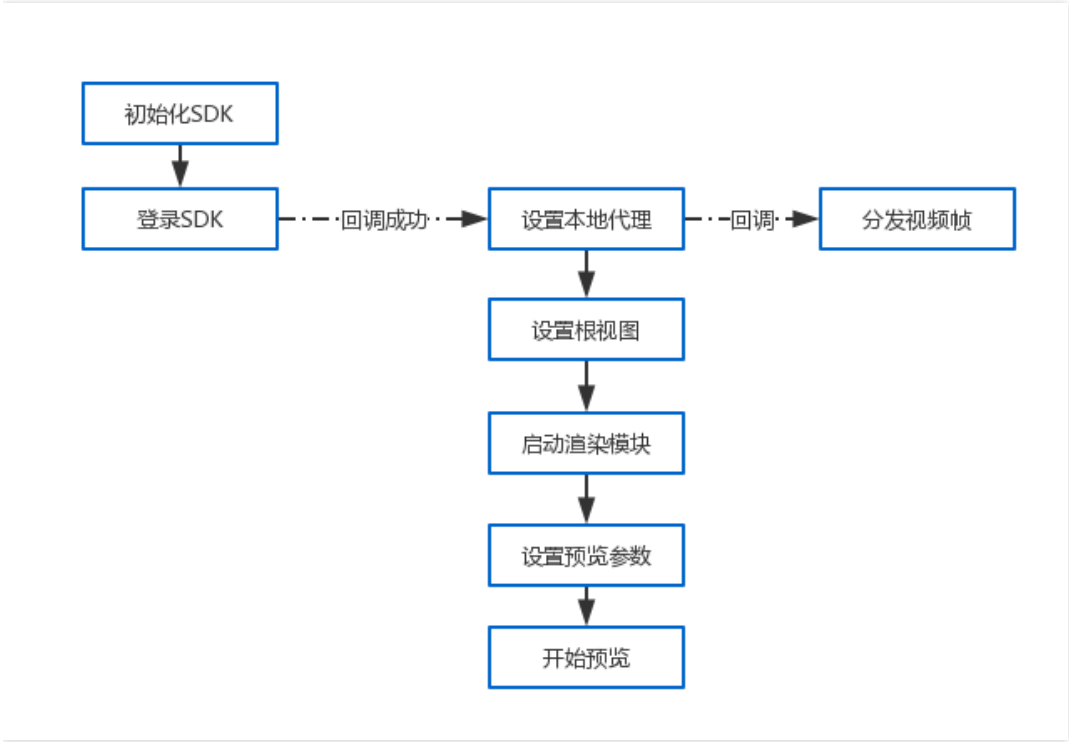
在此我们提供以下所讲到的完整 Demo 代码，如有需要请您自行下载。Demo 代码下载

相关概念

预览：

加入房间前试看本地画面，称为预览。普通的视频画面需要在加入房间后才能看到，如果想在加入房间前，先试看一下本地画面，则需要用到预览功能，预览画面无上行

流程图



具体实现

设置本地代理

1. 设置代理

```
QAVContext *context = [[ILiveSDK getInstance] getAVContext];
[context.videoCtrl setLocalVideoDelegate:self];
```

2. 代理回调中分发视频帧

```
- (void)OnLocalVideoPreview:(QAVVideoFrame *)frameData
{
    frameData.userId = [[ILiveLoginManager getInstance] getLoginId];
    ILiveFrameDispatcher *dispatch = [[ILiveRoomManager getInstance] getFrameDispatcher];
    [dispatch dispatchVideoFrame:frameData];
}
```

设置根视图

```
[[TILLiveManager getInstance] setAVRootView:self.view];
```

启动渲染模块

```
[[[ILiveRoomManager getInstance] getFrameDispatcher] startDisplay];
```

设置预览参数

```
ILiveCameraPreviewOption *option = [[ILiveCameraPreviewOption alloc] init];
option.width = 1280; //最低配置是192*114
option.height = 720;
option.fps = 20;
[[ILiveRoomManager getInstance] setCameraPreviewParam:option succ:^{
NSLog(@"set camera param succ");
} failed:^(NSString *module, int errId, NSString *errMsg) {
NSLog(@"set camera param fail");
}];
```

开始预览

```
[[ILiveRoomManager getInstance] enableCameraPreview:CameraPosFront enable:YES succ:^{
NSString *loginId = [[ILiveLoginManager getInstance] getLoginId];
[[TILLiveManager getInstance] addAVRenderView:[UIScreen mainScreen].bounds foruserId:loginId srcType:QAVVIDEO_SRC
_TYPE_CAMERA];
} failed:^(NSString *module, int errId, NSString *errMsg) {
NSLog(@"enable camera fail. m=%@,errid=%d,msg=%@",module,errId,errMsg);
}];
```

常见问题

- 看不到预览画面: 检查是否设置了根视图，检查是否在开启预览之前，已经设置了本地代理，并分发视频帧。
- 开启预览之后，再次进入房间，不需要再打开摄像头，但需要给视频赋予上行权限，接口是 `requestVideoAuth:succ:fail` 。

联系邮箱

如果对上述文档有不明白的地方，请反馈到trtcfb@qq.com

视频采集旋转

最近更新时间：2018-06-25 15:44:25

视频采集旋转操作会影响旁路直播、录制及观看
本文将指导您的客户端在渲染时定制视频画面旋转。

Android

效果图

自动旋转	手动旋转(不推荐)
	

源码下载

在此我们提供以下所讲到的完整 Demo 代码，如有需要请您自行下载。
Demo 代码下载

相关概念

重力感应

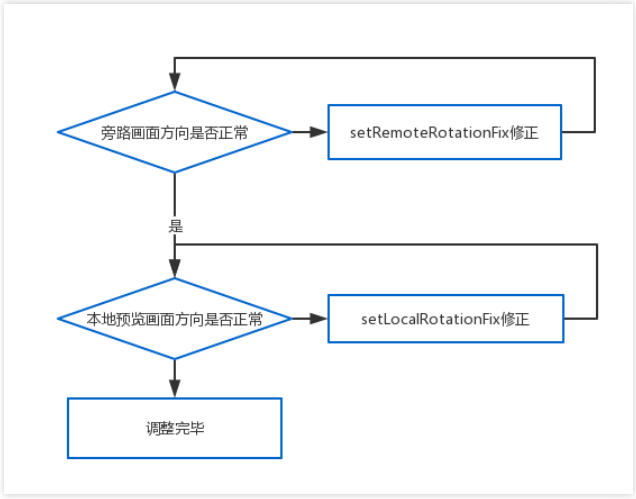
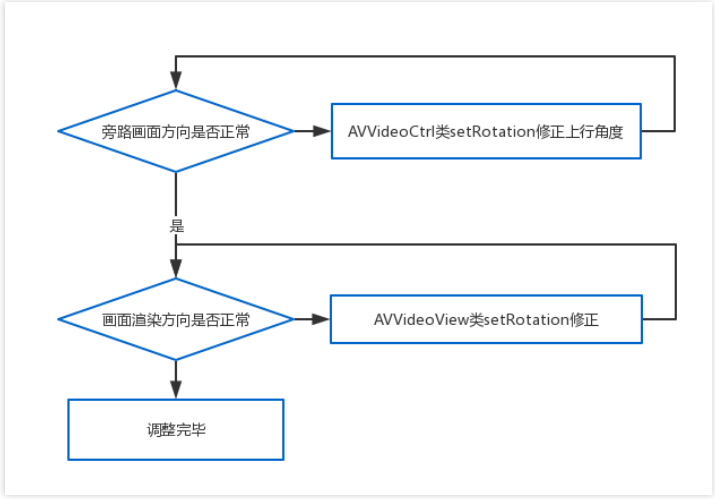
手机的重力感应技术是指手机受到重力影响时所对应出现的各种功能，这其中最重要的传感器是加速度感应器但又不限于加速度感应器，由

于摇动晃动手机所产生的变化感知也属于重力感应的一部分。
SDK 中通过重力感应来监测屏幕的方向来对视频画面进行旋转纠正。

自动旋转

iLiveSDK 默认是开启自动旋转功能的，即根据重力感应来视频画面进行旋转纠正，来保证不管手机旋转到哪个方向，看到的视频总是正的。

流程图

自动旋转	手动旋转
	

具体实现

固定或无重力感应设备初始角度

iLiveSDK 依赖重力感应来判断当前屏幕方向，所以对于一些固定或无重力感应的设备时，可能无法正常判断当前的屏幕方向。
这里可以通过渲染控件的 setDeviceRotation 来模拟重力感应设置设备角度:

```
arvRoot.setDeviceRotation(90);
```

设备摄像头方向校正

iLiveSDK 的自动旋转机制是对市场大多数手机做了兼容，但仍有可能在部分设备(尤其在非手机设备)上表现异常
此时用户可以通过 setLocalRotationFix 和 setRemoteRotationFix 来进行调整:

```
// 修正上行视频角度(不影响本地观看)，让其他用户看到自己的画面旋转 90 度
arvRoot.setRemoteRotationFix(90);
// 修正本地观看自己视频角度(不影响上行视频角度)，让自己看到自己的画面旋转 270 度
arvRoot.setLocalRotationFix(270);
```

手动旋转方向调整(不推荐）

用户可以通过 AVRootView 的 setAutoOrientation 方法配置为 false 关闭 sdk 的自动旋转机制。
关闭自动旋转意味着用户需要自己来设置数据上行角度，以及每一路视频的旋转角度。
设置方法如下:

```
// 设置本地视频数据的上行角度
((AVVideoCtrl)ILiveSDK.getInstance().getVideoEngine().getVideoObj()).setRotation(90);

// 设置一路视频的旋转角度
AVVideoView的setRotation来纠正渲染时的视频角度
```

需要注意前置摄像头与后置摄像头需要分开调整

API 说明

setDeviceRotation

属于渲染控件 AVRootView 的方法，用于模拟重力感应设置设备角度(自动旋转模式下会在设备旋转时覆盖)
参数：

名称	类型	描述
rotation	int	设备角度(0, 90, 180, 270)

setLocalRotationFix

属于渲染控件 AVRootView 的方法，用于纠正本地采集的摄像头数据在本地的旋转方向。
参数：

名称	类型	描述
rotation	int	设备角度(0, 90, 180, 270)

setRemoteRotationFix

属于渲染控件 AVRootView 的方法，用于纠正本地采集的摄像头数据上行的旋转方向(纠正对方看到自己的画面角度)。
参数：

名称	类型	描述
rotation	int	设备角度(0, 90, 180, 270)

setAutoOrientation

属于渲染控件 AVRootView 的方法，用于配置自动旋转的开关。
参数：

名称	类型	描述
enable	boolean	是否开启自动旋转机制

iOS

iOS 中不存在视频上行角度问题，直接默认配置即可

PC

PC 中不存在视频上行角度问题，直接默认配置即可

常见问题

联系邮箱

如果对上述文档有不明白的地方，请反馈到trtcfb@qq.com

观看视频旋转
观看视频旋转（android）

最近更新时间：2018-06-25 15:45:11

本课程讲解房间内画面旋转和拉伸相关问题

效果图

全屏模式	黑边模式
	

源码下载

在此我们提供以下所讲到的完整 Demo 代码，如有需要请您自行下载。Demo 代码下载

相关概念

画面一致

画面一致的场景:

> 视频画面的宽 < 视频画面的高
> 渲染视图的宽 < 渲染视图的高

以及:

> 视频画面的宽 > 视频画面的高
> 渲染视图的宽 > 渲染视图的高

画面不一致

画面不一致的场景:

> 视频画面的宽 < 视频画面的高
> 渲染视图的宽 > 渲染视图的高

以及:

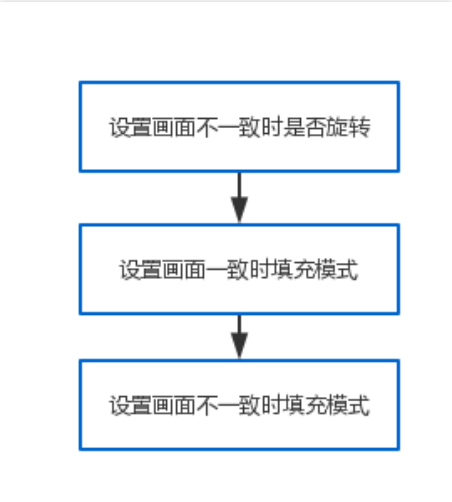
> 视频画面的宽 > 视频画面的高
> 渲染视图的宽 < 渲染视图的高

渲染模式

sdk提供全屏(放大)和黑边(缩小)两种渲染模式

模式名称	常量	描述
全屏模式	BaseVideoView.BaseRenderMode.SCALE_TO_FIT	通过放大视频画面并裁剪来解决尺寸不一致问题
黑边模式	BaseVideoView.BaseRenderMode.BLACK_TO_FILL	通过缩小视频画面并补充黑边来解决尺寸不一致问题

流程图



具体实现

尺寸不一致的渲染方式

我们知道

> 画面不一致时，视频画面与渲染视频的宽高比肯定是不一致的；
> 画面一致时，视频画面与渲染视频的宽高比仍有可能是不一致的；

所以在画面不拉伸变形的前提下，如果渲染画面就有了两种方案:

- 放大视频画面，然后做裁剪
  实现这里要有区分，画面一致时:

```
AVVideoView.setSameDirectionRenderMode(BaseVideoView.BaseRenderMode.SCALE_TO_FIT);
```

  画面不一致时:

```
AVVideoView.setDiffDirectionRenderMode(BaseVideoView.BaseRenderMode.SCALE_TO_FIT);
```

- 缩小画面，然后补黑边
  实现同样要区分，画面一致时：

```
AVVideoView.setSameDirectionRenderMode(BaseVideoView.BaseRenderMode.BLACK_TO_FILL);
```

  画面不一致时:

```
AVVideoView.setDiffDirectionRenderMode(BaseVideoView.BaseRenderMode.BLACK_TO_FILL);
```

画面不一致的渲染方式

在方向正常时仍有可能出现画面不一致(如移动端竖屏观看 PC 端的屏幕分享或外接摄像头)；
为最大程度展示画面，如果画面不一致时，我们可以配置是否自动旋转视频画面(默认是关闭)，来保证画面一致:

```
// 自动旋转视频画面，保证画面一致
AVVideoView.setRotate(true);
```

API 说明

setSameDirectionRenderMode

属于 AVVideoView 的方法，用于设置画面一致时的渲染方式。
参数：

名称	类型	描述

名称	类型	描述
mode	BaseVideoView.BaseRenderMode	渲染模式

setDiffDirectionRenderMode

属于 AVVideoView 的方法，用于设置画面一致时的渲染方式。
参数：

名称	类型	描述
mode	BaseVideoView.BaseRenderMode	渲染模式

setRotate

属于 AVVideoView 的方法，用于设置是否通过旋转视频画面来保证画面一致。
参数：

名称	类型	描述
enable	Boolean	开关(默认关闭)

常见问题

- 设置旋转无效，检查是否禁用了自动旋转(禁用后才生效)。
- 设置填充模式无效, 检查角度是否一致(角度一致时需使用sameDirectionRenderMode属性，否则使用diffDirectionRenderMode属性)。

联系邮箱

如果对上述文档有不明白的地方，请反馈到trtcfb@qq.com

观看视频旋转（iOS）

最近更新时间：2018-06-25 15:45:25

本课程讲解房间内画面旋转和拉伸相关问题

效果图

全屏模式	黑边模式
	

源码下载

在此我们提供以下所讲到的完整 Demo 代码，如有需要请您自行下载。 Demo 代码下载

相关概念

画面一致

画面一致的场景:

> 视频画面的宽 < 视频画面的高
> 渲染视图的宽 < 渲染视图的高

以及:

视频画面的宽 > 视频画面的高
渲染视图的宽 > 渲染视图的高

画面不一致

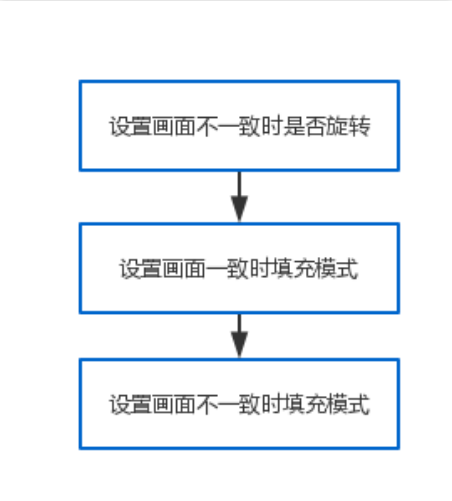
画面不一致的场景:

视频画面的宽 < 视频画面的高
渲染视图的宽 > 渲染视图的高

以及:

视频画面的宽 > 视频画面的高
渲染视图的宽 < 渲染视图的高

流程图



具体实现

创建房间

创建房间参考基础教程

禁用自动旋转

```
ILiveRenderView *view = [[TILLiveManager getInstance] addAVRenderView:[UIScreen mainScreen].bounds foruserId:user srcType:type];
view.autoRotate = NO;
```

旋转

```
NSArray *views = [[TILLiveManager getInstance] getAllAVRenderViews];
for (ILiveRenderView *view in views) {
view.rotateAngle = ILIVEROTATION_90;//枚举值，根据自己的需要旋转指定角度
}
```

角度一致时的填充

```
NSArray *views = [[TILLiveManager getInstance] getAllAVRenderViews];
for (ILiveRenderView *view in views) {
view.sameDirectionRenderMode = ILIVERENDERMODE_SCALEASPECTFILL;//枚举值，根据自己的需要设置填充模式
}
```

角度不一致时的填充

```
NSArray *views = [[TILLiveManager getInstance] getAllAVRenderViews];
for (ILiveRenderView *view in views) {
view.diffDirectionRenderMode = ILIVERENDERMODE_SCALEASPECTFILL;//枚举值，根据自己的需要设置填充模式
}
```

常见问题

- 设置旋转无效，检查是否禁用了自动旋转(禁用后才生效)
- 设置填充模式无效, 检查角度是否一致(角度一致时需使用sameDirectionRenderMode属性，否则使用diffDirectionRenderMode属性)

联系邮箱

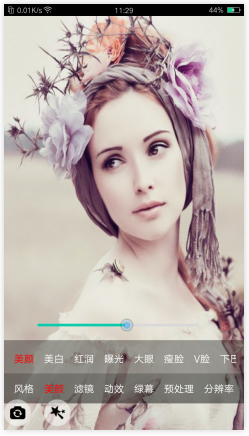
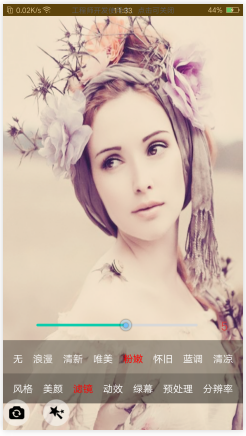
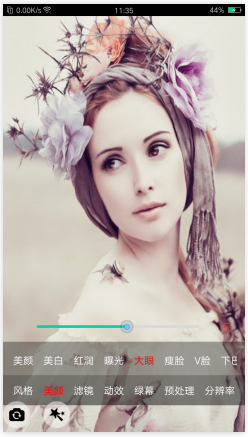
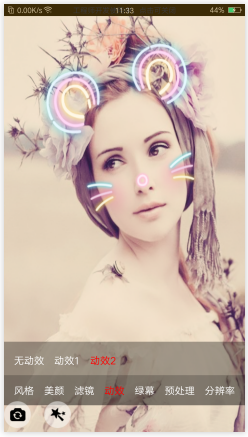
如果对上述文档有不明白的地方，请反馈到trtcfb@qq.com

视频特效（美颜挂件绿幕）
视频特效（android）

最近更新时间：2018-11-07 15:46:17

本文将指导您如何添加插件美颜 快速实现 美颜/滤镜/绿幕/动效/大眼/瘦脸等视频特效。

效果图

美颜	滤镜	大眼/瘦脸/小鼻等	动效	绿幕
				

源码下载

在此我们提供以下所讲到的完整 Demo 代码，如有需要请您自行下载。
Demo 代码下载

具体实现

1. 添加 licence(p 图收费版)

申请 p 图 licence；
将 licence 改名为 YTFaceSDK.licence 放入 app 的 assets 目录下

2. 集成美颜插件

p 图收费版：

```
// build.gradle 中，加载p图收费版
compile 'com.tencent.ilivefilter:liteav_pitu:1.1.22'
```

```
defaultConfig{
....
// 指定ndk架构
ndk {
// p图收费版；只支持 armeabi 架构
abiFilters 'armeabi'
}
}
```

非 p 图普通版：

```
// build.gradle 中，加载非p图普通版
compile 'com.tencent.ilivefilter:liteav_normal:1.1.22'
```

```
defaultConfig{
....
// 指定ndk架构
ndk {
// 非p图版，支持'armeabi', 'armeabi-v7a' 两种架构
abiFilters 'armeabi', 'armeabi-v7a'
}
}
```

3. 初始化美颜插件

```
boolean bGLContext = false; // iLiveSDK 场景下，设置当前为无 OpenGL 环境
TXCVideoPreprocessor mTxcFilter = new TXCVideoPreprocessor(this, bGLContext);
```

4. 处理视频数据

进入房间成功后，设置 iLiveSDK 数据回调，美颜插件处理数据

(**设置数据回调，一定要在进入房间成功后才有效。**)

```
// 设置数据回调
boolean bRet = ILiveSDK.getInstance().getAvVideoCtrl().setAfterPreviewListener(new AVVideoCtrl.AfterPreviewListener(){
@Override
public void onFrameReceive(AVVideoCtrl.VideoFrame var1) {

// 回调的数据，传递给 美颜插件 processFrame 接口处理;并且通过 var1.data 返回处理后的数据
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1) {
mTxcFilter.processFrame(var1.data, var1.width, var1.height, var1.rotate, var1.videoFormat, var1.videoFormat);
}
}
});
```

例外：

p 图版本，并且 iLiveSDK 版本 小于 1.7.0；只能使用 setLocalVideoPreProcessCallback 回调接口，否则 p 图功能无效。

iLiveSDK 版本，可通过 ILiveSDK.getInstance().getVersion() 获取

详情参考 开发说明

5. 设置效果参数

根据 API 说明，设置相应的效果， 如：

```
mTxcFilter.setBeautyLevel(5); // 设置美颜级别,范围 0～10
mTxcFilter.setWhitenessLevel(3); // 设置美白级别,范围 0～10
mTxcFilter.setRuddyLevel(2); // 设置红润级别，范围 0～10
```

6. 释放资源

退出房间时，必须释放插件美颜资源，否则下次进入房间，设置特效不生效。
日志也会出现“please realloc new TXCVideoPreprocessor”错误

```
ILVLiveManager.getInstance().quitRoom(new ILiveCallBack() {
@Override
public void onSuccess(Object data) {
// 取消 iLiveSDK 相机数据回调（参数传null）
boolean bRet = ILiveSDK.getInstance().getAvVideoCtrl().setAfterPreviewListener(null);
// 退出房间后，一定要销毁filter 资源；否则下次进入房间，setFilter将不生效或其他异常
mTxcFilter.release();
mTxcFilter = null;
}
@Override
public void onError(String module, int errCode, String errMsg) {
}
});
```

7. 扩展：其他使用场景

如：

1. 自定义采集
2. 需要支持纹理输入-->纹理输出场景
3. 自定义滤镜

请参考详细文档 开发说明

API说明

接口名	接口描述	参数定义	返回值
TXCVideoPreprocessor(Context activity, boolean bGLContext)	构造函数	**activity**：当前Activity的this指针 **bGLContext**：当前是否有Opengl 环境	无
逻辑代码接口			
void setListener(TXIVideoPreprocessorListener listener)	设置sdk数据监听回调	**listener**: sdk数据返回监听（用于接收process* 处理函数返回的数据）	无
void setNotifyListener(TXINotifyListener notify)	设置sdk事件监听回调	**notify**: sdk事件监听；事件为TXCVideoPreprocessor.EventVideoProcess.*	无

接口名	接口描述	参数定义	返回值
int processFrame(int texture, int width, int height, int processAngle, int inputFormat, int outputFormat)	纹理输入处理	**texture**：输入纹理 **width**：纹理宽 **height**：纹理高 **processAngle**：纹理旋转角度 **inputFormat**：视频输入格式（0：2D纹理输入 1：I420数据输入 2：RGBA 数据输出 3：NV21数据输入 4:外部纹理输入（如相机纹理，需要调用 setInputMatrix 设置旋转矩阵）） **outputFormat**：视频输出格式（0：2D纹理输出 1：I420数据输出 2：RGBA 数据输出 3：NV21数据输出）	-1：失败 >=0：成功
int processFrame(byte[] data, int width, int height, int processAngle, int inputFormat, int outputFormat)	原始数据输入处理	**data**：输入原始数据 **width**：数据宽 **height**：数据高 **processAngle**：纹理旋转角度 **inputFormat**：视频输入格式（0：2D纹理输入 1：I420数据输入 2：RGBA 数据输出 3：NV21数据输入 4:外部纹理输入（如相机纹理，需要调用 setInputMatrix 设置旋转矩阵）） **outputFormat**：视频输出格式（0：2D纹理输出 1：I420数据输出 2：RGBA 数据输出 3：NV21数据输出）	-1：失败 >=0：成功
void setInputMatrix(float[] mtx)	设置外部纹理输入时的旋转矩阵（仅在外部纹理输入时，才使用）	**mtx**: 外部纹理输入时，从 SurfaceTexture.getTransformMatrix() 中获取	无
void release()	释放sdk资源	无	无
美颜相关接口			
void setBeautyStyle(int style)	设置美颜风格	**style**: 美颜风格 0: 光滑 1: 自然 2: 朦胧	无
void setBeautyLevel(int level)	设置美颜级别	**level**: 美颜级别（0 - 9）	无
void setWhitenessLevel(int level)	设置美白级别	**level**: 美白级别（0 - 9）	无
void setRuddyLevel(int level)	设置红润级别	**level**: 红润级别（0 - 9）	无
滤镜相关接口			
int setFilterType(int type)	切换滤镜	**type**: 滤镜编号 1:无 2：浪漫 3：清新 4：唯美 5：粉嫩 6：怀旧 7: 蓝调 8:清凉 9: 日系	-1：失败 0：成功
void setFilterImage(Bitmap bmp)	设置滤镜图片	**bmp**: 滤镜图片	无
void setFilterImage(String imagePath)	设置滤镜图片路径	**imagePath**：滤镜文件路径	无
void setFilterMixLevel(final float specialValue)	设置滤镜程度	**specialValue**：滤镜程度（0.0 - 1.0）	无
视频编辑相关接口			

接口名	接口描述	参数定义	返回值
void setCrop(CropRect cutRect)	设置剪裁大小	**cutRect**：剪裁矩阵 cutRect.x：x轴坐标偏移 cutRect.y：y轴坐标偏移 cutRect.cropWidth：x轴剪裁长度 cutRect.cropHeight：y轴剪裁高度	无
void setRotate(int angle)	设置输出旋转顺时针角度	**angle**：输出顺时针旋转角度	无
void setOutputFrameSize(int width, int height)	设置输出长宽	**width**：输出数据宽 **height**：输出数据高	无
void setMirror(boolean enable)	设置输出图像左右镜像	**enable**：true：开启左右镜像 false：不开启左右镜像	无
void setWaterMark(Bitmap bitmap, float x, float y, float width)	设置水印（位置以左上角为原点）	**bitmap**: 水印图片BitMap **x**：(0.0 - 1.0)归一化坐标，左上角x轴偏移 **y**：(0.0 - 1.0)归一化坐标，左上角y轴偏移 width：(0.0 - 1.0)归一化宽度；左上角x轴宽度	无
void setWaterMarkList(final List< WaterMakeTag> markList)	设置多个水印（setWaterMark的加强版）	**markList**：多个水印 WaterMakeTag 链表	无
p图收费版相关接口			
void setFaceSlimLevel(int level)	设置瘦脸级别	**level**: 瘦脸级别（0 - 9）	无
void setEyeScaleLevel(int level)	设置大眼级别	**level**: 大眼级别（0 - 9）	无
void setFaceVLevel(int level)	设置V脸级别	**level**: V脸级别（0 - 9）	无
void setFaceShortLevel(int level)	设置短脸级别	**level**: 短脸级别（0 - 9）	无
void setChinLevel(int level)	设置长下巴级别	**level**: 长下巴级别（0 - 9）	无
void setNoseSlimLevel(int level)**	设置小鼻级别	**level**: 小鼻级别（0 - 9）	无
void setMotionTmpl(String tmplPath)	设置动态贴纸路径	**tmplPath**: 动态贴纸路径	无
boolean setGreenScreenFile(String path, boolean isLoop)	设置绿幕文件路径	**path**: 绿幕文件路径(目前图片支持jpg/png/bmp，视频仅支持mp4格式) **isLoop**：是否循环播放设置的视频文件（只针对视频）	无
boolean setGreenScreenParam(TXCGreenScreenParam param)	设置绿幕参数	**param**.fillMode：绿幕背景填充参数 **param**.xMirror：x轴是否镜像；因为Android相机前置摄像头为左右镜像；所以如果想要主播看到绿幕背景为正，则需要左右镜像绿幕	无

常见问题

为什么 p 图功能不生效？

1. 确认 licence 文件名称是否是 YTFaceSDK.licence；
2. 确认 licence 文件 是否在 assets 目录下；
3. processFrame 传入的processAngle角度，是否正确；iLiveSDK 一般传 var1.rotate；

为什么 licence 存在，但是加载会失败？

确认 licence 是否过期；可以找申请 licence 时的相关人员确认。

p 图版和非 p 图版有什么区别？ 非 p 图版是收费的吗？

1. p 图版，集成了 p 图人脸识别 sdk；支持 美颜/滤镜/水印等基础功能 + 绿幕/动效/大眼/瘦脸等人脸识别相关的 p 图特效，
2. 非 p 图版本，只支持美颜/滤镜/水印等基础功能，
3. 非 p 图版，是免费使用的，

为什么 p 图版本，so 库只支持 armeabi 架构？

1. 市面上大多数手机，都是 armeabi-v7a/arm64-v8a/x86 等架构；而这些架构， 都可以兼容 armeabi；
2. 为了 p 图 sdk 的体积考虑，所以只提供 armeabi 架构的 so 库

为何会分“有 GL 环境”和“无 GL 环境”两种模式？

1. 一般应用场景，只会出现“有 GL 环境”和“无 GL 环境”；从效率和使用方式上考虑，故 sdk 支持两种模式；
2. 如果用户使用场景（如：自定义采集），存在GL环境；为了不与现有 GL 环境冲突，使用“有 GL 环境”模式，支持纹理输入-->处理后纹理输出，效率和兼容性更好；
3. 如果用户使用场景（如：使用 iLiveSDK 采集），无 GL 环境；sdk 会单独开启一个线程，处理传入的原始视频数据，此时不支持纹理输入-->纹理输出，仅支持 原始数据输入-->原始数据输出；

接入 p 图版本，如何收费？

参见上面的“费用说明”

为什么通过 `compile 'com.tencent.ilivefilter:****'` ,提示 “Could not get resource '**.aar'” ，提示 aar 无法找到呢？**

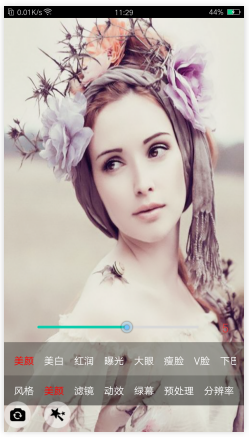
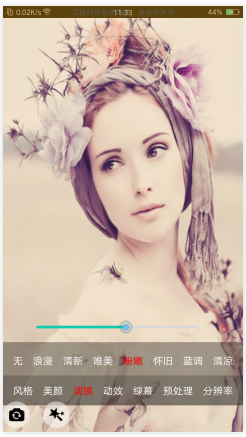
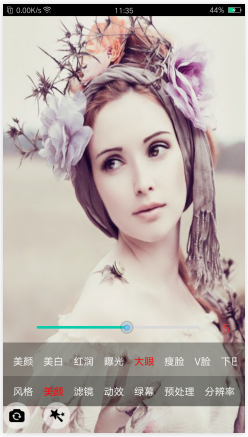
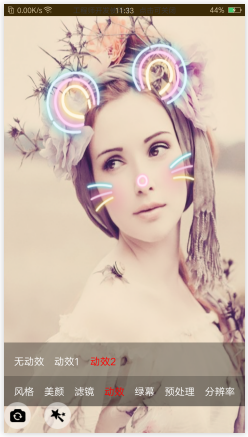
可能是 jcenter 服务器下载链接的问题；可以改成 `compile 'com.tencent.ilivefilter:****@aar'` 试试

视频特效（iOS）

最近更新时间：2018-09-14 16:02:59

本文将指导您在 ILiveSDK 中集成预处理插件（TXMVideoPreprocessor）实现视频特效功能。

效果图

美颜	滤镜	大眼/瘦脸/小鼻等	动效	绿幕
				

源码下载

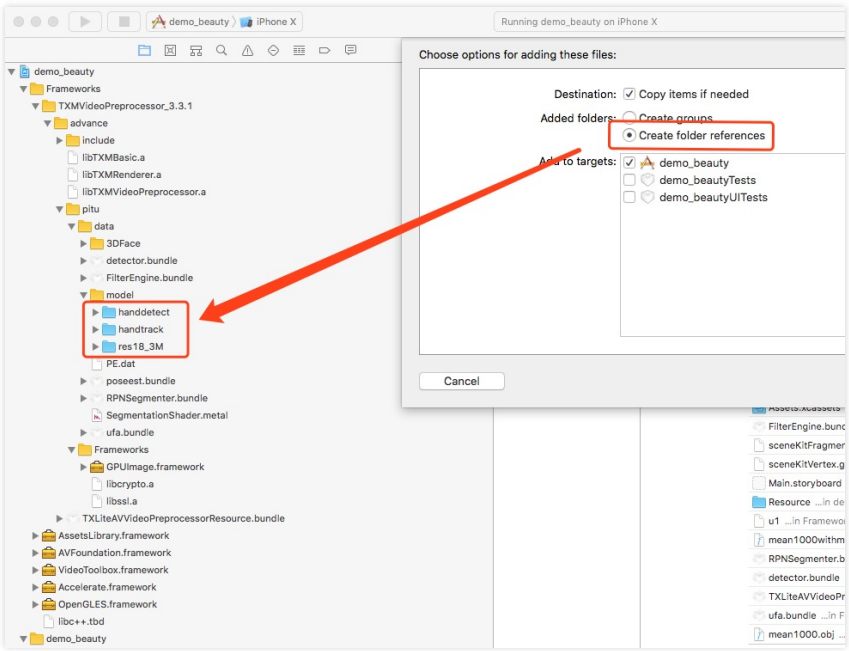
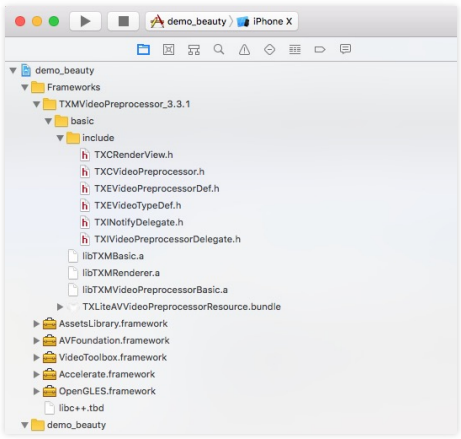
在此我们提供以下所讲到的完整 Demo 代码，如有需要请您自行下载。

Demo 代码下载

集成说明

引入依赖库

高级版本	基础版本

高级版本	基础版本
	
确保所有的资源都引入到Copy Bundle Resources，并且model文件夹下的资源以Create folder references方式引入	

- 温馨提示：

> (1) 如果不使用滤镜，可以删除 TXLiteAVVideoPreprocessorResource.bundle；
> (2) 高级版本需要申请 licence 并同步到 Copy Bundle Resources 中，licence 不能重命名，必须为 YTFaceSDK.licence；
> (3) 下载的 TXMVideoPreprocessor.advance 中的 Pitu 子文件夹及 YoutuBeauty 子文件夹中的所有 bundle，并同步到 Copy Bundle Resources 中，否则直接 crash；
> (4) 如果您是 AVSDK 的用户，processFrame 的 orientation 参数应该设置为 frameData.frameDesc.rotate，否则无法识别到人脸；
> (5) 使用滤镜功能时，确保设置融合度 setFilterMixLevel，否则默认为 0；
> (6) 如需要更多动效资源，请联系商务，并将资源添加到 Resource 文件夹下；

工程配置

> 基础版本无须做工程配置，用默认配置即可。

集成到ILiveSDK

1.对象创建

```
//引入头文件
#import "TXCVideoPreprocessor.h"
//声明变量
@property (nonatomic, strong) TXCVideoPreprocessor *preProcessor;
@property (nonatomic, assign) Byte *processorBytes;
//创建变量
self.preProcessor = [[TXCVideoPreprocessor alloc] init];
[self.preProcessor setDelegate:self];//TXIVideoPreprocessorDelegate
```

2.设置 ILiveSDK 数据回调

```
// 进房前设置数据帧回调
[[ILiveRoomManager getInstance] setLocalVideoDelegate:self];//QAVLocalVideoDelegate
```

3.调用处理接口

```
- (void)OnLocalVideoPreProcess:(QAVVideoFrame *)frameData
{
//设置美颜、美白、红润等参数
[self.preProcessor setBeautyLevel:5];
[self.preProcessor setRuddinessLevel:8];
[self.preProcessor setWhitenessLevel:8];
[self.preProcessor setOutputSize:CGSizeMake(frameData.frameDesc.width, frameData.frameDesc.height)];
//开始预处理
[self.preProcessor processFrame:frameData.data width:frameData.frameDesc.width height:frameData.frameDesc.height ori
entation:TXE_ROTATION_0 inputFormat:TXE_FRAME_FORMAT_NV12 outputFormat:TXE_FRAME_FORMAT_NV12];
//将处理完的数据拷贝到原来的地址空间，如果是同步处理，此时会先执行（4）
if(self.processorBytes){
memcpy(frameData.data, self.processorBytes, frameData.frameDesc.width * frameData.frameDesc.height * 3 / 2);
}
}
```

4.回调中保存数据

```
- (void)didProcessFrame:(Byte *)bytes width:(NSInteger)width height:(NSInteger)height format:(TXEFrameFormat)format ti
meStamp:(UInt64)timeStamp
{
self.processorBytes = bytes;
}
```

API 说明

预处理接口调用

```
/*
* 预处理数据
* @param sampleBuffer 帧数据（420f和y420）
* @param orientation 人脸识别方向（无需人脸识别：TXE_ROTATION_0）
* @param outputFormat 输出数据格式
* @return int 是否调用成功，0成功，-1失败
*/
```

```
- (int)processFrame:(CMSampleBufferRef)sampleBuffer orientation:(TXERotation)orientation outputFormat:(TXEFrameFormat)outputFormat;

/*
* 预处理数据
* @param data 帧数据
* @param width 宽
* @param height 高
* @param orientation 人脸识别方向（无需人脸识别：TXE_ROTATION_0）
* @param inputFormat 输入数据格式
* @param outputFormat 输出数据格式
* @return int 是否调用成功，0成功，-1失败
*/
- (int)processFrame:(Byte *)data width:(NSInteger)width height:(NSInteger)height orientation:(TXERotation)orientation inputFormat:(TXEFrameFormat)inputFormat outputFormat:(TXEFrameFormat)outputFormat;
```

数据处理完回调

```
/*
* 设置代理
* @param delegate 代理
*/
- (void)setDelegate:(id<TXIVideoPreprocessorDelegate>)delegate;

/*
* 设置通知（是否识别到人脸等通知）
* @param notify 通知
*/
- (void)setNotify:(id<TXINotifyDelegate>)notify;

@protocol TXIVideoPreprocessorDelegate <NSObject>
@optional
/*
* 添加水印前回调
* @param texture 纹理id
* @param width 宽
* @param height 高
* @return 处理后纹理id
*/
- (GLuint)willAddWatermark:(GLuint)texture width:(NSInteger)width height:(NSInteger)height;

/**
* 在OpenGL线程中回调，可以在这里释放创建的OpenGL资源
*/
- (void)onTextureDestoryed;

/*
* 数据处理回调
* @param sampleBuffer 帧数据
* @param timeStamp 时间戳
*/
- (void)didProcessFrame:(CMSampleBufferRef)sampleBuffer timeStamp:(UInt64)timeStamp;

/*
```

```
* 数据处理回调
* @param bytes 帧数据
* @param width 宽
* @param height 高
* @param format 输出数据格式
* @param timeStamp 时间戳
*/
- (void)didProcessFrame:(Byte *)bytes width:(NSInteger)width height:(NSInteger)height format:(TXEFrameFormat)format timeStamp:(UInt64)timeStamp;

/**
* 人脸数据回调（启用了pitu模块才有效，开启pitu模块必须是打开动效或大眼瘦脸）
* @prama points 人脸坐标
*/
- (void)onDetectFacePoints:(NSArray *)points;
@end
```

基础功能

美颜风格、美颜、美白、红润、滤镜、水印、裁剪、旋转、缩放、镜像

```
/*
* 设置输入裁剪区域
*/
@property (nonatomic, assign) CGRect cropRect;

/*
* 设置裁剪后旋转角度
*/
@property (nonatomic, assign) TXERotation rotateAngle;

/*
* 设置旋转后缩放大小
*/
@property (nonatomic, assign) CGSize outputSize;

/*
* 设置缩放后是否镜像
*/
@property (nonatomic, assign) BOOL mirror;

/*
* 设置水印是否镜像
*/
@property (nonatomic, assign) BOOL waterMirror;

/*
* 设置绿幕参数
*/
@property (nonatomic, strong) TXCGreenScreenParam *greenParam;
```

```
/*
 * 是否异步回调（默认NO）
 */
@property (nonatomic, assign) BOOL isAsync;

/*
 * 设置美颜风格
 * @param level 设置美颜类型，默认TXE_BEAUTY_TYPE_SMOOTH
 */
- (void)setBeautyStyle:(TXEBeautyStyle)style;

/*
 * 设置美颜（0-10）
 * @param level 美颜程度，0表示原图
 */
- (void)setBeautyLevel:(NSInteger)level;

/*
 * 设置美白（0-10）
 * @param level 美白程度，0表示原图
 */
- (void)setWhitenessLevel:(NSInteger)level;

/*
 * 设置红润（0-10）
 * @param level 红润程度，0表示原图
 */
- (void)setRuddinessLevel:(NSInteger)level;

/*
 * 设置水印
 * @param view 水印view
 */
- (void)setWaterMark:(UIView *)view;
```

可以使用sdk自带的滤镜资源，也可以自定义滤镜资源，通过设置融合度调整滤镜资源和图像的融合程度。

```
/*
* 设置滤镜
* @param type 滤镜类型
*/
- (void)setFilterType:(TXEFilterType)type;

/*
* 设置滤镜
* @param imagePath 滤镜资源路径
*/
- (void)setFilterImage:(NSString *)imagePath;

/*
* 设置滤镜
* @param image 滤镜图片
*/
- (void)setFilterUIImage:(UIImage *)image;
```

```
/*
* 设置滤镜融合度（0-1）
* @param level 滤镜融合度
*/
- (void)setFilterMixLevel:(CGFloat)level;
```

高级功能

大眼、瘦脸、v脸、短脸、下巴、瘦鼻、绿幕、动效。

```
/*
* 设置绿幕
* @param file 绿幕文件路径
*/
- (void)setGreenScreenFile:(NSString *)file;

/*
* 设置大眼（0-10）
* @param level 大眼程度
*/
- (void)setEyeScaleLevel:(NSInteger)level;

/*
* 设置瘦脸（0-10）
* @param level 瘦脸程度
*/
- (void)setFaceSlimLevel:(NSInteger)level;

/*
* 设置美型（0-10）
* @param level 美型程度
*/
- (void)setFaceBeautyLevel:(NSInteger)level;

/*
* 设置V脸（0-10）
* @param level V脸程度
*/
- (void)setFaceVLevel:(NSInteger)level;

/*
* 设置下巴拉伸或收缩（-10-10，0为正常）
* @param level 伸缩程度
*/
- (void)setChinLevel:(NSInteger)level;

/*
* 设置短脸（0-10）
* @param level 短脸程度
*/
```

```
- (void)setFaceShortLevel:(NSInteger)level;

/*
* 设置瘦鼻（0-10）
* @param level 瘦鼻程度
*/
- (void)setNoseSlimLevel:(NSInteger)level;

/*
* 设置动效
* @param templatePath 动效资源路径
*/
- (void)setMotionTemplate:(NSString *)templatePath;
```

将动效资源解压在 Resource 目录下，通过资源路径设置动效。

```
/*
* 设置动效
* @param templatePath 动效资源路径
*/
- (void)setMotionTemplate:(NSString *)templatePath;
```

使用绿幕需要先准备一个用于播放的 mp4 文件，通过调用以下接口即可开启绿幕效果。

```
/*
* 设置绿幕
* @param file 绿幕文件路径
*/
- (void)setGreenScreenFile:(NSString *)file;
```

常见问题

功能问题

sdK提供美颜、美白、红润、滤镜、大眼、瘦脸、动效贴纸、绿幕等功能，其中大眼、瘦脸、动效贴纸等以人脸识别为基础的功能是基于优图实验室的人脸识别技术和天天P图的美妆技术为基础开发的特权功能，通过跟优图和P图团队合作，将这些特效深度整合到图像处理流程中，以实现更好的视频特效。

费用问题

美颜、美白、红润等基础功能是免费的。基于人脸识别的功能由于采用了优图实验室的专利技术，授权费用约 50W/年（目前国内同类图像处理产品授权均在百万左右）。如有需要可以提工单或联系我们（jerryqian QQ:364993028 ），商务同学会提供 P 图 SDK，并替您向优图实验室申请试用 License。

联系邮箱

如果对上述文档有不明白的地方，请反馈到trtcfb@qq.com