蓝松短视频 SDK 开发手册

目录

一、蓝松 SDK 组成

1、 蓝松 SDK 包含了**视频编辑 SDK** 和 **AE 模版 SDK**
2、 此文档基于蓝松 SDK4.3.4 版本编辑
3、文档更新说明:

序号	文档更新内容	时间
1	更新 Android、IOS 的 API 说明	20210114
2	更新人像分割 SDK 库说明	20210519

二、SDK 集成和单位说明

2.1、Android 端

1. 我们的 SDK 以 module 的形式提供， 在你的项目中点击 File/import midule 导入,并在您主 module 中的 build.gradle 中增加 implementation project(':LanSongSDK')(如果是更新 SDK, 则先删除当前的 module,,并屏蔽 build.gradle 中的字段，然后在 import 导入,再打开屏蔽)
2. LanSongSDK module 下的 assets 下有 LanSongIFxxx 开头滤镜用到的各种图片资源，如您项目没有用到 LanSongIFxxx 开头的滤镜，则可以删除。
3. **其他的代码，各种资源，各种视频素材等均为演示所用，不属于 sdk 的一部分。**
4. 代码使用

```
1. LanSoEditor.initSDK(getApplicationContext(), null); // 初始化 SDK.[必须] (null 时,APP 名字一定是我们 demo 名字)

2. LanSoEditor.setTempFileDir(); //设置 SDK 处理过程中的临时文件缓存路径,也是我们容器最终生成视频的路径，您可以设置生成视频名字的前后缀．此代码是公开的，您点击后会看到更多注释 [可选]

3. LanSoEditor.setSDKLogOutListener //设置日志输出监听 [可选].
```

2.2、IOS 端

1. 拖动 LanSongEditorFramework.framework 到你的项目中，并在工程的 Build Phases 标签页的 Link Binary With Libraries 中把我们的 framework 放到最上面.
2. 在 Build Phases 的<Link Binary With Libraries>中增加 libz.tbd ，libbz2.tbd, libc++.tbd, libiconv.tbd
3. 在你代码中包括头文件： #import <LanSongEditorFramework/ LanSongEditor.h>

4. LanSongEditorBundle.bundle 是我们 sdk 中 以 LanSongIFxxx 开头用到的一些滤镜图片,如果没用到 LanSongIFxxx 开头的滤镜，则不用增加此文件。如用到则把 LanSongEditorBundle.bundle 拖动到你项目的 copy Bundle Resources 中
5. **其他的代码，各种资源，各种视频素材等均为演示所用，不属于 sdk 的一部分。**
6. 代码使用

初始化

1. 增加头文件

```
#import <LanSongEditorFramework/LanSongEditor.h>
```

2. 在您程序刚开始的地方初始化我们的 SDK：

```
[LanSongEditor initSDK:nil]//初始化 SDK [必须] (nil 时,APP 名字一定是我们 demo 名字)
```

2.3、时间单位

android

- 用到两种时间单位：float 类型和 long 类型；
- float 类型是秒，long 类型是微秒，
- 1 秒等于 1000*1000 微秒;所有时间参数是 float 类型,即为秒;所有时间参数是 long 类型，则是微秒；
- 我们的一个容器类在执行的时候,有进度回调 OnLanSongSDKProgressListener 和 OnLanSongSDKThreadProgressListener, 其中 ThreaProgress 是工作在内部线程中,直接调用监听给你, 没有经过 handle+message 机制, 监听的两个参数分别是(long currentPtsUs, int percnet); 其中 currentPtsUs:是当前即将处理的帧的时间戳, percent 是当前处理的百分比;

IOS

- ios 用到一种单位：CGFloat 类型， 浮点类型，单位秒；
- 视频在编码中, 我们默认是一秒钟一个 IDR 帧;

三、视频编辑 SDK

3.1、SDK 中用到的名字解释

1. composition: 合成，是一个容器，用来存放各种素材的容器，缩写是: compXXX, 比如 compDurationUs，则表示合成的时长,单位是微秒(Us)
2. original: 原始的, cut:时间裁剪; duration:时长;

3.2、合成容器:ConcatComposition

1. 字面意思是:拼接合成. 是以图片和视频的拼接的总时长作为最终导出视频的时长，容器有 3 大层，分别是:背景图层 ，拼接类图层 ，叠加类图层.
.
2. 背景层: 给容器整体增加一个背景图层，支持:颜色背景，图片背景，如果你要给不同的图层设置不同的背景,则可以用图层的画布/背景功能来分别设置;

3. 拼接层: 把用户选中的图片和视频前后拼接在一起,是前后叠加. 可以在开始前增加，也可以做预览过程中插入，可以指定时间点插入，可复制，可删除，可调节每个图层的前后位置，支持手势调节，支持图层的各种方法.

4. 叠加类图层: 在拼接层上面,叠加多个其他图层，是上下叠加. 比如叠加水印，叠加文字，叠加动画等等这些称之为叠加类型的图层，图层可调节上下位置，可设置显示开始时间点，可设置显示时长;可叠加，可删除，支持手势,旋转移动缩放滤镜马赛克等等，可执行父类图层的所有方法.
5. 这些 API 的关键词是: 在 LSOConcatCompositionView 中;，拼接:(concat)，叠加:(overLay)，背景(backGround);

3.3、视频和图片的混合编辑

1. 当前大部分视频编辑是拼接编辑，比如您选中了几个视频和几个图片，则是视频和图片混合的前后拼接合成的编辑.
2. 可以是单个视频，也可以是多个视频，也可以是一张图片或多种图片，或者图片和视频混合编辑.

3. 视频增加后,当前默认是整个视频时长增加进去，增加后，你可以设置视频时长,容器的总时长会改变，得到图层对象,可执行图层的各种方法.
4. 图片增加后，默认每张图片显示 3 秒，可调节显示时长，可复制,删除,插入图片,拼接图片不可以循环，得到图层对象,可执行图层的各种方法.
5. 如果设置了转场,则前一个层和后一层有部分画面是重叠，重叠时间可设置．设置转场后,
6. 最终容器的总长度是各个图片和视频的长度之和 ,如果有转场或视频变速,则对应的时长会改变;

3.4、图层之--时间调节

1. 素材本身的的时长．比如视频时长,声音时长，图片在放入后，默认是显示 3 秒;图片序列和 gif 等是其本身的时长；这个时长是素材本身的，我们称之为 AssetDuration 是资源时长，只能读取,不能修改，但你可以修改素材在容器中的显示时间和从什么时间点开始显示;
2. 可以修改的有：素材在容器中的显示时间点，是否循环，显示时长，如果循环,则会从显示时间点开始，一直循环到整个容器尾部，时间可以实时设置，实时调节,实时获取,当显示时长改变后，如果你要获取对应的缩略图,则缩略图中图片数组也会相应的改变;
3. 如果是视频素材，则支持裁剪视频长度，两个方法是: setCutStartTime 和 setCutEndTime,从视频的什么时间开始裁剪,和裁剪到什么地方；视频裁剪当前没有约束，但建议你最小裁剪值是 1 秒;
4. 这些 API 的名字是:资源时长(assetDuration)，当前图层的显示时长(displayDuration)，从容器的什么时间点开始显示:(startTimeOfComp);

3.5、图层之--手势 + 移动/缩放/旋转

1. 容器在创建时设置了宽度和高度，素材增加进去后，会默认放到容器的中间，即容器的宽度/2 和高度/2;
2. 每增加一个素材，会得到一个图层对象，图层都支持用手指直接操作屏幕 来完成 移动缩放旋转的 touch 事件,，你可以单指移动，双指缩放和旋转；当选中一个图层时，它的周边会出现一个红色的方框，表示选中当前图层，为了避免在操作时误触别的图层，你可以把别的图层锁定，我们提供丰富的 API 让你设置或获取相关的坐标，我们甚至定义了一个类 LSORect 来让你获取相对应当前预览窗的相对坐标 和 相对于容器中的绝对坐标;
3. 我们另外定义了各种常见的枚举类型，比如 4 种缩放类型，和上/下/左/右/中/左上/左下/右上/右下这 9 个位置点供您选择；提供缩放系数，缩放到实际大小,设置具体位置，相对屏幕位置等

4. 如果你要用代码的形式设置，则设置的位置是当前图层中心点的位置．可以缩放到指定大小，旋转则 0 度则是正上方为 0 点，角度是任意角度，比如设置为-100 度, -400 度，或 370 度, 540 度等等.
5. 这些 API 的关键词是：缩放大小(scaleSize)，缩放类型(scaleType)，缩放系数(scaleFactor)，位置类型(positionType)，位置中心点(centerPoint)，位置相对预览窗口坐标(positionInView)，旋转（rotation), setTouchEnable , xxxInView(是以当前预览为单位的宽高和位置;)

3.6、图层之--贴纸/水印/文字

1. 贴纸，水印，文字，在我们图层处理架构里，是以图像的形式呈现的，因为手机 GPU 处理的是图像，贴纸，水印，文字在底层是一个意思，认为是一个图像或一组连续的图像做处理.
2. 文字在 SDK 内部会各种转换方法，让你方便的以文字图层操作，操作完毕后底层会转换为图像来处理,如果我们提供的绘制文字不够详细，你也可以自己绘制,然后转换为图片或图片数组,设置到容器中.
3. 贴纸和水印是一个意思,我们提供了单个图片的图层,多张联系图片组成的图层,Gif 图层等，这些你可以方便的设置，可以设置一个,也可以设置多个.
4. 图层的增加是通过容器 Composition 的方法实现的，增加后，得到对应的图层对象;
5. 这些 API 的关键词是: android 的图片图层(BitmapLayer) , android 的图片序列图层(BitmapListLayer), ios 的图片图层(ImageLayer), ios 的图片序列图层(ImageListLayer), Gif 图层.

3.7、图层之--颜色调节/滤镜/美颜

1. 滤镜是对图层的画面做调节，和图层的缩放移动一样是图层的一个属性，颜色调节和滤镜和美颜，都是滤镜的不同效果．当前我们的美颜是仅对整体画面做磨皮和美白处理,没有做脸识别后的各种功能.
2. 当前支持的颜色调节包括：亮度/ 对比度/ 饱和度/ 锐化/色温/白平衡/色调/曝光度
3. 当前支持 80 种滤镜，包括常见的 17 种滤镜,当前建议你使用这 17 种滤镜，这些滤镜在我们 demo 里可以看到.
4. 这些 API 的关键词是：设置滤镜(setFilter,设置会把上一个替换掉),增加滤镜(addFilter)，移出滤镜(removeFilter),删除所有滤镜(removeAllFilter);

3.8、图层之--转场设置

1. 转场是两个素材前后拼接时播放时的一种过渡效果，比如上一段淡淡的消失，下一段逐渐出现等.
2. 当前的转场是 AE 设计师在 PC 端用 after Effect 设置好的，然后按照我们的格式，导出为 json 文档,SDK 还原出来在 PC 端设计的效果. 这样的好处是：开发人员不用自己用代码实现各种效果,只需要根据不同的效果,把对应的 json 送到 SDK 中，SDK 就实现了这样的效果，简单高效，并可以做到 Android 和 ios 的统一.
3. PC 端用 Ae 软件制作规范：用一张小于等于 544x960 的图片创建一个合成，然后对其做旋转缩放透明处理，形成动画；或对其做 mask 遮罩处理,形成动画；当前不支持同时做旋转缩放透明+ mask 的效果；合成的分辨率应小于等于 544x960，时长建议不大于 3 秒，以实际效果为准；
4. 当前支持各种遮罩转场和基础的移动旋转缩放转场；转场设置是通过每个具体的图层设置的，比如有图层 ABC3 个，要在 AB 之前增加转场,则设置 A 的 transition；要给 BC 之间加转场,则设置 B 的 transition；
5. 转场的时间可以设置,默认在是设置路径后，会默认是 1 秒钟，最大可设置为 5 秒钟；当前仅支持遮罩转场，即各种形式的透明.
6. 这些 API 的关键词是:转场(transition)；设置转场时间:(transitionDuration)，播放转场(playTransition)，取消(cancelTransition)；
7. 执行步骤是：你需要先设置转场路径，设置后，默认转场时间是 1 秒钟，如需预览，则 playtransition； 调用 cancelTransition 取消转场； 在设置路径后，你可以设置转场时间；设置后，如需要播放则 playTransition；如果你要应用到全局，则需要把一个 json 文件循环设置到每个图层对象；我们内部会共用同一个 json 内容；

3.9、图层之--入场动画/出场动画/指定时间点动画

1. 入场动画，出场动画，和转场一样，也是 Ae 设计师在 PC 端设计好后，导出为 json,送入到 SDK 中. SDK 解析还原出对应的效果.
2. PC 端用 Ae 软件制作动画规范：用一张小于等于 544x960 的图片创建一个合成，然后对其做旋转缩放透明处理，形成动画；或对其做 mask 遮罩处理,形成动画；合成的分辨率应小于等于 544x960，时长建议不大于 3 秒，，以实际效果为准；如果是旋转缩放透明则帧率建议是 40，若是 mask 遮罩,则帧率建议 25；
3. 因入场动画和出场动画从技术上讲，仅仅是插入到图层的时间点不同，SDK 可以设置开始从图层的哪个时间点开始，当前入场动画默认是 0，出场动画默认最后 json 的时间时长； 你可以调节这个出入场的持续时间，持续时间调节后，会把调节后的时长作为整个动画走完的时间，如果你设置的短,则动画走的快；如

果设置的长,则动画走的慢； 入场动画和出场动画是 setXXX 的方式， 每次设置后， 会把之前的动画移出；同一时刻只能有一个入场或出场动画；

4. 指定时间点动画，是你可以指定一个时间点增加动画，可以实时设置开始时间点，也可以设置动画时长，可以增加多个，每个素材最大支持 20 个动画。在每次设置后， 可预览或移除；
5. 这些 API 的关键词是： 动画:(Animation)；入场动画/出场动画:(setAnimationAtLayerHead/End)，指定时间点动画:(addAnimationAtCompTimeUs)预览动画 playAnimation；移除 removeAnimation;
6. 执行步骤是：增加 json 文件，会得到 LSOAnimation 对象,用这个对象可以 playAnimation 来预览; removeAnimation 删除；对象里有设置开始时间点和播放时长的方法;. 等要删除时,用 removeAnimation；当一个图层从容器中释放后，我们会销毁里面的所有的对象;

3.10、图层之--特效

1. 和动画一样的制作步骤，唯一不同的是：动画是从开始播放到结束，如果设置的时长不等于 json 中的默认时长,动画平均分配不同的动画快慢来达到设置的长度;而特效的效果时间是固定的，如果设置的时长 json 中的默认时长,则会循环播放特效;
2. 特效可设置当前图层的任意时刻点开始,我们举例了从头(head) 和尾(end)的两种方式;
3. 这些 API 的关键词是：特效:(Effect)，指定时间点增加特效:(addEffectAtCompTimeUs)；预览特效(playEffect)，移除特效:(removeEffect),
4. 执行步骤，和动画步骤一致;

3.11、图层之--画布(背景)/去水印/透明/镜像

1. 画布是对当前每个图层做的操作，是无论图层缩放多少，画布都会铺满整个容器,画布可以是颜色，图片，或画面的虚化做背景;
2. 去水印当前支持马赛克去水印，我们在图层类中举例了最大 4 个去水印的 API,可以设置位置,设置大小,设置马赛克像素的大小；如果你有更多个马赛克的需求,还是用 LanSongMosaicRectFilter 自行增加;
3. 透明我们描述为:不透明度，这样更好理解一些，和调节 RGB 一样,可以直接设置;参数 1.0 是完全不透明, 0.0 是完全透明;
4. 镜像:是把整个画面的左边镜像到右边，把右边镜像到左边；如果你打算把左边的一半镜像到右边,则用我们的 LanSongMirrorFilter 滤镜;

5. 这些 API 的关键词是：背景(backGround)，去水印(mosaicRect)，不透明度(opacityPercent)，镜像(mirror).

3.12、图层之--获取缩略图

1. 缩略图是每秒钟一帧，大小是 192x192 的图片对象;
2. 用 getDisplayThumbnailList 获取到每张图片大小是 192x192，每个图层最后获取到的图片宽度总和等于图层的缩略图显示时长(秒为单位)*192;
3. 我们有合成总时长改变回调(DurationChanged)：当原视频时长裁剪，变速,倒序,增加转场,都会触发此回调,建议在这里重新刷新所有的拼接层的时长，因为一个图片或视频的时长改变了，别的图片和视频的开始显示点等都会改变;
4. 我们的 demo 演示，是在 durationChanged 的时候刷新所有的拼接层的所有图层缩略图;

3.13、图层之--关键帧动画

1. 关键帧动画是指:在两个不同时间点分别对视频或图片做动作，然后 SDK 就会自动从一个时间点平滑过渡到另一个时间点状态的动画，称之为关键帧动画;
2. 比如你在时间 A 点把视频缩小一倍，在 B 点把视频放大一倍，则预览时 SDK 就会自动的从 A 点平滑的一点一点的放大视频,直到 B 点放大完毕,从而形成动画.
3. 当前关键帧动画支持 移动旋转缩放透明;
4. 关键帧的优先级最低，如果在同一时间段设置了关键帧和动画/特效/转场等 json 时，则动画/特效/转场有效果，关键帧无效;

3.14、图层之--调速

1. 视频支持调速，调速可设置 0.1 倍到 10.0 倍
2. 0.1 倍是放慢 10 倍，画面显示时长等于原来的 10 倍, 10.0 倍是加快 10 倍，画面时长等于原来的 1/10; 1.0 倍是原来的速度，可以很细小的调节,比如 0.1, 0.2, 0.3 等;
3. 当前在调速时，暂时不支持声音变速；声音默认是静音的；调速后,会触发容器的 durationChanged 回调，建议你在回调中去刷新所有缩略图;
4. 对应的 API 是: setVideoSpeed, getVideoSpeed;

3.15、声音图层

1. 外界输入的各种音效，无论是 wav 格式的音效，还是 mp3,m4a，或者录音 或者带有声音的视频，我们统称为声音. 这些声音是同一个 API 输入到 SDK 中，然后得到对应的声音图层.

2. 声音图层，可以指定时间点增加，可以设置循环，设置把声音的那一段裁剪后增加，设置音量大小，设置淡入淡出效果
3. 声音音量范围是 0---5.0 , 1.0 是默认声音. 0.0 是关闭声音; 5.0 是放大 5 倍；声音裁剪最小是 300 毫秒，最大是声音的原始长度;
4. 这些 API 的关键词是：音频图层:(LSOAudioLayer)；裁剪时长(setCutStartTime/setCutEndTime) 音量:(volume)

3.16、API 介绍

1、Android 视频播放 LSOEditPlayer 类说明

一、视频播放 LSOEditPlayer 类	
1	public void onCreateAsync(LSORatioType ratio, OnCreateListener listener) **功能：** • 设置播放容器宽高比; • 在 Activity 的 onCreate 中调用,设置播放容器的宽高比，设置后，会根据宽高比等比例缩放到当前 view 所在 layout 中; • 也可以在播放暂停时，重写布局此播放容器(relayout); **参数：** • ratio 宽高比; • listener 重新布局后，适配到 layout 后的回调;
2	public void onCreateAsync(List<LSOAsset> arrays, OnCreateListener listener) **功能：** • 设置播放容器的资源 **参数：** • arrays 容器资源数组，支持图片和视频. • listener 默认会以第一个图片或视频的宽度为参考,等比例缩放当前播放容器后回调;
3	public void onCreateAsync(List<LSOAsset> arrays,int width, int height, OnCreateListener listener) **功能：** • 设置播放容器资源，并设置宽度和高度 **参数：** • arrays 输入的图片或视频资源; • width 播放容器的宽度 • height 播放容器的高度 • listener 根据设置的高度和宽度，重新等比例布局当前 view，布局后返回监听;
4	public void onCreateAsync(int width,int height, OnCreateListener listener)

	功能： • 设置容器的宽度和高度，在播放设置，不可以用来初始化播放器; • 在第一次之后设置 **参数：** • width 宽度 • height 高度 • listener 适配当前 view 后的监听;
5	**public void onResumeAsync(OnResumeListener listener)** **功能：** • 在 Activity 的 onResume 中调用，用来执行 activity 的返回事件; **参数：** • listener 异步回调;
6	**public void onPause()** **功能：** • 在 Activity 中的 onPause 中调用;
7	**public void onDestroy()** **功能：** • 在 Activity 的 onDestroy 中调用
8	**public void prepareConcatAssets(OnAddAssetProgressListener listener)** **功能：** • 在当前 View 布局完毕后，准备当前播放器; **参数：** • listener 内部会开启一个线程，内部开启完毕后的回调;
9	**public void insertConcatAssetAtCurrentTime(List<LSOAsset> assetArray, OnAddAssetProgressListener listener1)** **功能：** • 在当前时间点插入图片或视频 **参数：** • assetArray 插入的图片或视频数组 • listener1 插入完毕后的回调;
10	**public void insertConcatAssetWithTime(List<LSOAsset> assetArray, long atCompUs, OnAddAssetProgressListener listener1)**

	功能： • 在指定的时间点插入图片或视频. • 内部流程是: • 先找指定时间点 在某个图层时间段内. • 找到图层后，如果时间点在图层前半部分,则向前插入；如后半部分,则向后插入; **参数：** • assetArray 插入的图片或视频数组 • atCompUs 在指定的时间点 • listener1 异步插入完毕后的回调;
11	**public void replaceConcatLayerAsync(LSOAsset asset, LSOLayer replaceLayer, OnAddAssetProgressListener listener)** **功能：** • 替换拼接图层; **参数：** • asset 资源 • replaceLayer 被替换的图层 • listener 异步替换;
12	**public LSOLayer getCurrentConcatLayerByTime(long compUs)** **功能：** • 获取指定时间点的拼接图层 **参数：** • compUs 指定时间点，单位微秒; 1 秒=1000*1000 微秒 • @return 返回图层对象
13	**public void addAssetAsync(LSOAsset asset, long atCompUs,OnAddAssetProgressListener listener1)** **功能：** • 在拼接层上叠加一个资源; **参数：** • asset 图片或视频资源 • atCompUs 指定时间点 • listener1 异步增加完毕后的回调监听;
14	**public void addVideoEffectAsync(LSOAsset asset, long atCompUs, boolean preview, OnAddAssetProgressListener listener1)** **功能：** • 在指定时间点增加一个视频特效，增加后，通过异步返回一个 LSOLayer 对象; **参数：** • asset 一个 color 和一个 mask 组成的透明视频动画资源，宽高:最大 720p，时长:最大 5 秒钟; • atCompUs 在指定时间点 • preview 插入后，是否预览 • listener1 插入完毕后的回调,回调中

15	public LSOLayer addBitmapLayer(String path, long atCompUs) **功能：** • 在拼接层上叠加一个图片图层 **参数：** • path 图片的完整路径 • atCompUs 在指定时间点增加，单位微秒
16	public LSOLayer addBitmapLayer(Bitmap bmp, long atCompUs) **功能：** • 增加图片图层， **参数：** • bmp 图片对象； • atCompUs 在指定时间点增加,单位微秒；
17	public LSOLayer addGifLayer(String gifPath, long atCompUs) **功能：** • 在拼接层上，增加一个 Gif 图层 **参数：** • gifPath gif 图层路径 • atCompUs 在指定时间点增加,单位微秒；
18	public LSOLayer addGifLayer(LSOAsset asset, long atCompUs) **功能：** • 在拼接层上，增加一个 Gif 图层 **参数：** • asset gif 文件的路径 • atCompUs 在指定时间点增加,单位微秒； • @return 返回 LSOLayer 图层对象. 此对象可设置图层的移动旋转缩放等功能.
19	public LSOAudioLayer addAudioLayer(String path, long startTimeOfComp) **功能：** • 增加声音图层； **参数：** • path 声音的完整路径； • startTimeOfComp 从合成的什么位置开始增加； • @return 返回声音对象，可以设置声音的音量，循环等功能；
20	public void removeLayerAsync(LSOLayer layer) **功能：** • 异步删除图层； • 删除成功后，会触发容器时长回调，在回调中你可以重新布局时间轴 **参数：** • layer 图层对象；

21	public void removeAllOverlayLayersAsync() **功能：** • 删除所有的叠加图层. • 叠加图层有:图片图层, gif 图层，图片序列图层等;
22	public void removeAudioLayerAsync(LSOAudioLayer layer) **功能：** • 删除声音图层 **参数：** • layer
23	public void removeALLAudioLayer() **功能：** • 删除所有的增加的声音图层;
24	public boolean isPlaying() **功能：** • 画面是否在播放 **参数：** • @return 是否在播放
25	public boolean isRunning() **功能：** • 当前播放器是否在运行. • 合成是一个线程，此返回的是当前线程是否在运行, • 线程运行不一定画面在播放,有可能画面暂停; **参数：** • @return 线程是否在运行
26	public boolean isExporting() **功能：** • 是否在导出; **参数：** • @return
27	public long getCurrentPositionUs() **功能：** • 获取当前播放器的时间. **参数：** • @return 当前时间，单位微秒
28	public List<LSOLayer> getAllConcatLayers()

	功能： • 获取播放器的所有拼接图层; **参数：** • @return 图层 list 拼接图层;
29	public List<LSOLayer> getAllOverLayLayers() **功能：** • 获取播放器中的所有叠加层 **参数：** • @return 图层数组，叠加的图层;
30	public List<LSOAudioLayer> getAllAudioLayers() **功能：** • 获取播放器中的所有声音图层
31	public void setOnBeforeRenderFrameListener(OnLanSongSDKBeforeRenderFrameListener listener) **功能：** • 在每一帧绘制前的回调，里面没有经过 handler，你增加的代码，在我们 render 线程中执行, **参数：** • listener 返回的是时间戳，单位 us;
32	public void setOnDurationChangedListener(OnLanSongSDKDurationChangedListener listener) **功能：** • 容器的总时长改变监听；播放器会在拼接层裁剪，设置显示时间，转场,变速等场合下，时长改变; • 返回的是当前总时长; **参数：** • listener 总时长改变监听，返回的 long 类型改变后的当前总时长
33	public void setOnLanSongSDKPlayProgressListener(OnLanSongSDKPlayProgressListener listener) **功能：** • 播放进度回调 • 监听中的两个参数是: onLanSongSDKExportProgress(long ptsUs, int percent); • 分别对应 当前处理的时间戳 和百分比; • 在 seek 或 pause 的时候,此监听不调用; **参数：** • listener
34	public void setOnTimeChangedListener(OnLanSongSDKTimeChangedListener listener) **功能：** • 容器的当前播放时间改变回调; • 只要是改变了，不管是 seek 的改变,还是自动播放的改变,都执行这里; **参数：** • listener 时间改变监听 long 类型是当前时间戳, int 类型是当前时间戳占总时长的百分比;
35	public void setOnPlayCompletedListener(OnLanSongSDKPlayCompletedListener listener)

	功能： • 视频播放完成进度; **参数：** • listener 播放完成监听
36	public void setOnExportProgressListener(OnLanSongSDKExportProgressListener listener)
	功能： • 导出进度回调; • 监听中的两个参数是: onLanSongSDKExportProgress(long ptsUs, int percent); • 分别对应 当前处理的时间戳 和百分比; **参数：** • listener 导出进度监听, long 类型是当前正在处理的时间戳, int 类型是进度百分比;
37	public void setOnExportCompletedListener(OnLanSongSDKExportCompletedListener listener)
	功能： • 导出完成回调; • 完成后，有 void onLanSongSDKExportCompleted(String dstVideo); • 对应的是:返回完成后的目标视频路径; **参数：** • listener 导出完成监听，返回 String 类型的导出视频完整路径;
38	public void setOnErrorListener(OnLanSongSDKErrorListener listener)
	功能： • 错误监听
39	public boolean start()
	功能： • 开始播放/恢复播放;
40	public void startExport(LSOExportType type)
	功能： • 开始导出 **参数：** • type 导出枚举类型
41	public void seekToTimeUs(long timeUs)
	功能： • 定位到某个位置. **参数：** • timeUs 时间，单位微秒;
42	public void pause()
	功能： • 播放预览暂停
43	public long getDurationUs()

	功能： • 获取当前播放器的总时长; **参数：** @return 时间，单位微秒;
44	public void setLooping(boolean is) **功能：** • 设置容器循环播放; **参数：** • is 是否循环;
45	public void cancelExport() **功能：** • 取消导出
46	public void cancel() **功能：** • 取消当前合成. • 取消后，会把内部线程全部退出，所有增加的图层都会释放;
47	public void setFrameRate(int frameRate) **功能：** • 设置导出帧率 • 不建议使用 **参数：** • frameRate 帧率最小 20,最大是 60
48	public void setExportBitRate(int bitRate) **功能：** • 设置导出时的码率, • 不建议使用 **参数：** • bitRate 码率，最小是 300K,不建议使用;

2、Android 视频编辑图层 LSOLayer 类说明

二、视频编辑图层 LSOLayer 类	
1	public String getOriginalPath()

	功能： • 获取原视频/图片的路径 **参数：** • @return
2	public long getOriginalDurationUs() **功能：** • 获取视频/图片的原始时长 **参数：** • @return
3	public int getOriginalWidth() **功能：** • 获取原始宽度 **参数：** • @return
4	public int getOriginalHeight() **功能：** • 获取原始高度 **参数：** • @return
5	public long getDisplayDurationUs() **功能：** • 获取显示时长; • 如果播放器中只有一个视频或图片，则此时长等于播放器的时长; **参数：** • @return
6	public long getThumbnailDurationUs() **功能：** • 获取缩略图的显示时间; • 注：因转场等功能,拼接层可能有重叠区域．因此显示时长不等于缩略图的时长; **参数：** • @return
7	public void setDisplayDurationUs(long durationUs) **功能：** • 设置显示时长; **参数：** • durationUs
8	public long getStartTimeOfComp()

	功能： • 获取当前图层在播放器中的开始时间 **参数：** • @return
9	public void setCutDurationUs(long startUs, long endUs) **功能：** • 设置视频的裁剪时长 **参数：** • startUs 开始裁剪时间 • endUs 结束裁剪时间
10	public long getCutStartTimeUs() **功能：** • 获取裁剪开始时间 **参数：** • @return
11	public long getCutEndTimeUs() **功能：** • 获取裁剪结束时间 **参数：** • @return
12	public void setCropRect(LSORect rect) **功能：** • 设置裁剪区域; **参数：** • @param rect
13	public void setCropRectToOriginal() **功能：** • 恢复到不裁剪
14	public void setCropRectPercent(float x, float y, float width, float height) **功能：** • 设置裁剪百分比.整个画面从左到右是 0.0--1.0; 从上到下是 0.0--1.0; **参数：** • x 宽度的开始比例 • y 裁剪高的开始比例 • width 裁剪宽度的比例 • height 裁剪高度的比例
15	public void setLooping(boolean is)

	功能： • 叠加层设置循环 **参数：** • is
16	public void setStartTimeOfComp(long atCompUs) **功能：** • 设置当前图层在播放器中的开始时间点; **参数：** • atCompUs
17	public void setVisibility(boolean is) **功能：** • 设置是否显示 **参数：** • is
18	public void setVisibility(int visibility) **功能：** • 设置是否显示 • 类型是 LSOLayer.VISIBLE 和 LSOLayer.INVISIBLE **参数：** • visibility
19	public int getVisibility() **功能：** • 获取是否显示 **参数：** • @return
20	public void resetScaleSize() **功能：** • 恢复到缩放前的大小
21	public void setScaleType(LSOScaleType type)

	功能： • 设置缩放枚举类型. • NONE：无缩放形式.则内部会根据不同的素材,采用默认形式; • ORIGINAL：原始大小,直接放入，不做任意处理; • FILL_COMPOSITION，忽略百分比,把宽度等于容器的宽度，高度等于容器的高度,填满整个容器,这样有些画面可能会变形; • CROP_FILL_COMPOSITION：裁剪填满 放入到容器中;把视频的中心点放到容器的中心点,然后 把画面等比例提填满整个容器,把多的部分裁剪掉. • VIDEO_SCALE_TYPE:视频缩放模式,如果视频宽度大于高度，则宽度和容器对齐，然后等比例调整高度;如果高度大于宽度，则反之. **参数：** • type
22	public void setScaledValue(float width, float height) **功能：** • 缩放到的实际值 **参数：** • width • height
23	public float getScaleWidth() **功能：** • 获取缩放的宽度 **参数：** • @return
24	public float getScaleHeight() **功能：** • 获取缩放的高度 **参数：** • @return
25	public float getRotation() **功能：** • 获取旋转角度. **参数：** • @return
26	public void setRotation(float angle) **功能：** • 设置旋转角度 **参数：** • angle

27	public void setLayerMirror(boolean flipHorizontal, boolean flipVertical) **功能：** • 设置图层是否镜像，上下镜像，左右镜像. **参数：** • flipHorizontal 水平镜像 • flipVertical 垂直镜像
28	public void cancelLayerMirror() **功能：** • 取消镜像
29	public boolean isMirrorX() **功能：** • 水平是否镜像 **参数：** • @return
30	public boolean isMirrorY() **功能：** • 垂直是否镜像 **参数：** • @return
31	public void setPosition(float xPos, float yPos) **功能：** • 设置当前图层中心点的坐标, xPos 是 X 轴的坐标,yPos 是设置 Y 轴的坐标. • 此坐标是以播放容器的大小为参考，比如容器的大小是 1280x720，您设置了 640,360 ，则图层在播放器中居中显示 **参数：** • xPos • yPos
32	public void setPosition(LSOLayerPosition position) **功能：** • 设置图层的枚举位置; • 枚举类型有：左上/左下/右上/右下/居中/上/下/左/右
33	public float getPositionX() **功能：** • 获取当前图层中心点的位置 X **参数：** • @return 横向坐标中心点，相对于播放容器本身而言;
34	public float getPositionY()

	功能： • 获取当前图层中心点的位置 Y **参数：** • @return 竖向坐标中心点，相对于播放容器本身而言
35	public void setOpacityPercent(float percent) **功能：** • 设置透明百分比; **参数：** • percent 百分比 范围从 0--1.0;
36	public void setBrightnessPercent2X(float percent2X) **功能：** • 调节当前图层画面的亮度 **参数：** • percent2X 范围是 0--2.0; 1.0 为默认值;
37	public void setContrastFilterPercent2X(float percent2X) **功能：** • 调节当前图层画面的对比度 **参数：** • percent2X 范围是 0--2.0; 1.0 为默认值;
38	public void setSaturationFilterPercent2X(float percent2X) **功能：** • 调节画面的饱和度; **参数：** • percent2X 范围是 0--2.0; 1.0 为默认值;
39	public void setWhiteBalanceFilterPercent2X(float percent2X) **功能：** • 调节画面的白平衡 **参数：** • percent2X 范围是 0--2.0; 1.0 为默认值;
40	public void setHueFilterPercent2X(float percent2X) **功能：** • 调节画面的色度 **参数：** • percent2X 范围是 0--2.0; 1.0 为默认值;
41	public void setExposurePercent2X(float percent2X)

	功能： • 调节画面的曝光度 **参数：** • percent2X 范围是 0--2.0; 1.0 为默认值;
42	public void setBeautyLevel(float level) **功能：** • 设置画面的磨皮级别 0.0--1.0; **参数：** • level 等级 0.0 是不磨皮, 1.0 是最高磨皮
43	public void setFilter(LanSongFilter filter) **功能：** • 设置滤镜，设置时，会把上一次设置的滤镜删除；如果不想删除则用 addFilter; **参数：** • filter 滤镜对象，此对象不可同时设置为多个图层中.
44	public void removeFilter(LanSongFilter filter) **功能：** • 移出滤镜 **参数：** • filter 增加的滤镜对象
45	public void addFilter(LanSongFilter filter) **功能：** • 增加滤镜,如增加多个，则多个滤镜是级联的关系，即画面把执行上一个滤镜的结果，作为下一个滤镜的输入; **参数：** • filter 滤镜对象
46	public void removeAllFilter() **功能：** • 删除所有滤镜
47	public void setVideoSpeed(float speed) **功能：** • 设置视频速度. **参数：** • speed 视频速度值，范围 0.1--10.0;
48	public float getVideoSpeed() **功能：** • 获取视频速度; **参数：** • @return 返回速度值

49	public void setVideoReverseAsync(boolean reverse, OnVideoReverseListener listener) **功能：** • 异步设置视频倒序 **参数：** • reverse 是否倒序 • listener 倒序后的监听;
50	public boolean isVideoReverse() **功能：** • 当前视频是否倒序 **参数：** • @return 是否倒序
51	public LSOEffect addEffectAtCompTimeUs(String jsonPath, long compUs) **功能：** • 用 json 的形式增加特效 **参数：** • jsonPath 增加一个特效 json 路径 • compUs 从播放器的什么时间点开始增加 • @return 增加后，返回特效对象，可设置开始时间和播放时长;
52	public LSOEffect addEffectAtLayerHead(String jsonPath) **功能：** • 在图层的头部增加一个特效 **参数：** • jsonPath json 路径 • @return 增加后，返回特效对象，可设置播放时长;
53	public LSOEffect addEffectAtLayerEnd(String jsonPath) **功能：** • 在图层的尾部增加一个特效 **参数：** • jsonPath 特效路径 • @return 增加后，返回特效对象，可设置播放时长;
54	public List<LSOEffect> getAllEffectList() **功能：** • 获取增加的所有特效 **参数：** • @return 所有特效数组
55	public void removeAllEffectList() **功能：** • 移出当前图层的所有特效

56	public void removeEffect(LSOEffect effect) **功能：** • 移出指定的特效 **参数：** • effect
57	public void playEffect(LSOEffect effect) **功能：** • 播放预览特效 **参数：** • effect 指定播放的特效对象,播放后会回到播放前的位置
58	public LSOAnimation addAnimationAtCompTimeUs(String jsonPath, long compUs) **功能：** • 增加动画 **参数：** • jsonPath json 格式的路径 • compUs 从播放器的时间点增加 • @return 返回动画对象，可设置开始时间和播放时长
59	public LSOAnimation setAnimationAtLayerHead(String jsonPath) **功能：** • 在图层的头部增加一个动画 **参数：** • jsonPath json 动画路径 • @return 返回动画对象，可设置播放时长
60	public LSOAnimation setAnimationAtLayerEnd(String jsonPath) **功能：** • 在图层的尾部增加一个动画 **参数：** • jsonPath 动画路径 • @return 返回动画对象，可设置播放时长
61	public List<LSOAnimation> getAllAnimationList() **功能：** • 获取所有动画 **参数：** • @return 所有动画对象
62	public void removeAllAnimationList() **功能：** • 移出所有动画
63	public void removeAnimation(LSOAnimation animation)

	功能： • 移出指定动画 **参数：** • animation 指定动画对象
64	public void playAnimation(LSOAnimation animation) **功能：** • 播放动画 **参数：** • animation 动画对象
65	public boolean setTransitionMaskPath(String maskJsonPath) **功能：** • 设置转场路径 **参数：** • maskJsonPath 遮罩转场路径 • @return 可以设置返回 true; 否则返回 false
66	public void setTransitionDurationUs(long duration) **功能：** • 设置转场时长 **参数：** • duration 时长,范围 100*100---3*1000*1000;
67	public void playTransition() **功能：** • 开始播放转场
68	public void cancelTransition() **功能：** • 取消转场
69	protected long getTransitionDurationUs() **功能：** • 获取转场时长; **参数：** • @return
70	public long getTransitionStartTimeOfComp() **功能：** • 获取从合成的开始时间; **参数：** • @return
71	public LSOMosaicRect getMosaicRect1()

	功能： • 获取一个马赛克区域，返回马赛克对象， • 可设置马赛克区域，是否禁止,马赛克像素点的宽度; **参数：** • @return 马赛克对象
72	public void setAudioVolume(float volume) **功能：** • 设置音频音量
73	public float getAudioVolume() **功能：** • 获取音频音量
74	public void setBackGroundBlurLevel(float level) **功能：** • 设置背景模糊级别 • 1.0 是轻微模糊,毛玻璃模糊. 8.0f 是完全,深度模糊; 0.0 是删除模糊效果; **参数：** • level
75	public void setBackGroundBitmap(String bitmapPath) **功能：** • 设置背景图片 **参数：** • bitmapPath
76	public void setBackGroundColor(int color) **功能：** • 设置背景颜色 **参数：** • color
77	public float getMaxBlurLevel() **功能：** • 获取最大模糊系数，当前返回的是 8.0;
78	public void getThumbnailAsync(OnLanSongSDKThumbnailBitmapListener listener) **功能：** • 异步获取缩略图 **参数：** • listener
79	public List<Bitmap> getThumbnailListWithCount(int count)

	功能： • 获取缩略图个数，此方法一定在 getThumbnailAsync 完成后设置 **参数：** • count 获取的图片张数 • @return 返回 bitmap 数组;

3、IOS 视频播放 LSOEditPlayer 类说明

一、视频播放 LSOEditPlayer 类	
1	- (id)initWithUrlArray:(NSArray<NSURL *> *)urlArray ratio:(LSOSizeRatio)ratio
	功能： • 初始化这个类，可以是一个图片或一个视频，也可以是多个视频．可以设置这个播放器的宽高比例，当比例是原始时，默认用数组中第一个 url 的宽高为默认宽高; • 当初始化失败后，返回 nil，大部分是没有授权文件时会失败; **参数：** • urlArray 用户选中的视频/图片 • ratio 容器的比例，如果原始比例，则填入 LSOSizeRatio_ORIGINAL,比例系数;
2	- (void)setCompositionView:(LSODisplayView *)view;
	功能： • 增加预览的显示窗口; • 和 AVPlayer 类似，视频播放器和播放 view 是分开的，你在创建好此对象后，需要拿到对象的 compositionSize，然后布局 LSODisplayView，再设置到此; • LSODisplayView 继承自 UIView,是一个显示控件;显示窗口一定要和合成(容器)的宽高成比例，可以不相等．我们提供最简单的集成方法，方便您参考; **参数：** • view 显示窗口
3	@property(readonly,atomic) CGFloat compDurationS
	功能： • 当前播放器的总时长.单位秒; (当你设置每个图层的时长后，此属性会改变．只读，不可写入)
4	@property (nonatomic,readonly) CGSize compositionSize
	功能： • 您在 init 的时候，设置的合成宽高．或设置的比例; • 当在运行中，调整播放器的比例后，此宽高是调整后的宽高
5	@property (nonatomic,assign) CGFloat frameRate
	功能： • 设置帧率, • [可选],范围是 10--60

6	@property (nonatomic,assign) int exportBitRate **功能：** • 设置码率，单位字节， • [可选],最低是 300*1024
7	@property(nonatomic, readonly) CGFloat currentTimeS **功能：** • 获取,当前播放的时间点,单位秒
8	@property (strong,atomic, readonly) NSMutableArray *overlayLayerArray **功能：** • 当前内部有多少个叠加层; • 如果你获取使用此变量，则会拷贝一份新的 NSMutableArray 返回. • 注意：请不要一直拷贝
9	@property (strong,atomic, readonly) NSMutableArray *concatLayerArray **功能：** 当前内部有多少个拼接层; 如果你获取使用此变量，则会拷贝一份新的 NSMutableArray 返回. 注意：请不要一直拷贝
10	@property (strong,atomic, readonly) NSMutableArray *audioLayerArray **功能：** • 当前有多少个音频层; • 如果你获取使用此变量，则会拷贝一份新的 NSMutableArray • 注意：请不要一直拷贝
11	@property (nonatomic, readwrite) LSOLayer *selectedLayer **功能：** • 设置一个图层为选中状态; • 当用户通过界面点击别的图层时,这个状态会被改变; • 可以设置，如果设置不选中，则设置为 nil
12	- (void)setBackGroundVideoLayerWithURL:(NSURL *)url completedHandler:(void (^)(LSOLayer *videoLayer))handler **功能：** • 设置一个背景视频;背景视频默认循环; **参数：** • url 背景视频的路径 • handler 背景视频设置完成时的回调
13	- (LSOLayer *)setBackGroundImageLayerWithImage:(UIImage *)image **功能：** • 设置一个背景图片
14	- (void)removeBackGroundLayer **功能：** • 删除背景图层
15	- (LSOLayer *)addVideoLayerWithURL:(NSURL *)url atTime:(CGFloat)startTimeS

	功能： • 增加一个视频叠加层. 叠加层是在拼接层的上面; **参数：** • url 视频的 Url 路径; • startTimeS atStartTime 在合成的指定时间开始增加
16	- (LSOLayer *)addImageLayerWithImage:(UIImage *)image atTime:(CGFloat)startTimeS **功能：** • 叠加一张图片图层; **参数：** • image 图片对象 • startTimeS 在播放器中的开始叠加时间
17	- (LSOLayer *)addGifLayerWithURL:(NSURL *)url atTime:(CGFloat)startTimeS **功能：** • 叠加一个 gif 图层 **参数：** • url gif 的完整 url 路径 • startTimeS 在播放器中的开始叠加时间
18	- (LSOLayer *)addMVWithColorURL:(NSURL *)colorUrl maskUrl:(NSURL *) maskUrl atTime:(CGFloat)startTimeS **功能：** • 叠加一个透明动画视频, 我们的透明动画是两个视频合并而成的. 一个是彩色视频/一个是黑白视频; **参数：** • colorUrl 透明动画视频中的彩色视频 • maskUrl 透明动画视频中的黑白视频 • startTimeS 从播放器中的开始叠加时间
19	- (LSOAudioLayer *)addAudioLayerWithURL:(NSURL *)url atTime:(CGFloat)startTimeS **功能：** • 增加一个声音图层; **参数：** • url url 路径 • startTimeS 开始时间
20	- (void)removeAudioLayer:(LSOAudioLayer *)audioLayer **功能：** • 删除声音图层 **参数：** • audioLayer 声音图层对象
21	-(BOOL)setOverlayerLayerPosition:(LSOLayer *)layer index:(int)index

	功能： • 设置叠加层的位置 **参数：** • layer 图层对象 • index 位置，最里层是 0
22	-(BOOL)setConcatLayerPosition:(LSOLayer *)layer index:(int)index
	功能： • 设置拼接层的位置. **参数：** • layer 拼接的图层 • index 图层的 index，从前到后，第一个层是 0；最后面的是: _concatLayerArray.count-1;
23	- (CGSize)updateCompositionRatio:(LSOSizeRatio)ratio
	功能： • 设置合成的比例 **参数：** • ratio 设置的比例
24	- (void)prepareConcatLayerAsync:(void (^)(NSArray *layerAray))handler
	功能： • 调用流程是，在 init 创建好此对象后，在需要播放时,调用 prepare 内部会根据输入的图片和视频数量，创建好对应的图层对象，在这个 handler 中返回;
25	- (void)insertConcatLayerWithImageArray:(NSArray<UIImage *> *)imageArray completedHandler:(void (^)(NSArray *layerAray))handler
	功能： • 增加拼接图片图层；可以是一张图片，或多张图片 **参数：** • imageArray <#imageArray description#> • handler 返回的 layerAray :是当前新增加的图层对象数组
26	- (void)insertConcatLayerWithArray:(NSArray<NSURL *> *)urlArray atTime:(CGFloat)compTimeS completedHandler:(void (^)(NSArray *layerArray,BOOL insertBefore))handler;
	功能： • 在容器的指定位置增加资源 • 在容器的指定时间点插入，时间点在图层前半部分,插入图层前,反之插入到图层后 **参数：** • urlArray url 数组 • compTimeS 在容器的指定时间点插入， • handle 完成后的回调,是当前新增加的图层对象数组
27	- (void)insertConcatLayerWitImageArray:(NSArray<UIImage *> *)imageArray atTime:(CGFloat)compTimeS completedHandler:(void (^)(NSArray *layerArray,BOOL insertBefore))handler

	功能： • 在容器的指定位置增加图片数组 **参数：** • imageArray 图片数组 • compTimeS 指定时间点插入 • handler 完成后的回调,是当前新增加的图层对象数组
28	- (void)replaceConcatLayerWithLayer:(LSOLayer *)currentLayer replaceUrl:(NSURL *)url completedHandler:(void (^)(LSOLayer *replacedLayer))handler
	功能： • 替换一个图层 **参数：** • currentLayer 当前要替换的图层 • url 要替换的路径 NSURL 格式 • handler 替换完成后的回调
29	-(void)splitConcatLayerByTime:(CGFloat)compTimeS layer:(LSOLayer *)layer
	功能： • 分割图层 **参数：** • compTimeS 从容器的相对时间点开始分割 • layer 要分割的图层对象，分割后的图层，会放到到拼接图层数组中，分离后需要更新时间轴
30	-(void)copyConcatLayerByLayer:(LSOLayer *)layer complete:(void(^)(void))completeBlock
	功能： • 复制一个拼接图层 **参数：** • layer 要复制的图层 • completeBlock 复制完毕后的回调；复制后的图层会放到到拼接图层数组中，分离后需要更新时间轴
31	-(LSOLayer *)copyVideoLayerByLayer:(LSOLayer *)layer
	功能： • 复制一个图层 **参数：** • layer 复制的图层对象
32	- (BOOL)removeLayer:(nullable LSOLayer *)layer
	功能： • 删除一个图层.
33	- (void)removeLayerArray:(nullable NSArray<LSOLayer *> *)layers

	功能： • 删除一组图层 **参数：** • layers 图层数组
34	- (BOOL)removeLayerByCompTime:(CGFloat)compTimeS
	功能： • 根据合成中的一个时间点，删除对应的图层 **参数：** • compTimeS 在合成中的时间,单位秒
35	-(LSOLayer *)getCurrentConcatLayerByTime:(CGFloat)compTimeS
	功能： • 根据当前在合成中的时间点，来获取一个图层对象; • 当前返回的是 LSOLayer 对象
36	- (void)applicationDidEnterBackground
	功能： • APP 进入后台时的回调; • 当用户从下向上滑动，让整个 APP 进入后台的时候, • 你可以调用整个方法，让合成线程进入后台
37	- (void)applicationDidBecomeActive
	功能： • APP 从后台恢复时的回调; • 当用户从 后台的状态下，恢复到当前界面，则调用这个 APP,恢复合成的运行
38	@property(nullable, nonatomic,copy) UIColor *backgroundColor
	功能： • 设置合成容器的背景颜色 • 当前暂时导出无用
39	-(BOOL)startPreview
	功能： • 播放预览
40	-(void)pause
	功能： • 暂停
41	-(void)resume
	功能： • 恢复播放
42	- (void)seekToTimeS:(CGFloat)timeS
	功能： • 定位到具体的时间戳; • 在调用后，会暂停当前界面的执行; • 你需要在完成 seek 后，调用 resume 来播放 • 预览有效
43	@property (readonly) BOOL isRunning

	功能： • 当前是否在运行
44	@property (readonly) BOOL isPausing
	功能： • 是否暂停
45	-(void)startExportWithRatio:(LSOExportSize)ratioType
	功能： • 开始导出
46	@property (readonly) BOOL isExporting
	功能： • 当前是否正在导出
47	-(void)cancel
	功能： • 取消整个合成线程 • 包括预览和导出，都取消
48	@property(nonatomic, copy) void(^playProgressBlock)(CGFloat progress,CGFloat percent)
	功能： • 进度回调, • 当在编码的时候，等于当前视频图层的视频播放进度 时间单位是秒;; • 工作在其他线程, • 如要工作在主线程,请使用: progress: 当前正在播放总合成的时间点,单位秒; percent: 当前总合成的时间点对应转换为的百分比; • 进度则是: progress/_duration; dispatch_async(dispatch_get_main_queue(), ^{ });
49	@property(nonatomic, copy) NSString *_Nullable(^mergeAVBlock)(NSString *video, NSString *audio)
	功能： • 适配 ios13 的 export 类问题
50	@property(nonatomic, copy) void(^playCompletionBlock)(void)
	功能： • 在异步线程执行此 block; 请用一下代码后调用; dispatch_async(dispatch_get_main_queue(), ^{ });
51	@property(nonatomic, copy) void(^exportProgressBlock)(CGFloat progress,CGFloat percent)
	功能： • 导出进度回调; • 在异步线程执行此 block; 请用一下代码后调用; dispatch_async(dispatch_get_main_queue(), ^{ });
52	@property(nonatomic, copy) void(^exportCompletionBlock)(NSString *_Nullable dstPath)

	功能： • 编码完成回调，完成后返回生成的视频路径; • 注意:生成的 dstPath 目标文件，我们不会删除. • 在异步线程执行此 block；请用一下代码后调用; dispatch_async(dispatch_get_main_queue(), ^{ });
53	@property(nonatomic, copy) void(^userSelectedLayerBlock)(LSOLayer *layer) **功能：** • 当前用户选中的图层回调; • 在异步线程执行此 block；请用一下代码后调用; dispatch_async(dispatch_get_main_queue(), ^{ });
54	@property(nonatomic, copy) void(^ _Nullable userTouchDownLayerBlock)(LSOLayer * _Nullable layer) **功能：** • 用户点击事件；用户手指按下. • 中间不需要增加 dispatch_async(dispatch_get_main_queue()
55	@property(nonatomic, copy) void(^ _Nullable userTouchMoveLayerBlock)(CGPoint point) **功能：** • 用户移动图层
56	@property(nonatomic, copy) void(^ _Nullable userTouchScaleLayerBlock)(CGSize size) **功能：** • 用户缩放图层事件
57	@property(nonatomic, copy) void(^ _Nullable userTouchRotationLayerBlock)(CGFloat rotation) **功能：** • 用户旋转图层
58	@property(nonatomic, copy) void(^ _Nullable userTouchUpLayerBlock)(void) **功能：** • 用户手指抬起
59	@property(nonatomic, copy) void(^compositionDurationChangedBlock)(CGFloat durationS) **功能：** • 合成容器时间改变回调; • 当整个容器的合成时间改变了，则触发回调; • 比如你对视频做裁剪,则会触发这里. • 在异步线程执行此 block；请用一下代码后调用; dispatch_async(dispatch_get_main_queue(), ^{ });
60	@property (nonatomic, assign) BOOL disableTouchEvent **功能：** • 禁止图层的 touch 事件

4、IOS 视频编辑图层 LSOLayer 类说明

二、视频编辑图层 LSOLayer 类	
1	@property (readonly,assign)CGFloat originalDurationS
	功能： • 素材的原始时长. • 比如视频本身的长度，只读.不会改变
2	@property (readwrite,assign) CGFloat displayDurationS
	功能： • 当前图层的显示时长; • 在没有裁剪时后变速等操作时，等于原始时长
3	@property (nonatomic,readonly) CGSize originalSize
	功能： • 输入资源的原始大小
4	@property (nonatomic,readonly) CGSize layerSize
	功能： • 图层在容器中的大小. • 视频放到容器中后，默认是对齐到边缘的,故图层大小会变; • 视频默认对齐到播放器的边缘. 图片居中显示
5	@property (readwrite,assign) BOOL looping
	功能： • 叠加图层是否循环播放; • 拼接图层设置后无效
6	@property(getter=isHidden) BOOL hidden
	功能： • 当前图层是否隐藏, • 可以用这个在新创建的图层做隐藏/显示的效果，类似闪烁，或创建好,暂时不显示等效果
7	@property (readwrite, nonatomic) CGFloat rotateAngle
	功能： • 角度值 0--360 度. 默认为 0.0 • 顺时针旋转
8	@property (readwrite, nonatomic) CGPoint centerPoint

	功能：
	• 设置或读取 <当前图层的中心点>在容器中的坐标;
	• 容器坐标的左上角是左上角为 0,0. 从上到下 是 Y 轴，从左到右是 X 轴;
	• 当一个图层增加到容器中，则默认是:
	positionX=drawPadSize.width/2;
	positionY=drawPadSize.height/2;

9	@property (readwrite,nonatomic) LSOPositionType positionType

	功能： • 设置当前位置，枚举类型. • 我们举例了常见的位置枚举类型: • 上下左右/左上/左下/右上/右下/居中，如果这些类型不满足您的需求，可以通过 centerPoint 来设置每个素材的位置
10	@property (readwrite,nonatomic) LSOScaleType scaleType
	功能： • 缩放的枚举类型; • kLSOScaleNONE：无缩放形式.则内部会根据不同的素材,采用默认形式; • kLSOScaleOriginal：原始大小 直接放入，不做任意处理; • kLSOScaleFillComposition：忽略百分比,把宽度等于容器的宽度，高度等于容器的高度,填满整个容器. • kLSOScaleCropFillComposition：匹配放入到容器中,等比例填满整个容器，把多的部分显示到容器的外面去; • kLSOScaleVideoScale:视频缩放模式,如果视频宽度大于高度，则宽度和容器对齐，然后等比例调整高度；反之亦然 • 如果以上不满足您的需求，则可以 scaleSize 设置具体缩放到的大小
11	@property (readwrite, nonatomic) CGSize scaleSize
	功能： • 把图层缩放到指定的大小; • 以像素为单位
12	@property (readwrite, nonatomic) CGSize scaleSizeInView
	功能： • 以显示窗口为单位，缩放大小，此宽高是相对于 LSODisplayView 而言; • 预览时有效
13	@property (readonly, nonatomic) CGSize layerSizeInView
	功能： • 以显示窗口为单位，获取图层的宽高; • 预览时有效
14	@property (readonly, nonatomic) BOOL isConcatImageLayer
	功能： • 是否是拼接图片层
15	@property (readonly, nonatomic) BOOL isConcatVideoLayer
	功能： • 是否拼接视频层
16	@property (readonly, nonatomic) BOOL isVideoLayer
	功能： • 是否是叠加的视频层
17	@property (readonly, nonatomic) BOOL isMVLayer
	功能： • 是否是叠加的 mv 透明动画层
18	@property (readonly, nonatomic) BOOL isImageLayer
	功能： • 是否是图片层
19	@property (readonly, nonatomic) BOOL isGifLayer

	功能： • 是否是 Gif 图层
20	@property (readonly, nonatomic) BOOL isImageArrayLayer
	功能： • 是否是图片数组图层
21	@property(readwrite,assign) CGFloat startTimeOfComp
	功能： • 在播放器中的开始时间，只工作在叠加的一些图层；比如叠加的图片/视频等
22	@property (readwrite, nonatomic) BOOL isConvertScalePosition
	功能： • 是否把缩放和位置的参数转换; • 默认是不转换
23	@property (nonatomic,assign)BOOL mirrorDrawX
	功能： • 在绘制的时候，横向图像镜像，左边的在右边，右边的在左边
24	@property (nonatomic,assign)BOOL mirrorDrawY
	功能： • 竖向图像镜像，上面的放下面，下面的放上面．默认不调整
25	@property(readwrite, nonatomic) CGFloat opacityPercent
	功能： • 不透明度. • 默认是 1.0；完全透明是 0.0
26	@property (readwrite,assign) CGFloat cutStartTimeS
	功能： • 裁剪时长的开始时间 • 相对于原视频的时间而言; • 仅对视频有效
27	@property (readwrite,assign) CGFloat cutEndTimeS
	功能： • 裁剪时长的结束时间; • 仅对视频有效
28	-(void)addKeyFrameAtCompTime:(CGFloat)compTimeS
	功能： • 添加关键帧
29	-(void)removeKeyFrameWithCompTime:(CGFloat)compTimeS
	功能： • 删除关键帧
30	-(void)removeAllKeyframe
	功能： • 删除所有关键帧
31	@property (readwrite, assign) CGFloat brightnessPercent

	功能： • 调节亮度的百分比. • 这里的百分比是两倍的关系,范围是 0--2.0; 其中 1.0 是默认值, • 0---1.0 是调小; • 1.0 是默认, • 1.0---2.0 是调大; • 如要关闭,则调整为 1.0.默认是关闭
32	@property (readwrite, assign) CGFloat contrastFilterPercent
	功能： • 调节对比度的百分比. • 这里的百分比是两倍的关系,范围是 0--2.0; 其中 1.0 是默认值, • 0---1.0 是调小; • 1.0 是默认, • 1.0---2.0 是调大; • 如要关闭,则调整为 1.0.默认是关闭
33	@property (readwrite, assign) CGFloat saturationFilterPercent
	功能： • 调节饱和度的百分比. • 这里的百分比是两倍的关系,范围是 0--2.0; 其中 1.0 是默认值, • 0---1.0 是调小; • 1.0 是默认 • 1.0---2.0 是调大; • 如要关闭,则调整为 1.0.默认是关闭
34	@property (readwrite, assign) CGFloat sharpFilterPercent
	功能： • 调节锐化的百分比. • 这里的百分比是两倍的关系,范围是 0--2.0; 其中 1.0 是默认值, • 0---1.0 是调小; • 1.0 是默认, • 1.0---2.0 是调大; • 如要关闭,则调整为 1.0.默认是关闭;
35	@property (readwrite, assign) CGFloat whiteBalanceFilterPercent
	功能： • 调节色温的百分比. • 这里的百分比是两倍的关系,范围是 0--2.0; 其中 1.0 是默认值, • 0---1.0 是调小; • 1.0 是默认, • 1.0---2.0 是调大; • 如要关闭,则调整为 1.0.默认是关闭
36	@property (readwrite, assign) CGFloat hueFilterPercent

	功能： • 调节色调的百分比. • 这里的百分比是两倍的关系,范围是 0--2.0; 其中 1.0 是默认值, • 0---1.0 是调小; • 1.0 是默认, • 1.0---2.0 是调大; • 如要关闭,则调整为 1.0.默认是关闭
37	@property (readwrite, assign) CGFloat exposurePercent **功能：** • 调节曝光度的百分比. • 这里的百分比是两倍的关系,范围是 0--2.0; 其中 1.0 是默认值, • 0---1.0 是调小; • 1.0 是默认, • 1.0---2.0 是调大; • 如要关闭,则调整为 1.0.默认是关闭
38	@property(readwrite, nonatomic) CGRect mosaicRect1InView **功能：** • 马赛克区域: • xy 以左上角开始，左上角是 0.0,0.0; • 宽度是一个百分比，最大是 1.0; • 高度是 一个百分比. 最大是 1.0; • 你实时传递过来的宽高和坐标，除以当前图层的宽高,转换为百分比后设置过来, • 宽高坐标可以通过 getCurrentRectInView 获取到当前图层在 compositionView 中的位置; • 我们举例最大 4 个马赛克区域;如果你需要更多的马赛克区域，则用 LSOMosaicRectFilter 执行增加; • 预览时有效
39	@property (readwrite,nonatomic) CGFloat beautyLevel **功能：** • 设置美颜, • 范围是 0.0---1.0; 0.0 是关闭美颜; 默认是 0.0;
40	@property (nonatomic,nullable, copy)LanSongFilter *filter **功能：** • 设置一个滤镜，设置后，之前增加的滤镜将全面清空. • 类似滤镜一个一个的切换.新设置一个，会覆盖上一个滤镜. • 如果滤镜是无，或清除之前的滤镜，这里填 nil;
41	-(void)addFilter:(nullable LanSongFilter *)filter **功能：** • 增加一个滤镜, • 增加后，会在上一个滤镜的后面增加一个新的滤镜. • 是级联的效果 • 源图像--->滤镜 1--->滤镜 2--->滤镜 3--->移动旋转缩放透明遮罩处理--->与别的图层做混合;
42	-(void)removeFilter:(nullable LanSongFilter *)filter **功能：** • 删除一个滤镜
43	-(void)removeAllFilter

	功能： • 删除所有滤镜
44	@property (nonatomic, assign)CGFloat audioVolume
	功能： • 调节素材中的音频音量. • 如果素材有音量,则设置，如果没有音量,则无效; • 范围是 0.0---8.0; • 1.0 是原始音量；大于 1.0,是放大；小于 1.0 是减低，如果降低建议是 0.1; • 如果是 0.0 则无声,
45	@property (readwrite, assign) CGFloat volumeFadeOutDurationS
	功能： • 音量淡出时长设置/获取; • 默认是 0.0,无淡出效果; • 最大为当前有效时长的 1/3;
46	@property (readwrite, assign) CGFloat volumeFadeInDurationS
	功能： • 音量淡入时长设置/获取; • 默认是 0.0；无淡入效果; • 最大为当前有效时长的 1/3;
47	@property (nonatomic, assign)CGFloat videoSpeed
	功能： • 视频的播放速度; • 范围是 0.1---10.0 • 默认 1.0；正常播放; • 建议的设置参数是: • 变慢: 0.1, 0.2, 0.4, 0.6,0.8 • 变快: 2.0, 3.0, 4.0, 6.0,8.0; • 当前仅是画面变速，变速过程中暂时无声音;
48	-(void)setVideoReverseAsync:(BOOL)isReverse asyncHandler:(void (^)(CGFloat percent, BOOL finish))handler
	功能： • 给视频图层设置倒序 • 异步是因为：在视频第一次倒序的时候，有一个异步预处理的过程; • percent:预处理百分比: 0---1.0; • 当前暂时不支持声音倒序. 在倒序时，默认音量为 0; • 调用此方法,会先触发容器暂停; • 在异步线程执行此 block；请在一下代码里调用; dispatch_async(dispatch_get_main_queue(), ^{ });
49	- (void)cancelVideoReverse
	功能： • 取消正在执行的视频倒序功能
50	@property(nonatomic, readonly)BOOL isReverse

	功能： • 当前是否在倒序状态
51	- (void)getThumbnailAsyncWithHandler:(void (^)(UIImage *image, BOOL finish))handler
	功能： • 异步获取当前缩略图; • 当前是每秒钟获取一帧;，一帧宽高最大是 100x100; **参数：** • image 是获取到的每一张缩略图; • finish 是 是否获取完毕; • 获取的缩略图的高度是 192 像素,高度是 192,返回所有图片的宽度总和是 当前要显示时长乘以 192; • 比如当前图层显示时长是 6.2 秒；则返回的缩略图是 6 张 192x192 的图片和一张 38x192 (38=192*0.2);
52	- (void)getOriginalThumbnailAsyncwithCount:(int)count handler:(void (^)(UIImage *image, BOOL finish))handler
	功能： • 获取原视频的缩略图; • 从原视频的 第 0 秒,到最后；返回的所有图片都是 192x192 像素大小; **参数：** • count 要获取的张数 ，如果不清楚获取几张图片，则用(int)originalDurationS; • handler 获取的异步回调;
53	- (void)getThumbnailAsyncWithDuration:(CGFloat)durationS handler:(void (^)(UIImage *image, BOOL finish))handler
	功能： • 获取指定时长的缩略图; • 从当前裁剪的开始时间,计算，每秒钟返回一帧，一帧的宽高是 192 **参数：** • durationS 要的时长 • handler 异步回调;
54	@property (readonly,assign) CGFloat thumbnailDisplayTimeS
	功能： • 缩略图的显示时间
55	@property(nonatomic, readwrite) NSMutableArray<UIImage *> *thumbImageArray
	功能： • 在你第一次调用过 getThumbnailAsyncWithHandler 后. 内部会保存到这个属性中. • 在下次获取的时候，则可以直接读取;
56	@property (nonatomic, assign)BOOL touchEnable
	功能： • 当前图层是否需要 touch 事件; • 默认是需要的;
57	@property (nonatomic, assign) CGFloat backGroundBlurLevel

	功能： • 背景虚化值. • 设置为 0,取消背景;
58	- (void)resetLayerLayout
	功能： • 把图层恢复到刚增加时的位置和大小
59	-(BOOL)isDisplay
	功能： • 当前图层是否在显示状态
60	- (LSOAnimation *) addAnimationWithJsonPath:(NSString *)jsonPath atCompS:(CGFloat) atCompS
	功能： • 用 AE 的 json 形式增加一个动画 **参数：** • jsonPath 动画文件 • atCompS 从播放器的什么位置增加
61	- (LSOAnimation *) setAnimationAtLayerHead:(NSString *)jsonPath
	功能： • 在图层的头部增加一个动画 **参数：** • jsonPath 动画文件,json 格式
62	- (LSOAnimation *) setAnimationAtLayerEnd:(NSString *)jsonPath
	功能： • 在图层的尾部增加一个动画 **参数：** • jsonPath 动画文件, json 格式
63	- (void)removeAnimation:(LSOAnimation *)animation
	功能： • 删除一个动画 **参数：** • animation 动画对象,在 addAnimation 的时候增加的
64	- (void)removeAllAnimationArray
	功能： • 删除所有动画
65	- (NSMutableArray *)getAllAnimationArray
	功能： • 获取所有的动画对象数组
66	-(BOOL) playAnimation:(LSOAnimation *)animation

	功能： • 预览一个动画 **参数：** • animation 预览会从动画的开始时间点,播放到结束时间点
67	- (LSOEffect *) addEffectWithJsonPath:(NSString *)jsonPath atCompS:(CGFloat) atCompS
	功能： • 用 AE 的 json 文件形式在指定时间点增加一个特效 **参数：** • jsonPath 特效的 json • atCompS 从容器的指定时间点
68	- (LSOEffect *) addEffectWithJsonAtLayerHead:(NSString *)jsonPath
	功能： • 在图层的头部增加一个特效 **参数：** • jsonPath 特效的 json 文件
69	- (LSOEffect *) addEffectWithJsonAtLayerEnd:(NSString *)jsonPath
	功能： • 在图层的尾部增加一个特效 **参数：** • jsonPath 特效的 json 文件
70	- (void)removeEffect:(LSOEffect *)effect
	功能： • 删除一个特效, **参数：** • effect 特效对象,从 addEffectXXX 得到的特效对象
71	- (void)removeAllEffectArray
	功能： • 删除所有的特效
72	-(BOOL) playEffect:(LSOEffect *)effect
	功能： • 预览一个特效 **参数：** • effect 预览会从特效的开始时间点,播放到结束时间点
73	- (NSMutableArray *)getAllEffectArray
	功能： • 获取所有的特效对象数组
74	@property(readwrite, assign)NSURL *transitionJsonUrl

	功能： • 设置转场的动画路径, json 格式; • 可通过这个获取是否设置了转场; 如果要取消转场;则这里等于 nil; • 设置后, 默认转场时间为 1 秒;
75	- (void)setMGTransitionWithColorUrl:(NSURL *)colorUrl maskUrl:(NSURL *)maskUrl
	功能： • 增加 mg 转场动画
76	@property(readwrite, assign)CGFloat transitionDurationS
	功能： • 设置或获取转场时间 • 在设置转场后有效; • 时间范围是 0---5.0 秒; • 如转场时间 大于图层时间, 则等于图层时间; • 可以通过转场时间,判断当前图层是否有转场功能;
77	@property(readonly, nonatomic)CGFloat transitionMaxDurationS
	功能： • 转场可设置的最大值; • 只能获取; • 最大值等于, 当前图层的 1/3 和下一个图层时长的 1/3 和 3 秒 的最小值;
78	@property(readonly, nonatomic)BOOL isAddTransition
	功能： • 是否增加了转场
79	- (BOOL)playTransition
	功能： • 预览转场; • 你需要在设置转场后调用才有效;
80	- (void)cancelTransition
	功能： • 取消转场
81	@property (nonatomic, readonly) CGFloat transitionStartTimeOfCompS
	功能： • 转场相对于播放器的开始时间
82	@property (readwrite,nonatomic)NSURL *videoURL
	功能： • 如果是拼接的是视频, 或叠加的是视频, 则可以获取到 videoURL 路径
83	@property (readwrite,nonatomic)UIImage *uiImage
	功能： • 拼接或叠加一张图片时, 增加的图片对象
84	@property (readonly, nonatomic, nullable) NSURL *reverseVideoUrl
	功能： • 获取倒序的视频路径; • 在设置倒序,并倒序完成后获取; • 在容器释放或 图层释放后, 内部会删除;

四、AE 模版 SDK

4.1、模板素材类： LSOAexAeModule 和 LSOAexImage

LSOAexModule 类是用来设置一个模板的各种素材，比如一个模板在导出时会导出背景视频文件，透明 mv 视频，声音等其他素材，加载后你会得到这个模板的宽度，高度， 可替换图片数量，以及每张图片的开始时间和 图片显示时长。

lsoaeximage 就是图片类，我们已经在解析时，对每张图片按照时间出现的先后顺序做了排序，你获取的第 1 个索引，就是最先显示的图片，你要替换视频图片为视频或图片则调用 updapath，你可以在开始时间调用，也可以在模板加入到预览播放器中调用； 替换时有一个 option 选项，当前支持设置缩放形式和开始时间，option 的枚举缩放，支持裁剪到全屏、完整缩放， 视频类型缩放；updatePath 替换返回布尔类型，表示替换成功或失败。 可多次调用。 如果是视频则可以通过 getThumbnailAsync 异步获取需要的缩略图。

4.2、AE 播放类：LSOAexplayerView

用来播放一个 AE 模板（LSOAexModule）， 支持播放进度回调、播放完成回调，导出进度回调，导出完毕回调、定位到时间，定位到 AexImage， 暂停 pause、恢复 resume，开始播放，开始导出， 有执行错误回调， 播放的 AexImage 改变回调，并支持手势，当点击播放画面时，会返回用户当前点击了哪个 LSOAexImage。当用户在播放过程中要替换素材时， 可通过 AexImage 的 updatePath 方法替换， iOS 需要在调用 updateWithUrl 后另外再执行 aexComposition 的 updateAexImage；我们内部检查到你替换后，会显示替换后的画面，并渲染到当前 module 中。

iOS 版本的 API 是：LSOAexComposition 和 LSOAexDisplayView 两个类组成， 一个是合成，一个是播放窗口。

4.3、手势说明：

在播放中，当用户手指点击画面时，播放器会暂停，并返回给你用户选中的 AexImage 对象，可直接更改 AexImage 中的视频或图片，更改后，播放器会相应的播放替换后的视频或图片。 当用户图片或视频宽高不等于 AE 模板中的图片宽高时， 在替换时，默认会根据 option 的缩放类型做缩放，并从画面中心显示； 在用户点击播放画面时， 可单指移动调整要显示画面的哪部分，也可以双指缩放和旋转来调整显示的画面， 调整后的画面就是最终要生成的画面。如果你两个图片在 AE 制作时是重叠的，有可能点击图片的时候，会选中另一张图片,而非你要的图片,建议在制作时不要重叠（下一个版本会做进一步的优化）。

4.4、AexImage 的改变回调方法说明:

我们内部是查找哪个 aexImage 的开始时间点 和当前时间的差值最小，而返回哪个 aexImage，如果你多个图片在时间点上是重叠的，有可能返回的 aexImage 不是你要的结果，这是正常情况，可和 AE 设计师商量如何避免

五、人像分割 SDK

5.1、IOS 的人像分割用到的库

库：
- tnn.framework
- TenLineSegmnetFramework.framework
- 说明: 拖动到 Build Phases 的 Link Binary 中.

模型：
- 模型文件:TEN_LINE_IOS.MODEL
- 协议文件: TEN_LINE_IOS.MODEOPROTO
- TNN : tnn.metallib
- 说明: 拖动到工程里,在 Build Phases 中可以找到即可.

需要依赖系统的库：
- Accelerate.framework (重要)
- CoreML.framework (重要)
- Metal.framework (重要)
- CoreMedia.framework
- CoreGraphics.framework
- libc++.tbd
- UIKit.framework
- AVFoundation.framework
- Foundation.framework
- 说明:以上库加入一次即可.、

调用
- **三个方法：**

唯一类:TLAPISegment
/// 初始化模型
/// @param modelPath model 的路径
/// @param protoPath 协议的路径

```
- (instancetype)initWithModelPath:(NSString *)modelPath protoPath:(NSString *)protoPath;


/// 分割一帧画面
/// @param image_buffer 原始的数据,
/// @param callback 分割完毕后的回调, 回调工作在 UI 线程;
- (void)segmentPixelBuffer:(CVPixelBufferRef)image_buffer callback:(segmentCallback2)callback;


/// 释放模型
- (void)relaseLSO;
```

5.2、不用人像分割，可删除的文件

Android：

1. assets 下的 tenlinev2.model
2. jniLibs/arm64-v8a 下的 libTenLineJni.so
3. jniLibs/armeabi-v7a 下的 libTenLineJni.so

IOS：

1. tnn.framework
2. TenLineSegmnetFramework.framework
3. tnn.metallib
4. TEN_LINE_IOS.MODEL
5. TEN_LINE_IOS.MODEOPROTO
6. TenLineFast_ios.MODEL
7. TenLineFast_ios.MODELPROTO

六、联系我们

网站：http://www.lansongtech.com/
eMail：support@lansongtech.com
手机：18006716739
QQ：1852600324
地址：杭州市 余杭区 文一西路 1324 号利尔达物联网科技园 6 号楼 2001