

网易云课堂《安卓高级开发工程师》班级资料

# 音频播放与音视频同步

## 云信拉流端(二)

网易云课堂《安卓高级开发工程师》班级资料

# 课程目标

学习通过底层操作系统服务(OpenSL ES学习)

手写实现音视频同步

# 什么是OpenES SL

## ■ 音视频播放

OpenSL ES 全称是：Open Sound Library for Embedded Systems，是一套无授权费、跨平台、针对嵌入式系统精心优化的硬件音频加速API。它为移动多媒体设备上的本地应用程序开发者提供标准化, 高性能, 低响应时间的音频功能实现方法，并实现软硬件音频性能的直接跨平台部署，降低执行难度，促进高级音频市场的发展

■ 一句话概述：OpenSL ES 是一套针对移动式平台的音频标准。

# 播放音频Android支持哪些

01

MediaPlayer

02

AudioTrack

03

OpenES SL

# OpenSL ES 播放的优势

## ■ 优势

- (1) C 语言接口，需要在 NDK 下开发，能更好地集成在 native 应用中
- (2) 运行于 native 层，播放 速度极快，延时低 对于音视频同步中以音频为准的最为适合不过
- (3) 减少java层频繁的反射调用，如果通过AudioTrack播放需要将解码后得pcm数据反射java 增加开销



# 如何学习底层直接调用Android服务

■ 地址 <https://github.com/googlesamples/android-ndk>

Android NDK samples with Android Studio <http://developer.android.com/ndk>

1,289 commits

5 branches

0 releases

48 contributors

Apache-2.0

Branch: master

New pull request

Find File

Clone or download

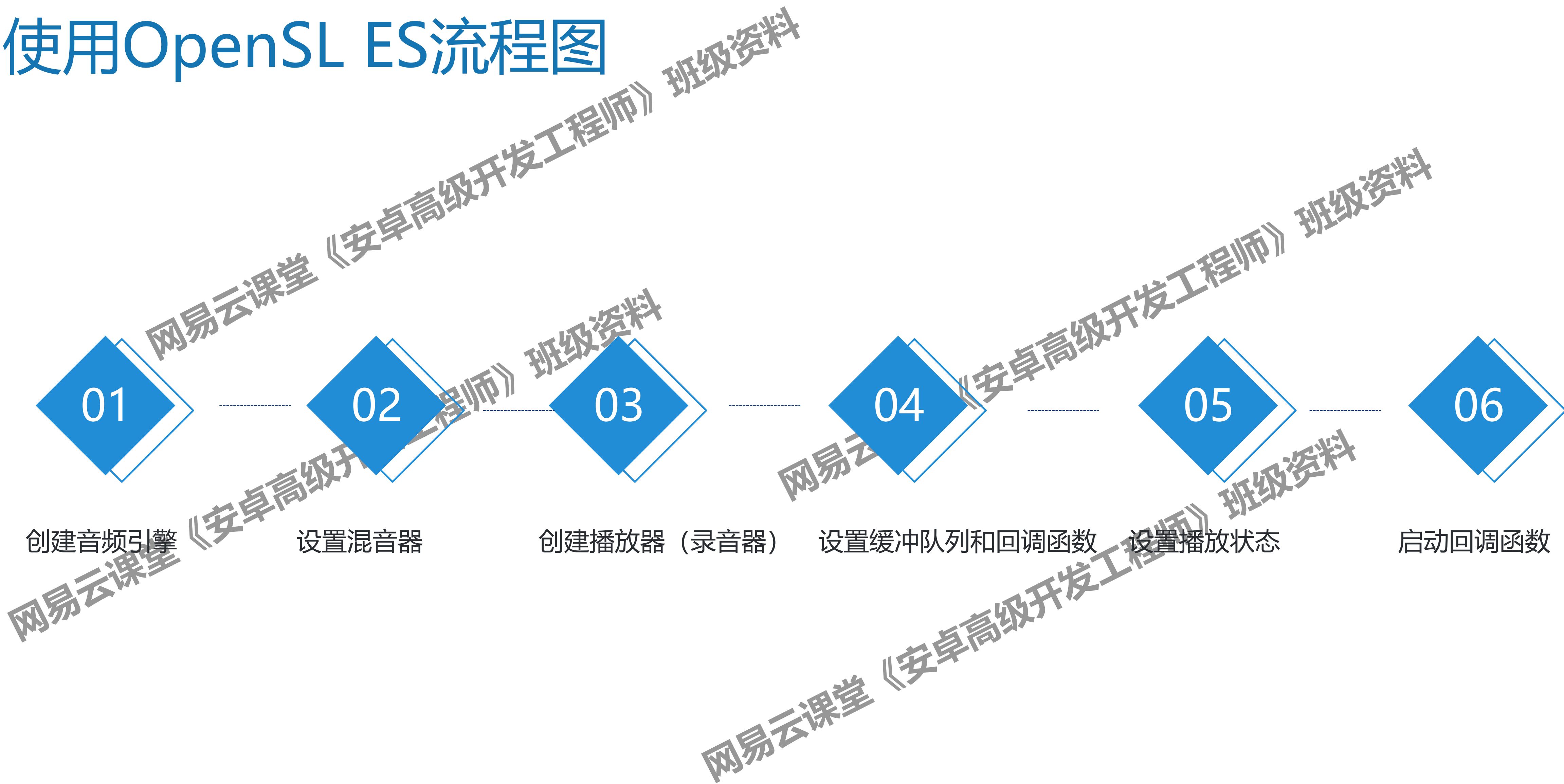
ggfan added more instruction to build native-midi with studio (#630)

Latest commit 86ceeb2 on 15 Mar

.ci_tools	CI script update: no need to install with constraintLayout 1.0.2+ (#618)	4 months ago
audio-echo	Update build script to version 3.3.1 (#622)	4 months ago
bitmap-plasma	Update build script to version 3.3.1 (#622)	4 months ago
builder	Update build script to version 3.3.1 (#622)	4 months ago
camera	Update build script to version 3.3.1 (#622)	4 months ago
display-p3	Update build script to version 3.3.1 (#622)	4 months ago
endless-tunnel	Update build script to version 3.3.1 (#622)	4 months ago
gles3jni	Update build script to version 3.3.1 (#622)	4 months ago
hello-cdep	Update build script to version 3.3.1 (#622)	4 months ago
hello-gl2	Update build script to version 3.3.1 (#622)	4 months ago
hello-jni	Update build script to version 3.3.1 (#622)	4 months ago
hello-jniCallback	Update build script to version 3.3.1 (#622)	4 months ago
hello-libc	Update build script to version 3.3.1 (#622)	4 months ago
hello-neon	Update build script to version 3.3.1 (#622)	4 months ago
kotlin-app	Update build script to version 3.3.1 (#622)	4 months ago
native-activity	Update build script to version 3.3.1 (#622)	4 months ago
native-audio	Update build script to version 3.3.1 (#622)	4 months ago
native-codec	Update build script to version 3.3.1 (#622)	4 months ago
native-media	Update build script to version 3.3.1 (#622)	4 months ago
native-midi	added more instruction to build native-midi with studio (#630)	3 months ago
native-plasma	Update build script to version 3.3.1 (#622)	4 months ago
nn_sample	Update build script to version 3.3.1 (#622)	4 months ago
other-builds	Update build script to version 3.3.1 (#622)	4 months ago
san-angeles	Update build script to version 3.3.1 (#622)	4 months ago
sensor-graph	Update build script to version 3.3.1 (#622)	4 months ago

- Camera 底层操作相机
- bitmap-plasma 底层直接渲染Bitmap
- native-activity native层操作Activity
- native-audio native层播放音频
- native-media native层控制多媒体
- native-plasma Bitmap中细胞质效果
- nn\_sample Android中网络库
- sensor-graph native层获取地图经纬度
- Webp native层加载和渲染webp

# 使用OpenSL ES流程图



# OpenGL ES

网易云课堂《安卓高级开发工程师》班级资料

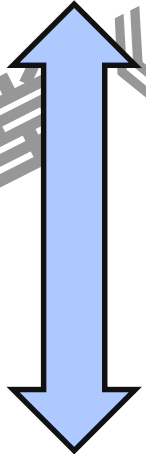
网易云课堂《安卓高级开发工程师》班级资料

网易云课堂《安卓高级开发工程师》班级资料

网易云课堂《安卓高级开发工程师》班级资料



# 云信课堂项目框架



JNI (调用native)  
JavaCallHelper(反射调用java)

# 控制层

- 作用:
- 初始化FFmpeg参数设置
  - 控制进度播放(开始播放, 暂定, 调节播放)
  - 从视频文件(视频流)解析Packet

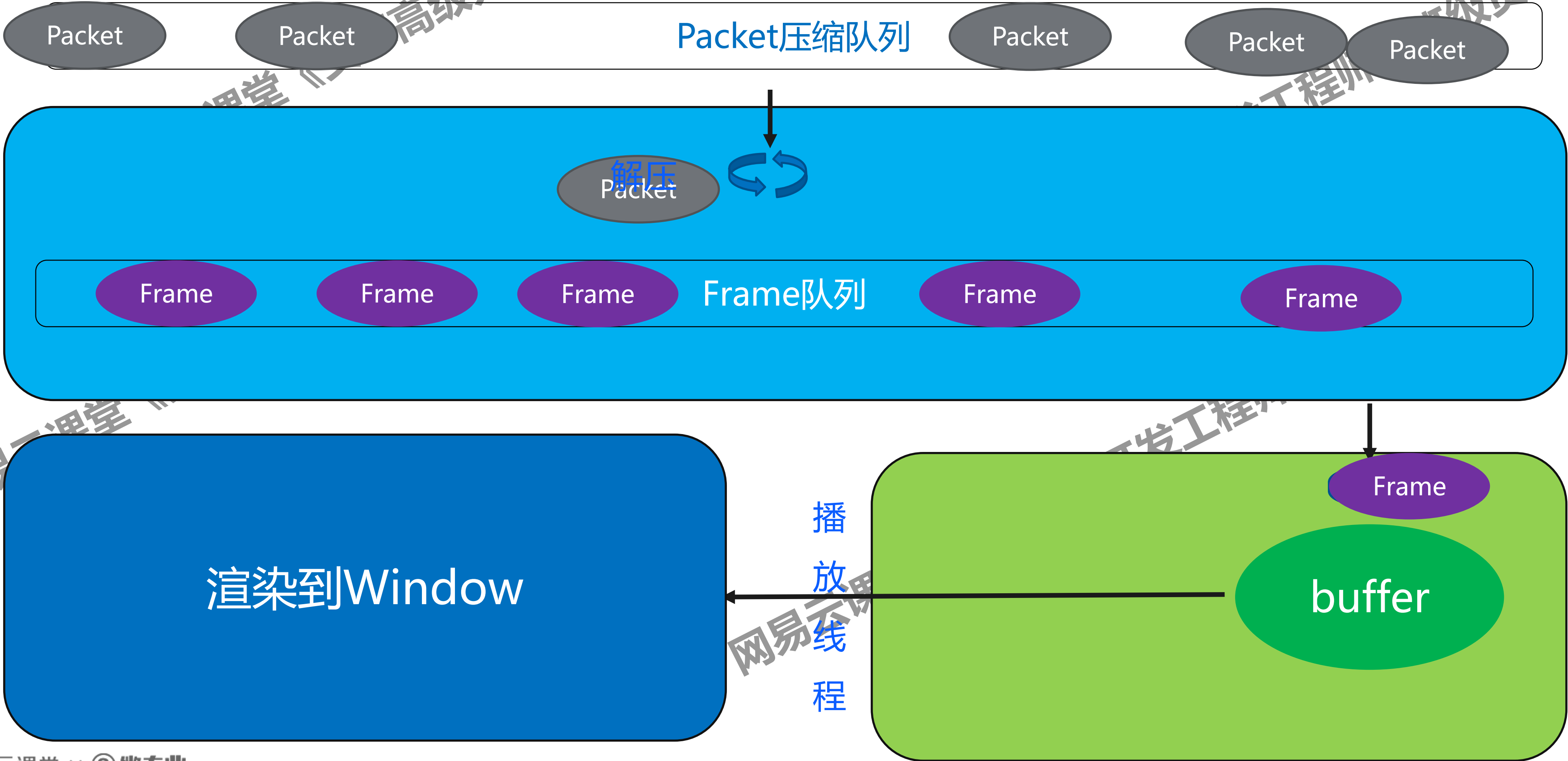


# 视频层

作用: 解码Packet 转换成Frame

解  
码  
线  
程

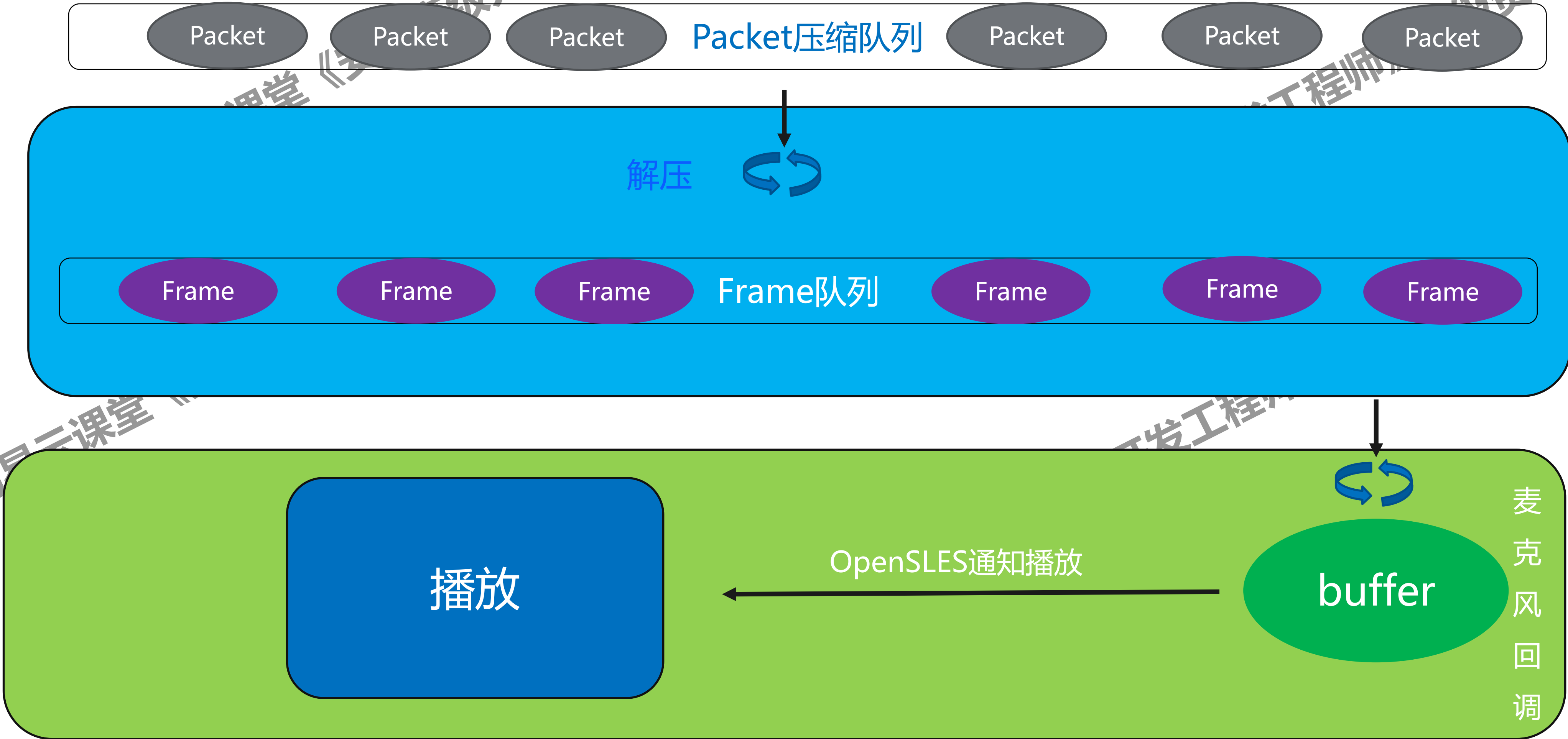
控  
制  
层



# 音频层

作用: 解码Packet 转换成Frame

解  
压  
线  
程





# 音视频同步

问题：音视频不同步

原因：由于FFmpeg是靠CPU来进行解码，音视频解码器都是异步进行解码。





# 音视频同步解决方案

## ■ 音视为准

以音频为准，视频与音频同步，音频只管播放，视频有超前和延后。如果超前放慢速度，如果延后 减少视频播放等待时间，如果延后很严重，队列积累时间过长，这直接丢弃，进入最新解码帧

## ■ 视频为准

视频画面每次循环不变，音频根据视频来 延迟或等待

## ■ 自定义时间为准

根据开始加载视频时间为0时间，后面的音频帧和视频帧依赖自定义时间进行同步

# AVRational

```
typedef struct AVRational{  
    int num; ///  
    int den; ///  
} AVRational;
```

AVRational这个结构标识一个分数

1 num为分子

2 den为分母。

# AVRational

## ■ 意思

DTS , Decoding Time Stamp, 解码时间戳, 告诉解码器packet的解码顺序。

PTS , Presentation Time Stamp, 显示时间戳, 指示从packet中解码出来的数据的显示顺序。

音频中二者是相同的, 但是视频由于B帧(双向预测)的存在, 会造成解码顺序与显示顺序并不相同, 也就是视频中DTS与PTS不一定相同。

# AVRational time\_base;

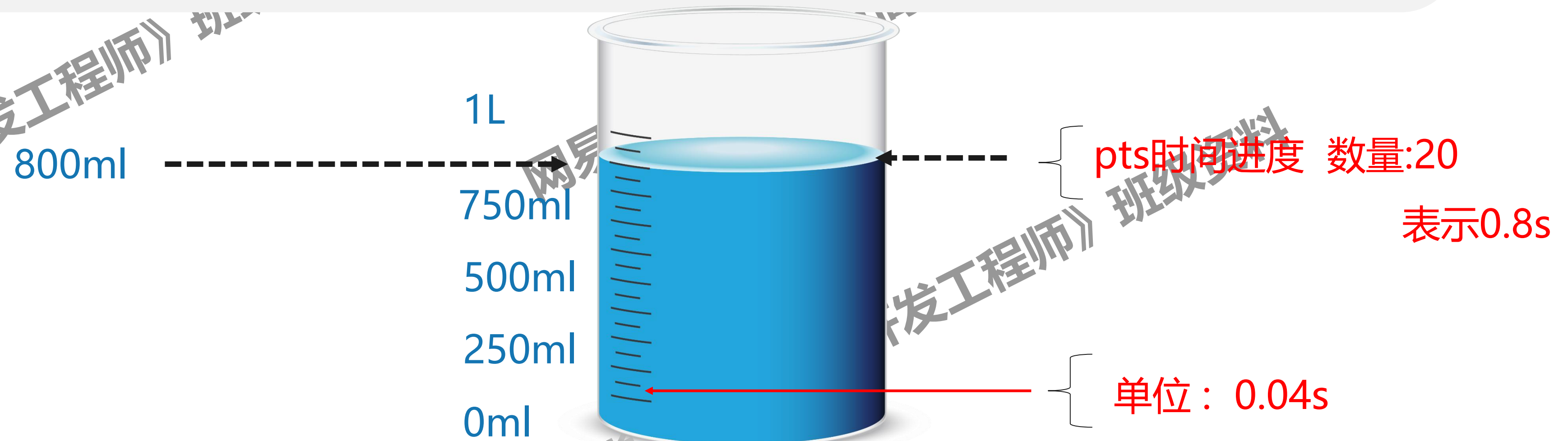
- 含义是时间 刻度

```
AVStream->AVCodecContex->time_base {1,100}
```

音频中二者是相同的，但是视频由于B帧（双向预测）的存在，会造成解码顺序与显示顺序并不相同，也就是视频中DTS与PTS不一定相同。

# 什么是单位

单位是人为定义的，如什么是1ml 你可以自己定义1ml的实际大小，国际公认是4摄氏度的水的10cm\*10cm\*10cm的容器体积为1L。这是标准，地球人都认可这种标准  
可是在音视频中 这种 参考标准会发生变化，用户可以自定义标准。标准可以用time\_base表示

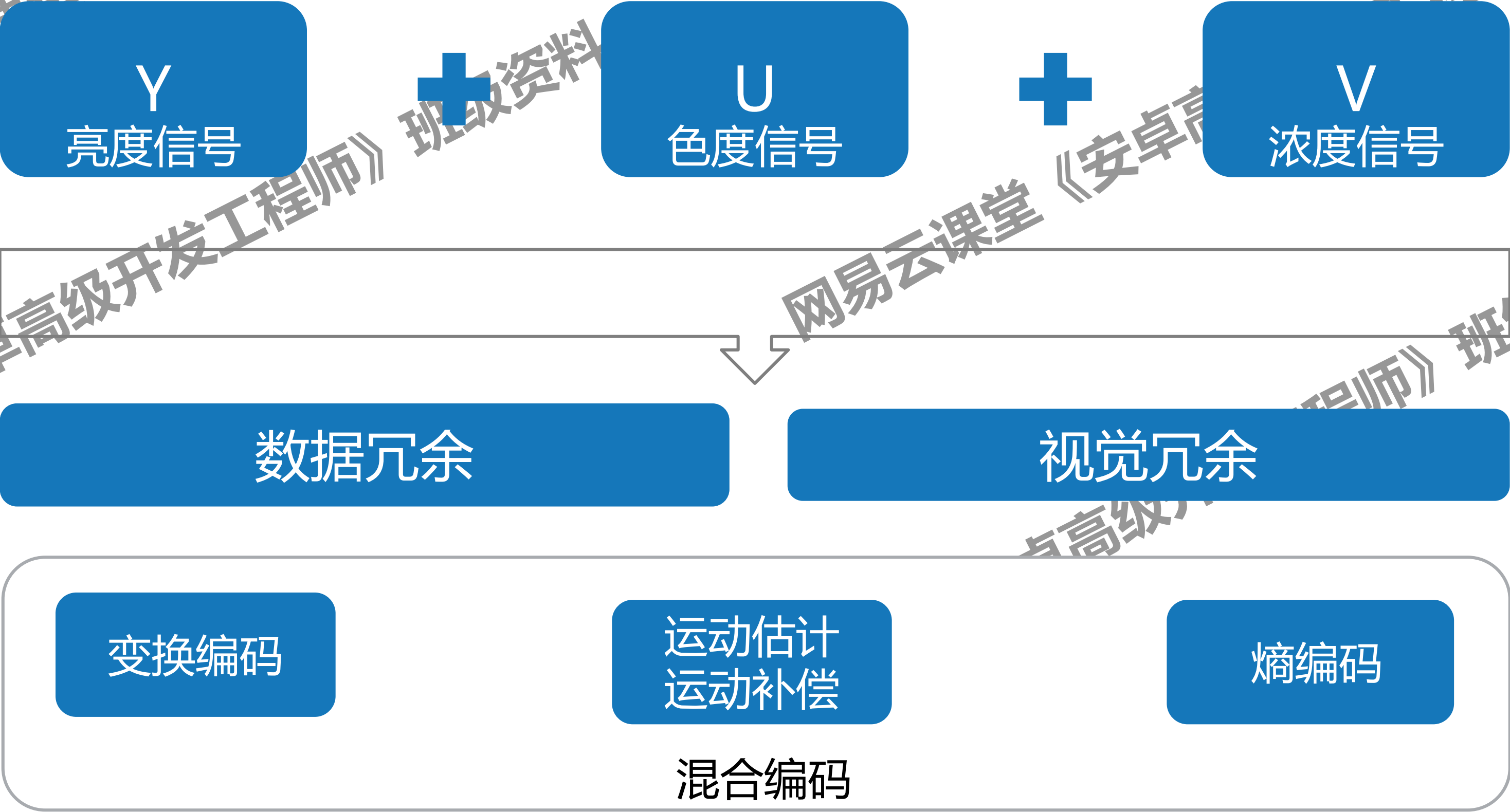


实际上time\_base的意思就是时间的刻度：如（1/25），那么时间刻度就是1/25 每一份是0.04s



# 视频编码原理

YUV信号采集之后通过数据冗余和视觉冗余这两个基本条件，再通过混合编码算法、交换编码、运动估计和运动补偿、熵编码三种结合来进行压缩编码。



# 变换编码

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

8x8图像块

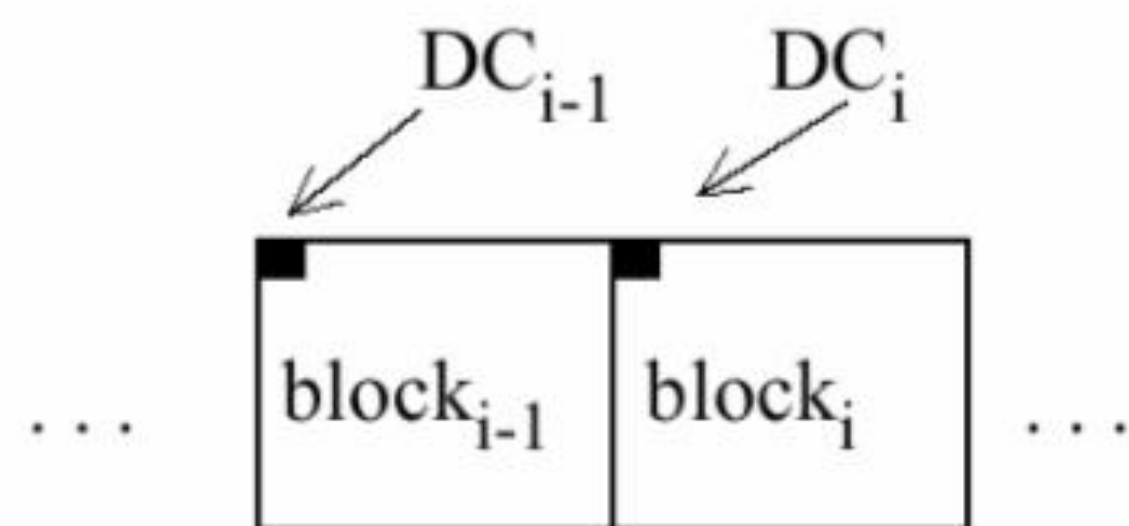
236	-1.0	-12	-5.2	2.1	-1.7	-2.7	-1.3
-22	-18	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-11	9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	1.6	-0.1	0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
-1.8	-0.2	1.6	-0.1	-0.8	1.5	1	-1
-1.3	-0.4	-0.3	-1.5	0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

图像块经过DCT变换后的系数

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

量化后的DCT系数

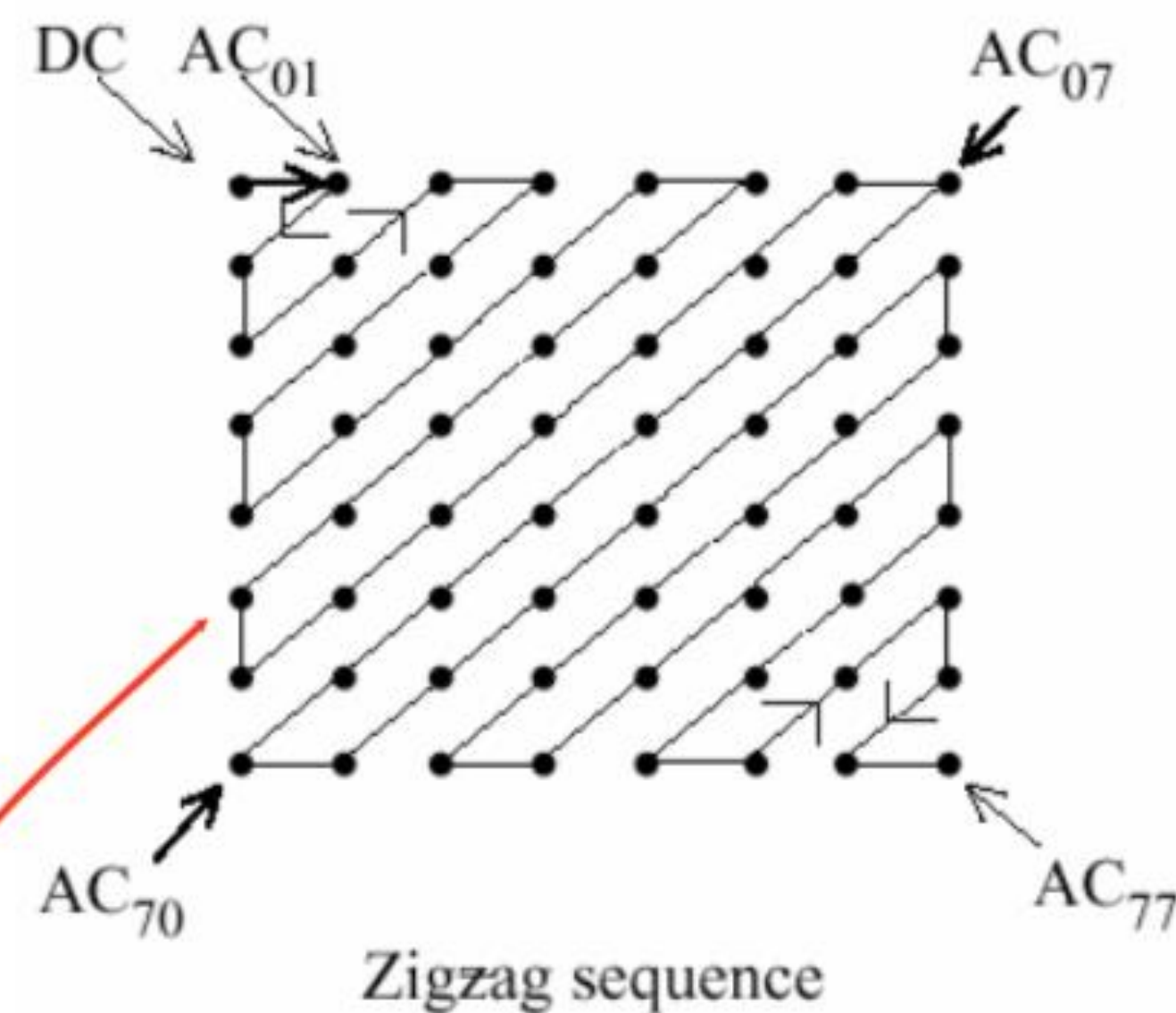
# 熵编码



$$\text{DIFF} = \text{DC}_i - \text{DC}_{i-1}$$

Differential DC encoding

Huffman coded



Runlength coding  
then huffman or  
arithmetic coding



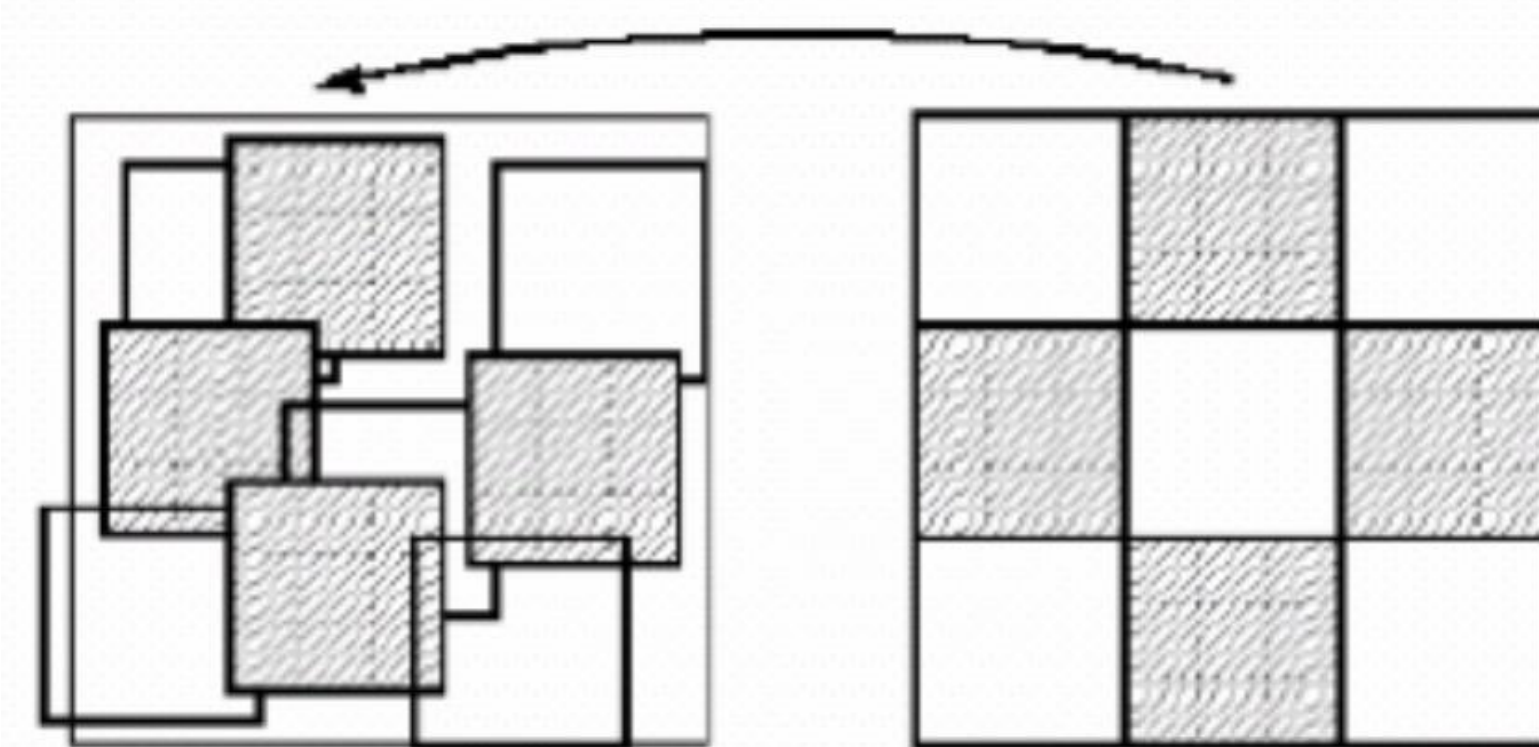
出处：视频压缩编码和音频压缩编码的基本原理



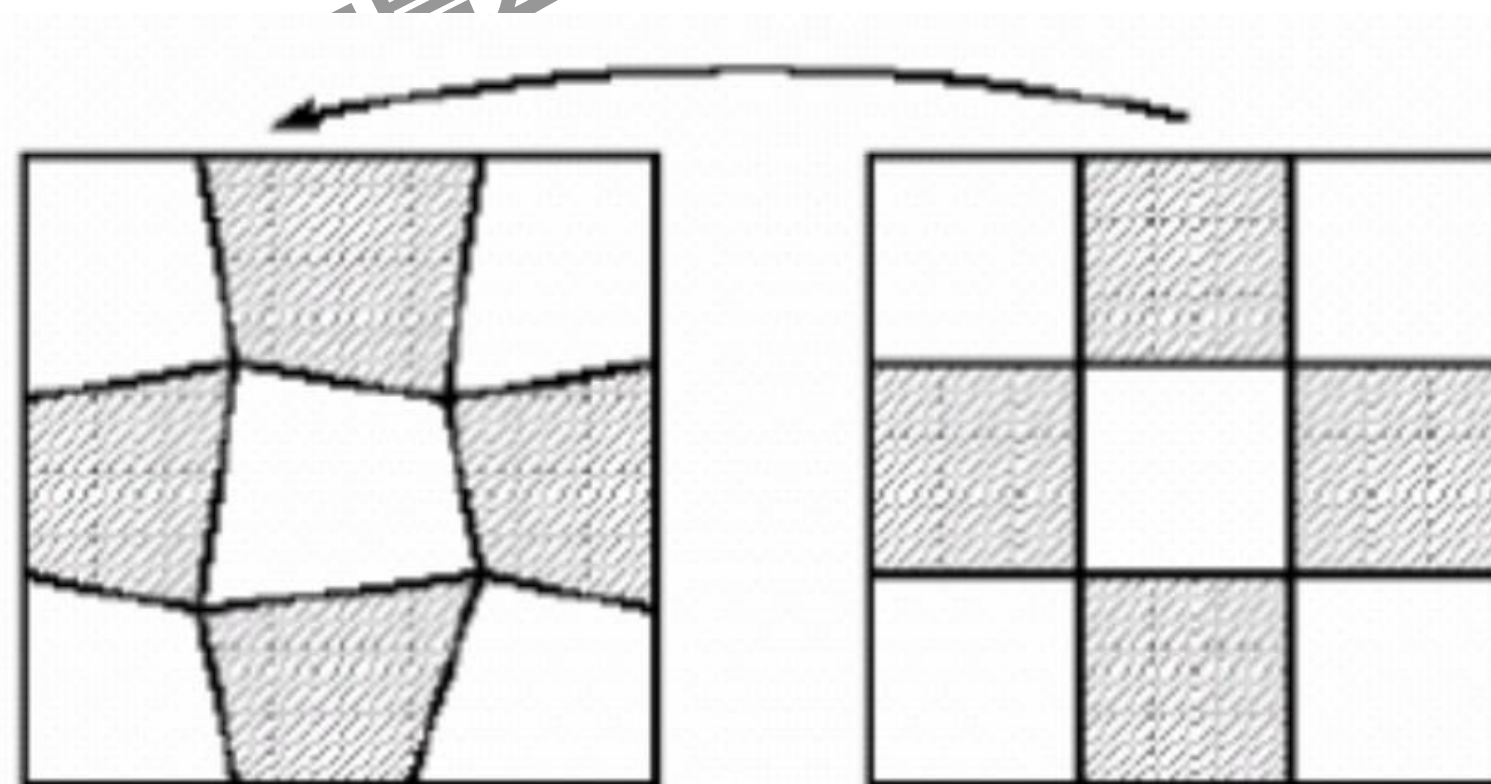
# 运动估计和运动补偿

## 运动估计示例

基于块的运动估计  
基于网格的运动估计



a 基于块的运动估计

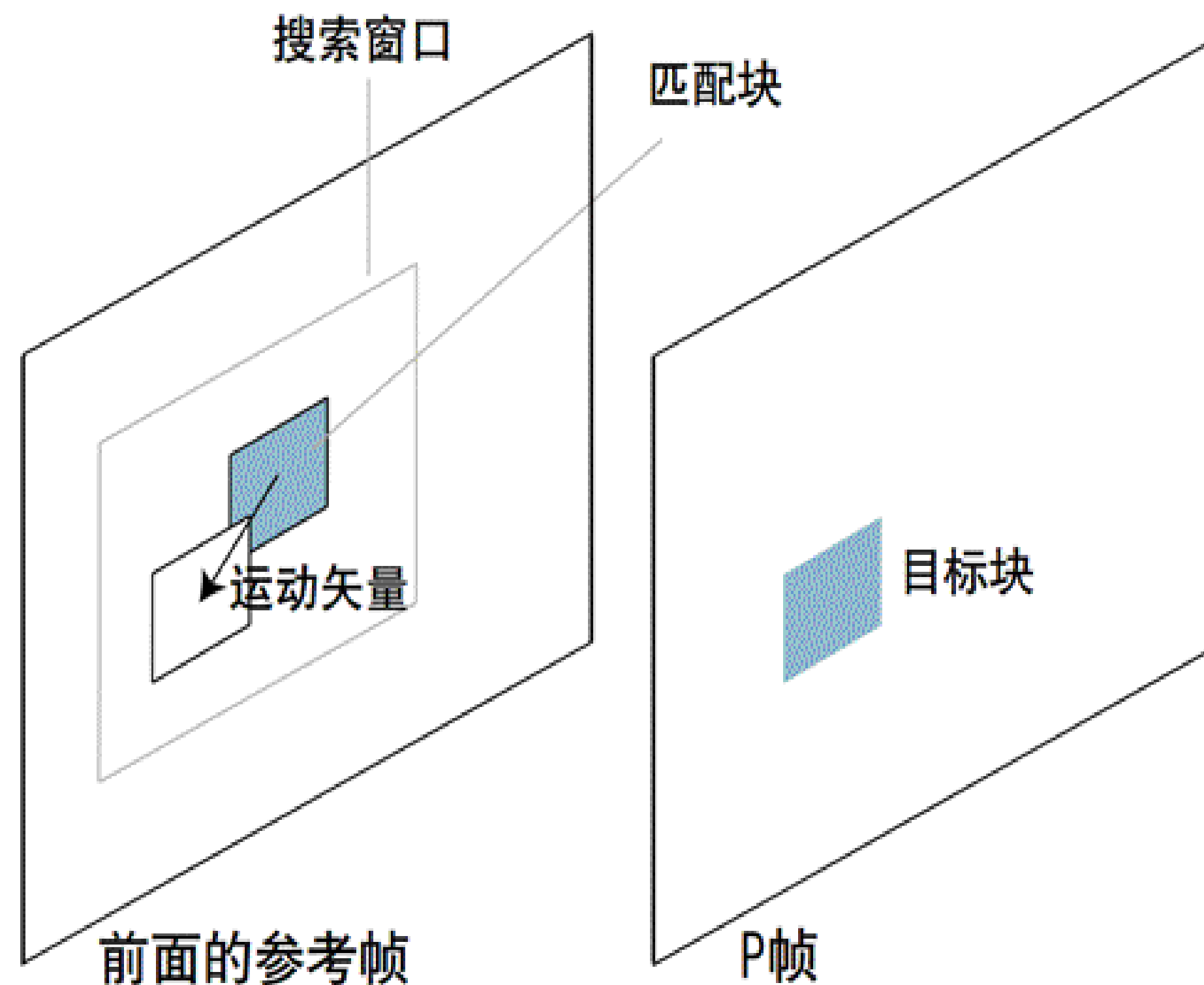


b 基于网格的运动估计

# 运动估计和运动补偿

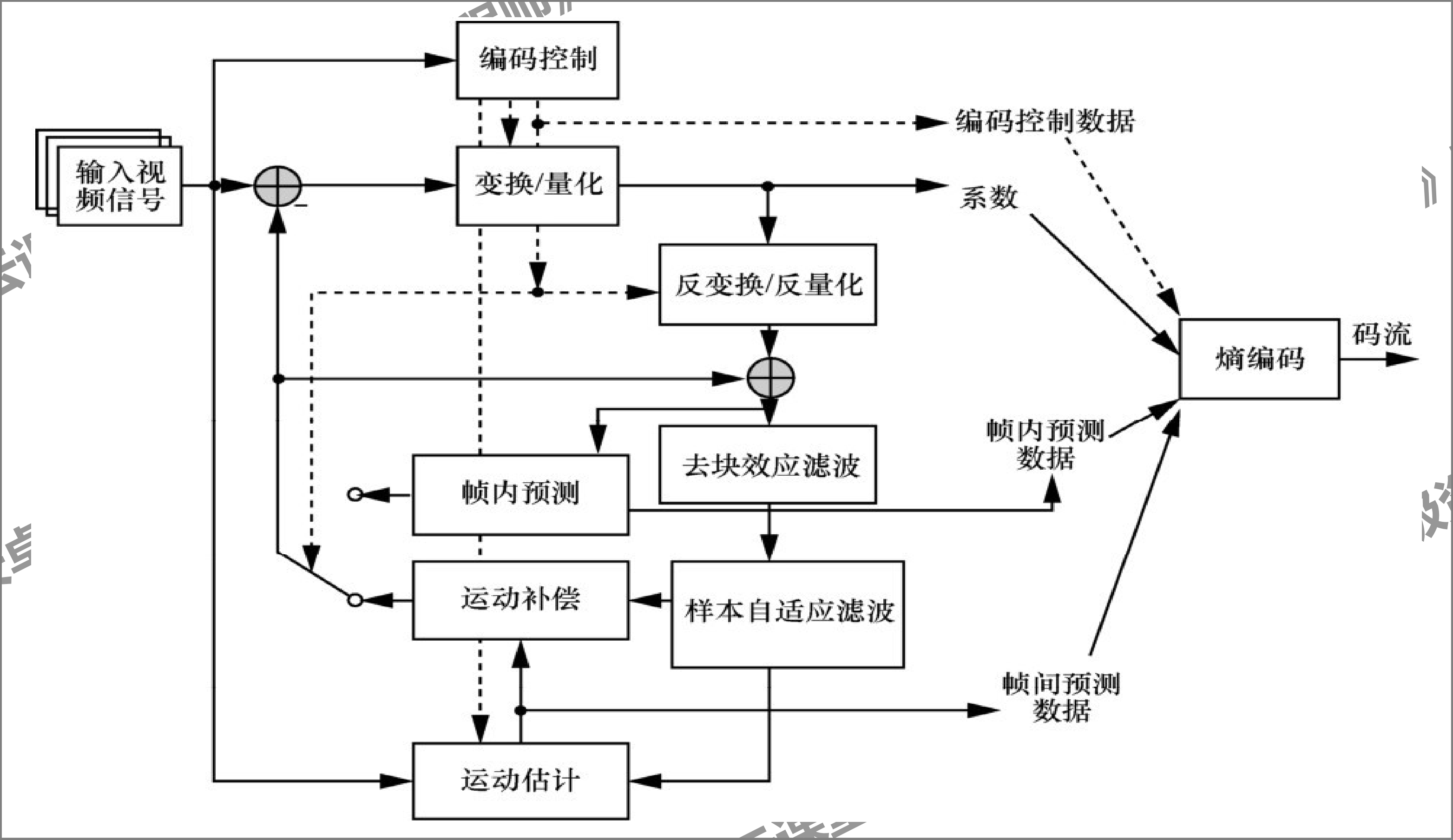
## 运动补偿示例

参考帧  
P帧





# 混合编码



混合编码模型，该模型普遍应用于MPEG1，MPEG2，H.264等标准中。

# 音频编码原理

数字音频信号

掩蔽效应

频谱掩蔽效应

时域掩蔽效应

# 什么是掩蔽效应

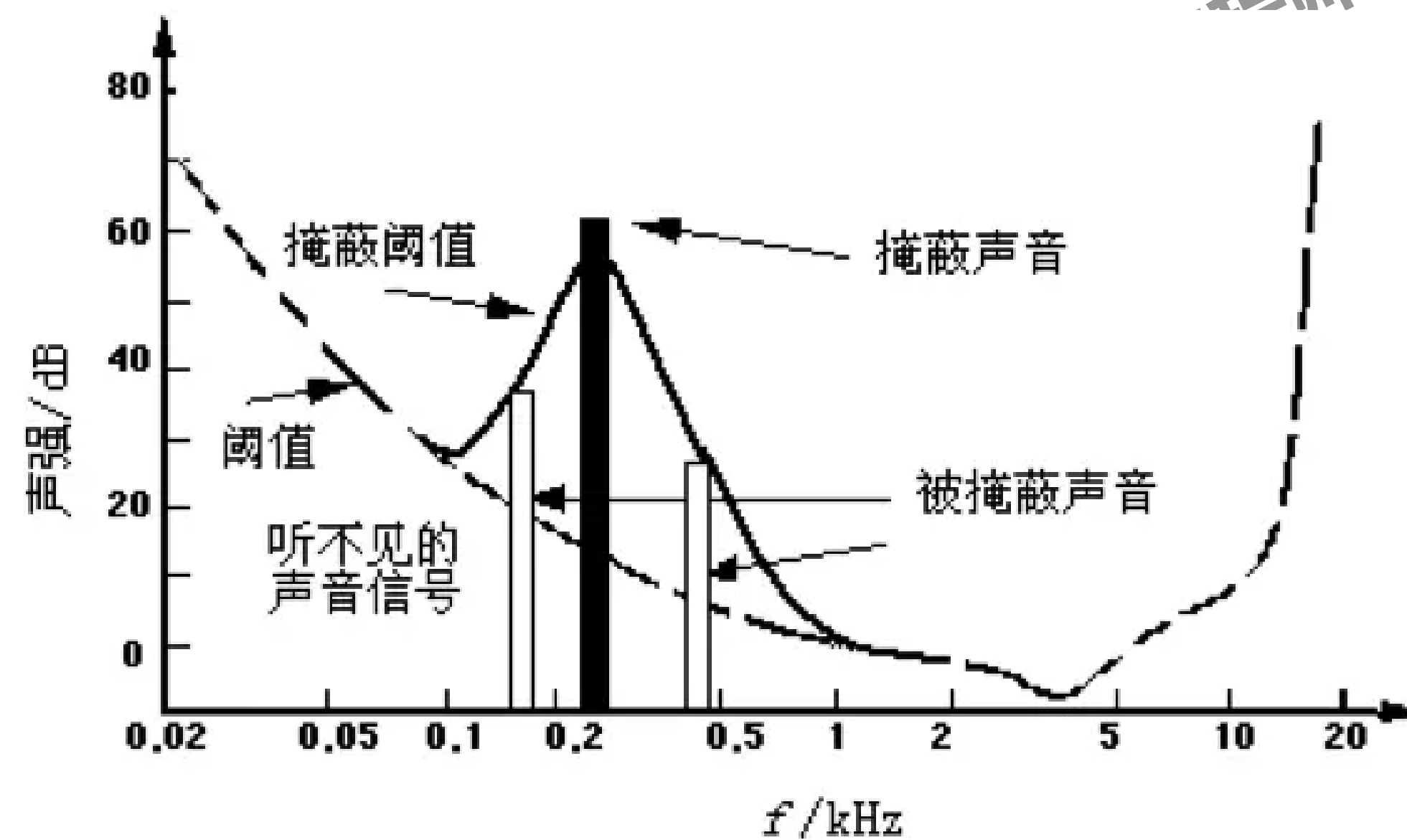
人耳对一个声音的听觉灵敏度因为另一个声音的存在而降低的现象，我们称为**掩蔽效应**：

1. 频率相近的两个纯音间易发生掩蔽效应，且较强声音易掩蔽较弱的声音
2. 低频纯音可以有效掩蔽高频纯音，反之作用小。



# 频谱掩蔽效应

2:发生在掩蔽声与被掩蔽声同时作用时  
1:强音会掩蔽其频率周围的弱音



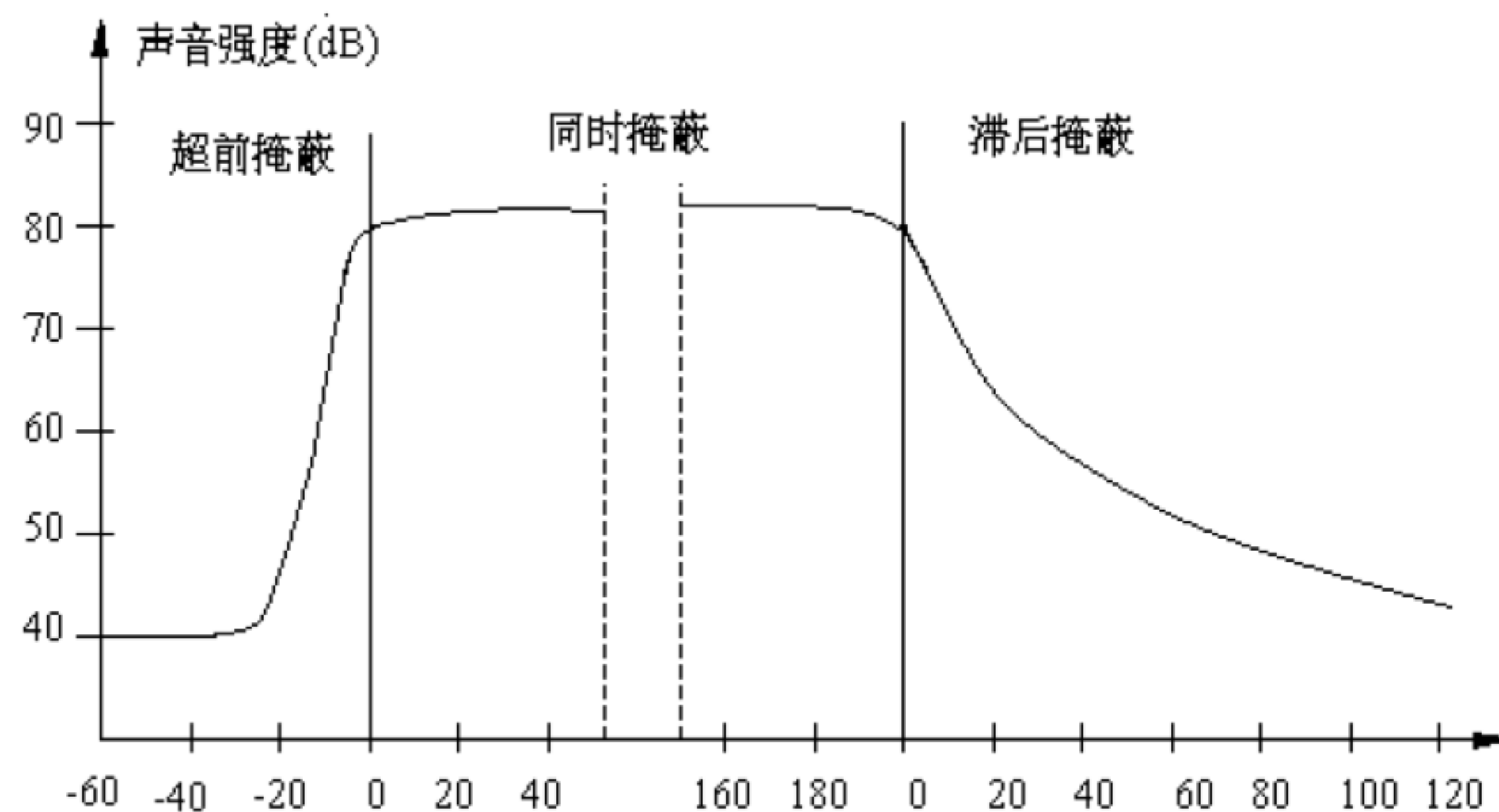
# 时域掩蔽效应

发生在掩蔽声与被掩蔽声不同时出现的时候

1:前掩蔽：是指人耳在听到强信号之前的短暂时间内，已经存在的弱信号会被掩蔽而听不到。

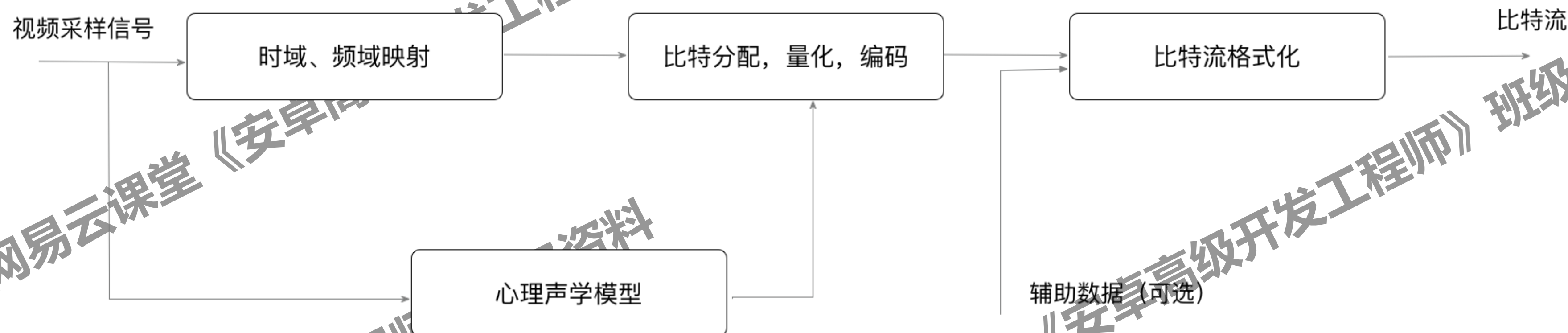
2:同时掩蔽：是指当强信号与弱信号同时存在时，弱信号会被强信号所掩蔽而听不到。

3:后掩蔽：是指当强信号消失后，需经过较长的一段时间才能重新听见弱信号。





# 音频压缩编码



数字音频编码系统模型

对每一个音频声道中的音频采样信号，首先都要将它们映射到频域中，这种时域到频域的映射可通过子带滤波器实现。每个声道中的音频采样块首先要根据心理声学模型来计算掩蔽门限值，然后由计算出的掩蔽门限值决定从公共比特池中分配给该声道的不同频率域中多少比特数，接着进行量化以及编码工作，最后将控制参数及辅助数据加入数据之中，产生编码后的数据流。

# 谢谢观看