iOS 视频 iOS 开发 信息技术（IT） h264 修改

iOS 系统中，H.264 视频流可以硬件解码吗？ 具体如何实现？ 修改

写补充说明

添加评论 分享 • 邀请回答 举报

13 个回答 按投票排序

姚冬，程序员

109

wzw0114、尚沛、张艺麟 等人赞同

这个问题都问了两年多了，没有很好的回答，我最近正好搞定了iOS的硬解码 H.264，借这个问题来分享下经验。

其实至少从iPhone4开始，苹果就是支持硬件解码了，但是硬解码API一直是私有API，不开放给开发者使用，只有越狱才能使用，正常的App如果想提交到AppStore是不允许使用私有API的。

从iOS8开始，可能是苹果想通了，开放了硬解码和硬编码API，就是名为 VideoToolbox.framework的API，需要用iOS 8以后才能使用，iOS 7.x上还不行。

这套硬解码API是几个纯C函数，在任何OC或者 C++代码里都可以使用。

首先要把 VideoToolbox.framework 添加到工程里，并且包含以下头文件。
#include <VideoToolbox/VideoToolbox.h>

解码主要需要以下三个函数
VTDecompressionSessionCreate 创建解码 session
VTDecompressionSessionDecodeFrame 解码一个frame
VTDecompressionSessionInvalidate 销毁解码 session

首先要创建 decode session，方法如下：

```
OSStatus status = VTDecompressionSessionCreate(kCFAllocatorDefault,
                                               decoderFormatDescription,
                                               NULL, attrs,
                                               &callBackRecord,
                                               &deocderSession);
```

其中 decoderFormatDescription 是 CMVideoFormatDescriptionRef 类型的视频格式描述，这个需要用H.264的 sps 和 pps数据来创建，调用以下函数创建 decoderFormatDescription
CMVideoFormatDescriptionCreateFromH264ParameterSets

需要注意的是，这里用的 sps和pps数据是不包含“00 00 00 01”的start code的。

attr是传递给decode session的属性词典

```
        CFDictionaryRef attrs = NULL;
        const void *keys[] = { kCVPixelBufferPixelFormatTypeKey };
//      kCVPixelFormatType_420YpCbCr8Planar is YUV420
//      kCVPixelFormatType_420YpCbCr8BiPlanarFullRange is NV12
        uint32_t v = kCVPixelFormatType_420YpCbCr8BiPlanarFullRange;
        const void *values[] = { CFNumberCreate(NULL, kCFNumberSInt32Type, &v) };
        attrs = CFDictionaryCreate(NULL, keys, values, 1, NULL, NULL);
```

其中重要的属性就一个，kCVPixelBufferPixelFormatTypeKey，指定解码后的图像格式，必须指定成NV12，苹果的硬解码器只支持NV12。

callBackRecord 是用来指定回调函数的，解码器支持异步模式，解码后会调用这里的回调函数。

如果 decoderSession创建成功就可以开始解码了。

```
            VTDecodeFrameFlags flags = 0;
            //kVTDecodeFrame_EnableTemporalProcessing | kVTDecodeFrame_EnableAsynchr
            VTDecodeInfoFlags flagOut = 0;
            CVPixelBufferRef outputPixelBuffer = NULL;
            OSStatus decodeStatus = VTDecompressionSessionDecodeFrame(deocderSession
                                                                      sampleBuffer,
                                                                      flags,
                                                                      &outputPixelBu
                                                                      &flagOut);
```

其中 flags 用0 表示使用同步解码，这样比较简单。
其中 sampleBuffer是输入的H.264视频数据，每次输入一个frame。
先用CMBlockBufferCreateWithMemoryBlock 从H.264数据创建一个CMBlockBufferRef实例。
然后用 CMSampleBufferCreateReady创建CMSampleBufferRef实例。
这里要注意的是，传入的H.264数据需要Mp4风格的，就是开始的四个字节是数据的长度而不是“00 00 00 01”的start code，四个字节的长度是big-endian的。
一般来说从 视频里读出的数据都是 “00 00 00 01”开头的，这里需要自己转换下。

解码成功之后，outputPixelBuffer里就是一帧 NV12格式的YUV图像了。
如果想获取YUV的数据可以通过

```
    CVPixelBufferLockBaseAddress(outputPixelBuffer, 0);
    void *baseAddress = CVPixelBufferGetBaseAddress(outputPixelBuffer);
```

获得图像数据的指针，需要说明baseAddress并不是指向YUV数据，而是指向一个CVPlanarPixelBufferInfo_YCbCrBiPlanar结构体，结构体里记录了两个plane的offset和pitch。

但是如果想把视频播放出来是不需要去读取YUV数据的，因为CVPixelBufferRef是可以直接转换成OpenGL的Texture或者UIImage的。
调用CVOpenGLESTextureCacheCreateTextureFromImage，可以直接创建OpenGL Texture

从 CVPixelBufferRef 创建 UIImage

```
CIImage *ciImage = [CIImage imageWithCVPixelBuffer:pixelBuffer];
UIImage *uiImage = [UIImage imageWithCIImage:ciImage];
```

解码完成后销毁 decoder session
VTDecompressionSessionInvalidate(deocderSession)

硬解码的基本流程就是这样了，如果需要成功解码播放视频还需要一些H.264视频格式，YUV图像格式，OpenGL等基础知识。

还是有很多小细节要处理的，无法在这里一一说明了，有人有问题可以在评论里讨论。
从解码到播放，大约1000行代码左右，主要是OpenGL渲染的代码比较多。

苹果官方的示例代码：
WWDC - Apple Developer

苹果的例子下载链接实效了，我也找不到那个例子，我自己写了一个。
stevenyao/iOSHardwareDecoder · GitHub

编辑于 2015-08-17　　19 条评论 • 作者保留权利

7

bbcallen，ijkplayer

Wayne Wang、人人通过、跳Tiao 等人赞同

VideoToolbox 本身文档里的 demo 够用了，但如果希望与 FFmpeg 甚至其他第三方 demuxer 集成，目前我所知道可以拿来直接用的有这么些：

1. 上面说的VLC，不过 VideoToolbox 相关 patch 在 review 过程中遇到很多阻碍，一年过去了，还没有 merge 进 master。自己 merge 一下，肯定能用，不过因为VLC自身太过插件化了，优化起来很蛋疼。

2. XBMC（现在改名叫kodi），我能找到的最早完整实现，不过使用了部分私有API，上架可能有麻烦，使用时注意修改。

3. ijkplayer，现已随哔哩哔哩 iOS 客户端加入AppStore豪华午餐。
相关的主要逻辑在 ijkplayer/ios/IJKMediaPlayer/IJKMediaPlayer/ijkmedia/ijkplayer/ios/pipeline at master · Bilibili/ijkplayer · GitHub

4. Vitamio 貌似也支持 iOS 硬解，不过我没有确认是否使用了 VideoToolbox.

编辑于 2015-05-21　　7 条评论 • 作者保留权利

0

匿名用户

iOS 支持 H.264 視頻流硬解碼：

HTTP Live Streaming Overview: Frequently Asked Questions | developer.apple.com/lib...