



灰 信 网 (软件开发博客聚合)

程序员专属的优秀博客文章阅读平台

相关文章

RTP载荷H264解包过程分析，FFMPEG解码QT展示
将H264码流打包成RTP包
RTP协议解析和H264码流提取
RTP打包与发送H.264实时码流
RTP-H264码流
WIRESHARK 提取H264码流
H264/H265码流类型
SPRINGBOOT开发环境安装(JDK+MAVEN+INTELLIJ IDEA)
基于 OPENCV3.3 ANDROID 人脸检测
CSS瀑布流布局



热门文章

JDBC自建工具类
数据增强
我的一个博客文章
CF - C. PLASTICINE ZEBRA
MATPLOTLIB.PYPLOT画子图的两种方式(面向对象和PYPLOT)
DOCKER-COMPOSE搭建REDIS主从复制
SPRINGCLOUD与CONSUL整合实现负载均衡时不生效的问题
易语言大漠插件模块制作后台找字FINDSTREX和FINDSTRFASTEXTS
RETROFIT--请求方法那些事儿
ANDROID全局异常捕获

推荐文章

DUBBO+ZOOKEEPER示例代码_SPRING SPRINGBOOT集成DUBBO
ANDROID 进程模块获取
PYTHON-元组AND字典

RTP(载荷H264码流)解包与封包

标签: 音视频开发

一、H264介绍

1.1 H264概述

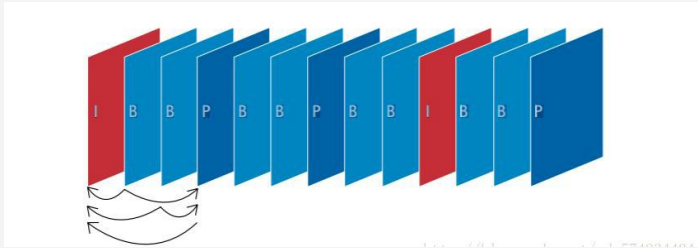
h264是一种视频压缩标准。

经过压缩后的帧分为：帧，P帧和B帧：

- I帧：关键帧，采用帧内压缩技术。（自身可以通过视频解压缩算法解压成一张单独的完整的图片）
- P帧：向前参考帧，在压缩时，只参考前面已经处理的帧（只需要参考前面的I帧或P帧）。采用帧首压缩技术。
- B帧：双向参考帧，在压缩时，它即参考前面的帧，又参考它后面的帧（需要同时参考前面和后面的I帧或P帧）。采用帧间压缩技术。

除了I/P/B帧外，还有图像序列GOP。

- GOP：两个I帧之间是一个图像序列，在一个图像序列中只有一个I帧



(图出自H.264视频压缩标准白皮书)

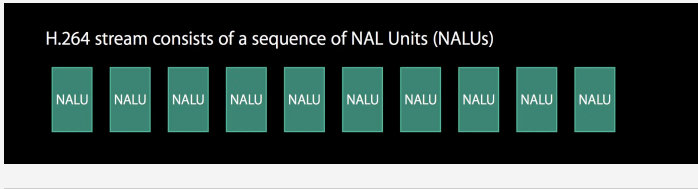
在H.264基准类中，仅使用帧和P帧以实现低延时，因此是网络摄像机和视频编码器的理想选择。

1.2 H264原始码流结构

H264功能分为两层，VCL(视频编码层)和NAL(网络提取层)

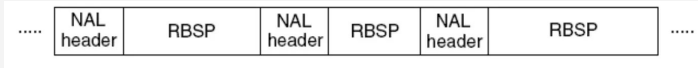
- VCL：包括核心压缩引擎和块，宏块和片的语法级别定义，设计目标是尽可能地独立于网络进行高效的编码。
- NAL：负责将VCL产生的比特字符串适配到各种各样的网络和多元环境中，覆盖了所有片级以上的语法级别。

VCL数据传输或者存储之前，会被映射到一个NALU中，H264数据包含一个个NALU。如下图：



一个NALU = 一组对应于视频编码的NALU头部信息 + 一个原始字节序列负荷(RBSP, Raw Byte Sequence Payload)

NAL单元格式如下图：



一个原始的NALU单元结构包含：
[StartCode]+[NALU Header]+[NALU Payload] 三部分（StartCode，是一个NALU单元开始，必须是00 00 00 01 或者00 00 01）

H.264的编码视频序列包括一系列的NAL单元，每个NAL单元包含一个RBSP。编码片（包括数据分割片IDR片）和序列RBSP结束符被定义为VCL NAL单元，其余为NAL单元。典型的RBSP单元序列如图2所示。每个单元都按独立的NAL单元传送。单元的信息头（一个字节）定义了RBSP单元的类型，NAL单元的其余部分为RBSP数据。



1.3 NAL单元

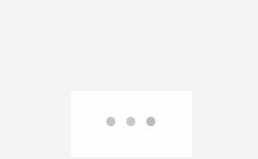
每个NAL单元是一个一定语法元素的可变长字节字符串，包括包含一个字节的头信息（用来表示数据类型），以及若干整数字节的负荷数据。一个NAL单元可以携带一个编码片、A/B/C型数据分割或一个序列或图像参数集。

NALU头由一个字节组成，它的语法如下：



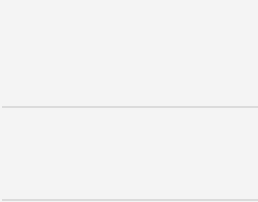
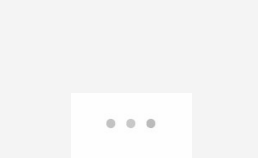
NAL单元按RTP**按序传送。其中，T为负荷数据类型，占5bit；R为重要性指示位，占2个bit；最后的F为禁止位，占1bit。具体如下：

学习 笔记 之 ANDROID 篇——SERVICE (2.0)
C++ 之 STRING 类
ANDROID STUDIO：通过 ARTIFACTORY 搭建本地 仓库 优化 编译 速度
转, JAVA H5 复杂 TABLE 导出 EXCEL 并 下载
WDATEPICKER 时间 插件
基于 GEC6818 智能家居 的实现 -- 点亮 LED 灯
【LEETCODE 每日一题】[困难]57. 插入区间



相关标签

C++
QT
算法
RTP 流媒体
编解码
音视频开发
H.264
H.265
AVCC
HVCC



COPYRIGHT © 2010-2022 - ALL RIGHTS RESERVED - WWW.FREESION.COM

- NALU类型位: 可以表示NALU的32种不同类型特征，类型1~12是H.264定义的，类型24~31是用于H.264以外的，RTP负荷规范使用这其中的一些值来定义包聚合和分裂，其他值为H.264保留。
- 重要性指示位: 用于在重构过程中标记一个NAL单元的重要性，值越大，越重要。值为0表示这个NAL单元没有用于预测，因此可被解码器抛弃而不会有错误扩散；值高于0表示此NAL单元要用于无漂移重构，且值越高，对此NAL单元丢失的影响越大。
- 禁止位: 编码中默认值为0，当网络识别此单元中存在比特错误时，可将其设为1，以便接收方去掉该单元，主要 用于适应不同种类的网络环境（比如有线无线相结合的环境）。

264常见的帧头数据为：

00 00 00 01 **67** (SPS)

00 00 00 01 **68** (PPS)

00 00 00 01 **65** (IDR 帧)

00 00 00 01 **61** (P帧)

上述的**67,68,65,61**，还有41等，都是该NALU的识别级别。

F：禁止为，0表示正常，1表示错误，一般都是0

NRI：重要级别，11表示非常重要。

TYPE：表示该NALU的类型是什么，

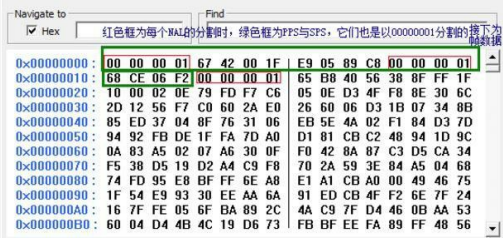
见下表，由此可知7为序列参数集(SPS)，8为图像参数集(PPS)，5代表帧。1代表非帧。

由此可知，61和41其实都是P帧（type值为1），只是重要级别不一样（它们的NRI一个是11BIN，一个是10BIN）

NALU类型是我们判断帧类型的利器，从官方文档中得出如下图：

nal_unit_type	NAL类型	
0	未使用	
1	不分区、非 IDR 图像的片	2, 3, 4
2	片分区 A	2
3	片分区 B	3
4	片分区 C	4
5	IDR 图像中的片	2, 3
6	补充增强信息单元（SEI）	5
7	序列参数集	0
8	图像参数集	1
9	分界符	6
10	序列结束	7
11	码流结束	8
12	填充	9
13..23	保留	
24..31	未使用	

1.4 H264 帧判断



最上面图的码流对应的数据来层层分析，以00 00 00 01分割之后的下一个字节就是NALU类型，将其转为二进制数据后，

解读顺序为从左往右算，如下：

- (1) 第1位禁止位，值为1表示语法出错
- (2) 第2~3位为参考级别
- (3) 第4~8为是nal单元类型

例如上面00000001后有67,68以及65

其中0x67的二进制码为：

0110 0111

4-8为00111，转为十进制7，参考第二幅图：7对应序列参数集SPS

其中0x68的二进制码为：

0110 1000

4-8为01000，转为十进制8，参考第二幅图：8对应图像参数集PPS

其中0x65的二进制码为：

011 00101

4-8位为00101，转为十进制5，参考第二幅图：5对应IDR图像中的片(I帧)

所以判断是否为帧的算法为：

(NALU类型 & 0001 1111) = 5 即 (NALU类型 & 31) = 5
比如0x65 & 31 = 5

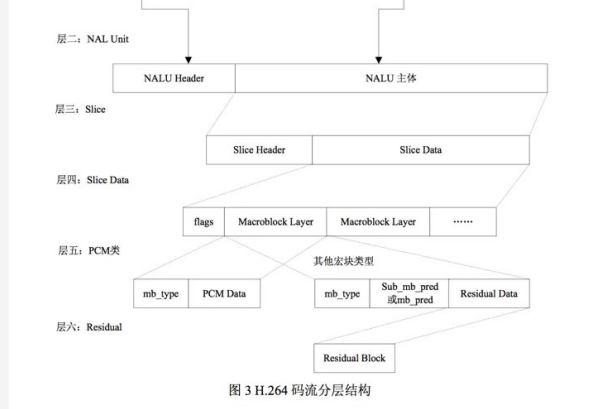
即：int value = buf[4] & 0x0f; // 5是帧, 7是sps 8是pps

二、RTP打包发送H264之封包详解

RFC3984是H.264的baseline码流在RTP方式下传输的规范，这里只讨论FU-A分包方式，

H264的码流结构





1、单个NAL包单元

12字节的RTP头后面的就是音视频数据，比较简单。一个封装单个NAL单元包到RTP的NAL单元流的RTP序号必须符合NAL单元的解码顺序。

对于 NALU 的长度小于 MTU 大小的包，一般采用单一 NAL 单元模式。

对于一个原始的 H.264 NALU 单元常由[Start Code][NALU Header][NALU Payload]三部分组成, 其中 Start Code 用于标示这是一个

NALU 单元的开始, 必须是“00 00 00 01”或“00 00 01”, NALU 头仅一个字节, 其后都是 NALU 单元内容。

打包时去除“00 00 01”或“00 00 00 01”的开始码, 把其他数据封包的 RTP包即可

```

1 | 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
2 | 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
3 | +-----+
4 | F|NRI| type |
5 | +-----+
6 | |
7 | | Bytes 2..n of a Single NAL unit |
8 | |
9 | | +-----+
10 | | :...OPTIONAL RTP padding |
11 | +-----+
12 |

```

如有一个 H.264 的 NALU 是这样的：

[00 00 00 01 67 42 A0 1E 23 56 0E 2F ...]

这是一个序列参数集 NAL 单元, [00 00 00 01] 是四个字节的开始码, 67 是 NALU 头, 42 开始的数据是 NALU 内容。

封装成 RTP 包将如下:

[RTP Header] [67 42 A0 1E 23 56 0E 2F]

即只要去掉 4 个字节的开始码就可以了

2. 组合封包模式

当 NALU 的长度特别小时, 可以把几个 NALU 单元封在一个 RTP 包中

3、FU-A 的分片格式

数据比较大的H264视频包, 被RTP分片发送。12字节的RTP头后面跟随的就是FU-A分片：

而当 NALU 的长度超过 MTU 时, 就必须对 NALU 单元进行分片封包, 也称为 Fragmentation Units (FUs)

```

1 | 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
2 | 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
3 | +-----+
4 | FU indicator | FU header |
5 | +-----+
6 | |
7 | | FU payload |
8 | |
9 | | +-----+
10 | | :...OPTIONAL RTP padding |
11 | +-----+
12 | Figure 14. RTP payload format for FU-A

```

1) FU indicator有以下格式:

```

1 | +-----+
2 | 0|1|2|3|4|5|6|7|
3 | +-----+
4 | F|NRI| Type |
5 | +-----+
6 |

```

FU指示字节的类型域的28, 29表示FU-A和FU-B, NRI域的值必须根据分片NAL单元的李域的值设置。（此处Type就是rtp分片类型）见下表

.Type	Packet	Type name
0	undefined	-
1-23	NAL unit	Single NAL unit packet per H.264
24	STAP-A	Single-time aggregation packet
25	STAP-B	Single-time aggregation packet
26	MTAP16	Multi-time aggregation packet
27	MTAP24	Multi-time aggregation packet
28	FU-A	Fragmentation unit
29	FU-B	Fragmentation unit
30-31	undefined	

2) FU header的格式如下:

```

1 | +-----+
2 | 0|1|2|3|4|5|6|7|
3 | +-----+
4 | S|E|R| Type |
5 | +-----+

```

S: 1 bit 当设置成1,开始位指示分片NAL单元的开始。当跟随的FU荷载不是分片NAL单元荷载的开始, 开始位设为0。

E: 1 bit 当设置成1,结束位指示分片NAL单元的结束, 即, 荷载的最后字节也是分片NAL单元的最后一个字。当跟随的FU荷载不是分片NAL单元的最后分片,结束位设置为0。

R: 1 bit
保留位必须设置为0, 接收者必须忽略该位。

Type: 5 bits
此处的Type就是NALU头中的Type,取1-23的那个值, 表示 NAL单元荷载类型定义, 见图4

4、拆包和解包

拆包: 当编码器在编码时需要将原有一个NAL按照FU-A进行分片, 原有的NAL的单元头与分片后的FU-A的单元头有如下关系:

原始的NAL头的前三位为FU indicator的前三位, 原始的NAL头的后五位为FU header的后五位,

FU indicator与FU header的剩余位数根据实际情况决定。

解包: 当接收端收到FU-A的分片数据, 需要将所有的分片包组合还原成原始的NAL包时, FU-A的单元头与还原后的NAL的关系如下:

还原后的NAL头的\位是由FU indicator的前三位加FU header的后五位组成, 即:

nal_unit_type = (fu_indicator & 0xe0) | (fu_header & 0x1f)

参考:

<https://blog.csdn.net/machh/article/details/52165292>

<https://www.jianshu.com/p/8edb448cf22e>

版权声明: 本文为qq_15559817原创文章, 遵循 CC 4.0 BY-SA 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_15559817/article/details/106135451



智能推荐



CRC 详解

CRC-知识解析 cyclic redundancy check 写在前面的话: 之前在做学校项目的时候用到了CRC 原理, 但在网上查找的过程中, 发现讲解CRC知识的资源很多, 但是对新手比较友好的、讲的十分清楚的又很少, 很多博主也不求甚解, 弄得读起来心中常常不由自主地奔腾过上千个“为什么”“为什么”, 本文是我在阅读了许多资料的基础上整理、...



JAVA基础（异常概述）

1. 什么是异常【1】异常的概述 异常就是Java程序在运行过程中出现的错误。【2】异常的分类 Throwable Throwable 类是 Java 语言中所有错误或异常的超类, 两个子类的实例, Error 和 Exception, 通常用于指示发生了异常情况。Error Error 是 Throwable 的子类, 用于指示合理的应用程序不应该试图捕获的严重问题。服务器宕机,数据库崩...



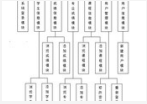
【PTA】凯撒密码（20 分）

注意输入的n可能是-10000之类的数字, 要善用%。直接对大小写进行ASCII码的判断即可, 没必要打表。...



使用KUBEADM安装KUBERNETES1.10.1【CENTOS7.3离线安装DOCKER, KUBEADM, KUBECTL, KUBELET, DASHBOARD】KUBERNETESV1.10.1

参考资料: <https://kubernetes.io/docs/tasks/tools/install-kubeadm/>
<https://blog.csdn.net/yitaidn/article/details/79937316> <https://www.jianshu.com/p/9c7e1c957752>
<https://www.jianshu.com/p/3ec8945a864f> 环境:3台...



基于MVC模式的研究生信息管理系统

一、前言 本系统开发语言为C#, 数据库为Access2010, 开发环境为VS2010, 模式MVC。主要功能为系统管理员添加专业信息、开设的课程信息和系统用户信息, 对用户进行权限设置并对其进行维护;普通管理员录入研究生的基本信息, 为学生添加学号和默认密码;任课教师对研究生的成绩进行录入;研究生根据学号进行个人信息查询和成绩查询。二、系统模块划分 三、数据库设计 用户信息"userinf...



猜你喜欢



图像增强库ALBUMENTATIONS（一）

1.大体的认识、有用没用的bb 官网:<https://albumentations.ai/> github: <https://github.com/albumentations-team/albumentations> 例子:https://github.com/albumentations-team/albumentations_examples 特点: 分类、目标检测、分割等任务都支持增强, 与pyto...



SWITCH语句练习题

练习题一、写一个switch语句, 不生产大表也不生产小表, 贴出对应的反汇编 反汇编代码: 练习二、写一个switch语句, 只生成大表,贴出对应的反汇编 反汇编代码: 练习题三、写一个switch语句, 生成大表和小表,贴出对应的反汇编 反汇编代码: switch语句总结 编译器会根据相应的条件生成多种代码1.2.3. 1、当条件数量不多或者条件差值太乱时就会生成第一种代码判断跳转判断条件的代码。2...



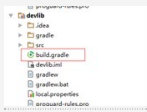
.NET STANDARD 来日苦短去日长

作者: Richard 翻译: 精致码农-王亮 原文: <http://dwz.win/Q4h> 自从 .NET 5 宣贯以来, 很多人都在问这对 .NET Standard 意味着什么, 它是否仍然重要。在这篇文章中, 我将解释 .NET 5 是如何改进代码共用并取代 .NET Standard 的, 我还将介绍什么情况下你仍然需要 .NET Standard。概要 .NET 5 将是一个具有统一功能和 API...



JAVA集合:ARRAYLIST (JDK1.8 源码解读)

ArrayList 概述 重要方法分析 add 构造 ge...



GRADLE学习 笔记, 常用命令, 多渠道打包等

本文整理自: <http://stormzhang.com/devtools/2014/12/18/android-studio-tutorial4/>
<http://stormzhang.com/devtools/2015/01/05/android-studio-tutorial5/>
<http://stormzhang.com/devtools/2015/01/15/android-studio-...>