

剖析ffmpeg视频解码播放：时间戳的处理

原创

泡沫oO

于 2023-05-31 17:46:38 发布

阅读量2.3k

收藏6

点赞数1

版权

分类专栏：

C/C++ 多媒体编程实践

文章标签：

ffmpeg

音视频

开发语言

C++

C语言

C/C++ 多媒体编程实践

专栏收录该内容

23 订阅

53 篇文章

订阅专栏

一、视频播放基础理论

1.1 视频编码和解码基础

视频编码^Q和解码是视频播放的基础，理解它们的工作原理对于深入理解视频播放至关重要。在这一部分，我们将详细介绍视频编码和解码的基础知识。

视频编码（Video Encoding）是将原始视频 **数据转换**^Q为特定格式的过程，以便于存储或传输。视频编码的主要目标是压缩原始视频数据，以减少所需的存储空间和带宽。视频编码通常涉及到以下几个步骤：

- 帧间预测（Inter-frame Prediction）

这个步骤是通过比较相邻的帧来预测当前帧的内容。例如，如果两个相邻的帧的内容非常相似，那么我们可以只存储第一个帧的内容和两个帧的差异，而不是存储两个帧的完整内容。
- 变换编码（Transform Coding）

这个步骤是将像素数据转换为频域数据，以便于压缩。常用的变换编码方法包括离散余弦变换（DCT）和快速傅里叶变换（FFT）。
- 量化（Quantization）

这个步骤是将连续的数据转换为离散的数据，以便于压缩。量化的过程通常会损失一些信息，但是如果量化的级别选择得当，这种损失可以被人眼忽略。
- 熵编码（Entropy Coding）

这个步骤是将数据编码为比特流，以便于存储或传输。常用的熵编码方法包括哈夫曼编码（Huffman Coding）和算术编码（Arithmetic Coding）。

视频解码（Video Decoding）则是视频编码的逆过程，它将编码后的视频数据转换回原始的视频数据。视频解码的过程通常涉及到以下几个步骤：

- 熵解码（Entropy Decoding）

这个步骤是将比特流解码为数据。这个过程是熵编码的逆过程。
- 反量化（Dequantization）

这个步骤是将离散的数据转换回连续的数据。这个过程是量化的逆过程。
- 反变换编码（Inverse Transform Coding）

这个步骤是将频域数据转换回像素数据。这个过程是变换编码的逆过程。
4.

帧间预测解码（Inter-frame Prediction Decoding）：这个步骤是通过预测的差异和参考帧来恢复当前帧的内容。这个过程是帧间预测的逆过程。

以上就是视频编码和解码的基础知识。理解这些知识点可以帮助我们更好地理解 **视频播放**^Q的过程。在接下来的部分，我们将深入探讨时间戳的作用以及如何处理时间戳。

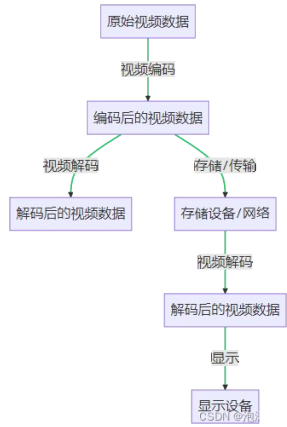
1.2 时间戳的作用与理解(PTS和DTS)

在视频播放中，时间戳是非常重要的部分，它决定了视频帧的解码和显示顺序。在视频编码中，我们主要关注两种时间戳：解码时间戳（Decoding Time Stamp, DTS）和显示时间戳（Presentation Time Stamp, PTS）。

解码时间戳（DTS）是指视频帧应该何时被解码。在一些复杂的视频编码格式中，例如使用了B帧（双向预测帧）的编码格式，帧的解码顺序可能不同于它们的显示顺序。在这种情况下，DTS就变得非常重要，因为它可以帮助解码器确定何时解码每一帧。

显示时间戳（PTS）则是指视频帧应该何时被显示。在播放视频时，我们需要根据PTS来控制每一帧的显示时间，以保证视频的播放速度与视频的帧率相匹配，也可以保证音频和视频的同步。

下面的图示展示了视频数据从原始数据到显示设备的整个过程，其中包括了视频编码、存储/传输、视频解码和显示等步骤。在这个过程中，DTS和PTS起到了关键的作用。



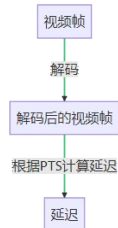
1.3 根据PTS控制视频帧的显示

在播放视频时，我们需要根据视频帧的显示时间戳（PTS）来控制每一帧的显示时间。这通常涉及到在每一帧之间添加适当的延迟，以保证每一帧都在正确的时间被显示。

具体来说，我们可以通过比较当前时间和下一帧的PTS来计算出需要的延迟。例如，假设当前时间是T1，下一帧的PTS是T2，那么我们需要等待(T2-T1)的时间后再显示下一帧。如果T2小于或等于T1，说明下一帧应该立即被显示。

这种方法可以保证视频的播放速度与视频的帧率相匹配，也可以保证音频和视频的同步（如果有音频的话）。但是，这需要播放器有能力精确地控制延迟，这在某些情况下可能是个挑战（例如，如果播放器的计时精度不够，或者系统的调度延迟太大）。

下面的图示展示了如何根据PTS控制视频帧的显示的过程：



目录

发布首篇原创文章，
原力分+10，点亮新秀勋章

一、视频播放基础理论

1.1 视频编码和解码基础

1.2 时间戳的作用与理解（PTS和DT...

1.3 根据PTS控制视频帧的显示

二、视频播放中的时间戳处理

2.1 如何获取和解析PTS和DTS

2.2 如何根据PTS控制视频帧的显示

2.3 解码时间戳（DTS）在解码中的...

三、处理复杂视频编码格式的挑战

3.1 B帧（Bi-directional Predicted F...

3.2 变帧率（Variable Frame Rate）...

分类专栏		
	C/C++ 应用工程师...	付费 37篇
	C/C++ 编程世界: 探索C/...	426篇
	C/C++ 基础知识	35篇
	C/C++ 函数和过程	45篇
	C/C++ 关键字指南	19篇
	C++ 中的封装和抽象...	15篇
	C++ 多态的精彩实现	20篇
	C/C++ 数据结构	28篇
	C++ 的并发编程	16篇
	C++内存管理	14篇
	C++ 泛型编程精选...	52篇
	C++ 新特性	23篇
	C/C++ 其他实用手段	17篇
	C/C++ 软件设计思路	14篇
	C/C++ 编程规范分享	2篇
	基本算法	4篇
	CMake核心教程：从...	53篇
	C/C++ 多媒体编程...	53篇
	C/C++性能优化	21篇
	设计模式的精髓：用最...	43篇
	Qt应用开发·探索Qt的...	120篇
	通信技术探索：连接未来	5篇
	QML动态界面之美：Qt Q...	29篇
	Linux系统编程：从入门...	127篇
	交叉编译的艺术与科...	14篇
	Shell 编程设计	7篇
	Shell命令集合	309篇
	协议探秘之旅	15篇
	网络编程知识	3篇
	编程与辅助软件	8篇
	C/C++ 工程师 日常篇	13篇
	C/C++应用开发面试精选	36篇
	C++项目设计：理论、实...	11篇
	软考_软件设计师专题	131篇
	探索计算机世界：从基础...	12篇
	Python 基础教程	29篇
	基础的rust的英文教程	5篇
	计算机相关 词汇记忆	1篇



二、视频播放中的时间戳处理

2.1 如何获取和解析PTS和DTS

在视频播放中，时间戳是非常关键的一部分，它决定了视频帧的解码和显示顺序。在这个小节中，我们将详细介绍如何获取和解析显示时间戳（PTS，Presentation Time Stamp）和解码时间戳（DTS，Decoding Time Stamp）。

首先，我们需要明确一点，PTS和DTS都是在解码前就已经知道的，它们都是存储在AVPacket结构中的。当你从媒体文件中读取一个AVPacket时，你就可以获取到这个AVPacket的DTS和PTS。

在FFmpeg这样的媒体库中，你可以通过以下方式获取AVPacket的DTS和PTS：

```
1 AVPacket* packet = av_packet_alloc();
2 // 假设你已经打开了媒体文件，并创建了AVFormatContext* formatContext
3 int ret = av_read_frame(formatContext, packet);
4 if (ret == 0) {
5     int64_t dts = packet->dts;
6     int64_t pts = packet->pts;
7     // 你现在可以使用dts和pts了
8 }
```

在这个例子中，我们首先创建了一个AVPacket，然后使用av_read_frame函数从媒体文件中读取一个帧。如果读取成功，我们就可以直接从AVPacket中获取DTS和PTS。

需要注意的是，DTS和PTS的单位通常是时间基（time base），而不是秒或者毫秒。时间基是一个表示时间的单位，它通常是一个分数，表示一帧的时间长度。你可以通过AVStream的time_base字段获取到时间基。

在获取到DTS和PTS后，你可能需要将它们转换为你需要的时间单位。在FFmpeg中，你可以使用以下方式将DTS和PTS转换为秒：

```
1 AVStream* stream = formatContext->streams[packet->stream_index];
2 double dts_in_seconds = dts * av_q2d(stream->time_base);
3 double pts_in_seconds = pts * av_q2d(stream->time_base);
```

在这个例子中，我们首先获取到了包含当前帧的流（AVStream），然后使用av_q2d函数和流的时间基将DTS和PTS转换为秒。

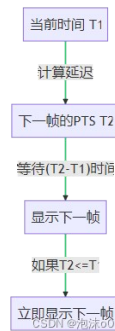
至此，我们已经知道了如何获取和解析PTS和DTS。在下一节中，我们将讨论如何根据PTS来控制视频帧的显示。

2.2 如何根据PTS控制视频帧的显示

在视频播放中，我们需要根据显示时间戳（PTS）来控制每一帧的显示时间，以保证视频的播放速度与视频的帧率相匹配，也可以保证音频和视频的同步。具体来说，我们可以通过比较当前时间和下一帧的PTS来计算出需要的延迟。

例如，假设当前时间是T1，下一帧的PTS是T2，那么我们需要等待(T2-T1)的时间后再显示下一帧。如果T2小于或等于T1，说明下一帧应该立即被显示。

下面是一个简单的示意图，描述了这个过程：



这种方法可以保证视频的播放速度与视频的帧率相匹配，也可以保证音频和视频的同步。但是，这需要播放器有能力精确地控制延迟，这在某些情况下可能是个挑战（例如，如果播放器的计时精度不够，或者系统的调度延迟太大）。

在实际的编程中，我们可以使用各种方法来实现这个延迟。例如，我们可以使用线程睡眠（thread sleep）、定时器（timer）或者事件循环（event loop）等方法。具体的选择取决于你的应用程序的需求和环境。

在下一节中，我们将讨论解码时间戳（DTS）在解码中的作用和处理方法。

2.3 解码时间戳(DTS)在解码中的作用

解码时间戳（DTS，Decoding Time Stamp）是指媒体帧（音频或视频）应该何时被解码。这在视频流中特别重要，因为在许多情况下，帧的解码顺序并不同于它们的显示顺序。例如，许多视频编码格式使用B帧（双向预测帧），这些帧需要在它们之前和之后的帧都被解码后才能解码。在这种情况下，DTS可以帮助解码器确定何时解码每一帧。

具体来说，DTS的主要作用是帮助解码器确定何时解码每一帧。在一些复杂的视频编码格式中，例如使用了B帧（双向预测帧）的编码格式，帧的解码顺序可能不同于它们的显示顺序。在这种情况下，DTS就变得非常重要，因为它可以帮助解码器确定何时解码每一帧。

在实际的编程中，我们可以通过以下方式获取AVPacket的DTS：

```
1 AVPacket* packet = av_packet_alloc();
2 // 假设你已经打开了媒体文件，并创建了AVFormatContext* formatContext
3 int ret = av_read_frame(formatContext, packet);
4 if (ret == 0) {
5     int64_t dts = packet->dts;
6     // 你现在可以使用dts了
7 }
```

在这个例子中，我们首先创建了一个AVPacket，然后使用av_read_frame函数从媒体文件中读取一个帧。如果读取成功，我们就可以直接从AVPacket中获取DTS。

总的来说，虽然DTS和PTS都是重要的，但在大多数情况下，你可以直接从队列中取出AVPacket并解码，不需要特别关注DTS。你需要关注的是PTS，因为它决定了每一帧的显示时间。

三、处理复杂视频编码格式的挑战

3.1 B帧(Bi-directional Predicted Frame)的理解和处理

在视频编码中，B帧（双向预测帧）是一种特殊的帧类型，它的内容是根据它之前和之后的帧来预测的。这种预测方式使得B帧能够更有

效地压缩视频，从而减小视频文件的大小。然而，B帧也给视频的解码和播放带来了一些挑战。

首先，我们来理解一下B帧的工作原理。在一个典型的视频序列中，我们可能会有以下的帧序列：

1		I	P	B	B	P	B	B	I
---	--	---	---	---	---	---	---	---	---

在这个序列中，帧（Intra-coded Picture）是关键帧，它是完全自足的，可以独立解码。P帧（Predicted Picture）是预测帧，它的内容是根据前一个帧或P帧预测的。B帧则是双向预测帧，它的内容是根据前后的帧或P帧预测的。

因为B帧的内容是根据前后的帧预测的，所以在解码B帧时，我们需要先解码它前后的帧。这就意味着，B帧的解码顺序（DTS）通常早于它的显示顺序（PTS）。例如，对于上面的帧序列，如果我们按照显示顺序（PTS）来解码，那么解码顺序（DTS）可能是这样的：

1		I	P	P	B	B	P	B	B	I
---	--	---	---	---	---	---	---	---	---	---

可以看到，所有的B帧都被移动到了它们前后的P帧之后。

这种解码顺序和显示顺序的不同，给视频的解码和播放带来了一些挑战。在实际应用中，我们需要根据每一帧的DTS来进行解码，然后再根据每一帧的PTS来进行显示。这就需要在解码和播放时，都能正确地处理DTS和PTS。

在下一节中，我们将详细介绍如何处理这种挑战。

3.2 变帧率 (Variable Frame Rate) 视频的处理

变帧率（Variable Frame Rate，VFR）是一种特殊的视频编码方式，其中的帧率并不是固定的，而是可以根据视频内容的变化而变化。例如，如果视频的某一部分内容变化较小（例如，静止画面或者慢动作），那么这一部分的帧率可能会降低；反之，如果视频的某一部分内容变化较大（例如，快速运动或者大量细节），那么这一部分的帧率可能会提高。这种方式可以更有效地压缩视频，从而减小视频文件的大小。

然而，变帧率也给视频的解码和播放带来了一些挑战。因为帧率不再是固定的，所以我们不能简单地通过帧率来预测每一帧的显示时间戳（PTS）。相反，我们需要根据每一帧的实际PTS来进行显示。

在处理变帧率视频时，我们需要注意以下几点：

- 解码顺序和显示顺序可能不同：**和B帧一样，变帧率视频中的解码顺序（DTS）和显示顺序（PTS）可能不同。我们需要根据每一帧的DTS来进行解码，然后再根据每一帧的PTS来进行显示。
- PTS可能不连续：**在固定帧率的视频中，相邻两帧的PTS之差通常是固定的。但在变帧率的视频中，相邻两帧的PTS之差可能会变化。我们需要正确处理这种情况。
- 需要精确的计时：**因为PTS可能会变化，所以我们需要有能力精确地控制每一帧的显示时间。这可能需要高精度的计时器，以及能够精确控制显示硬件的能力。

3.3 网络传输中的挑战

在网络上接收和播放视频流时，网络的状况可能会对视频的解码和播放带来一些挑战。例如，网络的延迟、丢包、带宽变化等都可能影响到视频的解码和播放。

- 网络延迟：**网络的延迟会影响到视频帧的接收时间，从而影响到帧的解码和显示时间。如果网络延迟较大，那么可能需要在播放器中添加一定的缓冲，以减少因网络延迟引起的播放卡顿。
- 丢包：**在网络传输中，可能会出现丢包的情况。如果丢失的是关键帧（帧），那么可能会影响到后续帧的解码，因为后续的预测帧（P帧）和双向预测帧（B帧）都依赖于关键帧。在这种情况下，可能需要请求重传丢失的帧，或者跳过不能解码的帧。
- 带宽变化：**如果网络的带宽发生变化，那么可能需要动态调整视频的码率，以适应当前的网络状况。这可能需要支持多码率的视频编码格式，以及能够动态切换码率的播放器。

在处理这些挑战时，我们需要注意以下几点：

- 预缓冲：**为了减少因网络延迟引起的播放卡顿，我们可以在播放器中添加一定的预缓冲。预缓冲的大小可以根据网络的状况动态调整。
- 错误恢复：**为了处理丢包的情况，我们需要在播放器中添加错误恢复的机制。这可能包括请求重传丢失的帧，或者跳过不能解码的帧。
- 码率切换：**为了适应网络带宽的变化，我们需要支持动态码率切换。这可能需要支持多码率的视频编码格式，以及能够动态切换码率的播放器。

四、实际应用中的时间戳处理策略

4.1 面对固定帧率视频的处理策略

在处理固定帧率（Fixed Frame Rate）的视频时，我们可以利用视频帧率的固定性质来简化时间戳（Timestamp）的处理。固定帧率意味着每一帧的显示时间间隔是固定的，这个间隔就是帧率的倒数。例如，如果视频的帧率是30帧/秒，那么每一帧的显示时间间隔就是1/30秒。

在这种情况下，我们可以通过以下步骤来处理时间戳：

- 读取第一帧的显示时间戳（PTS）：**当我们从媒体文件中读取第一帧时，我们可以获取到这一帧的PTS。这个PTS将作为我们的基准时间，用来计算后续每一帧的显示时间。
- 计算每一帧的预期PTS：**由于我们知道视频的帧率，所以我们可以计算出每一帧的预期PTS。具体来说，第n帧的预期PTS就是基准时间加上n乘以帧间隔。例如，如果基准时间是T，帧率是30帧/秒，那么第n帧的预期PTS就是T + n/30。
- 根据预期PTS控制每一帧的显示：**当我们解码出一帧时，我们可以获取到这一帧的实际PTS。然后，我们可以比较这一帧的实际PTS和预期PTS，以决定何时显示这一帧。如果实际PTS早于预期PTS，那么我们需要等待一段时间后再显示这一帧；如果实际PTS晚于预期PTS，那么我们应该立即显示这一帧。

这种处理策略的优点是简单易懂，适用于大多数的视频播放场景。然而，它也有一些局限性。首先，它假设视频的帧率是固定的，这在实际应用中可能并不总是成立。其次，它假设每一帧的解码时间可以忽略不计，这在处理高清或者复杂编码格式的视频时可能并不成立。因此，当我们面对这些挑战时，我们可能需要采用更复杂的时间戳处理策略。

4.2 面对变帧率视频的处理策略

变帧率（Variable Frame Rate，VFR）视频是一种帧率不固定的视频，这种视频的每一帧的显示时间间隔可能会有所不同。处理VFR视频的时间戳（Timestamp）比处理固定帧率（Fixed Frame Rate，FFR）视频的时间戳更为复杂，因为我们不能简单地假设每一帧的显示时间间隔是固定的。

在处理VFR视频的时间戳时，我们需要采取以下步骤：

- 读取每一帧的显示时间戳（PTS）：**当我们从媒体文件中读取每一帧时，我们需要获取到这一帧的PTS。这个PTS将告诉我们这一帧应该在何时被显示。
- 计算每一帧的显示延迟：**由于VFR视频的每一帧的显示时间间隔可能不同，所以我们需要为每一帧计算一个显示延迟。具体来说，第n帧的显示延迟就是第n帧的PTS减去第n-1帧的PTS。例如，如果第n-1帧的PTS是T1，第n帧的PTS是T2，那么第n帧的显示延迟就是T2 - T1。
- 根据显示延迟控制每一帧的显示：**当我们解码出一帧时，我们可以获取到这一帧的显示延迟。然后，我们需要等待这个延迟的时间后再显示这一帧。

这种处理策略的优点是它可以准确地控制每一帧的显示时间，从而保证VFR视频的播放质量。然而，它也有一些局限性。首先，它需要为每一帧计算显示延迟，这可能会增加处理的复杂性。其次，它假设每一帧的解码时间可以忽略不计，这在处理高清或者复杂编码格式的视频时可能并不成立。因此，当我们面对这些挑战时，我们可能需要采用更复杂的时间戳处理策略。

4.3 面对复杂编码格式的处理策略

在处理一些复杂的视频编码格式时，我们可能需要采用更复杂的时间戳（Timestamp）处理策略。这是因为这些编码格式可能会使用一些特殊的帧类型，例如B帧（Bi-directional Predicted Frame），这些帧的解码顺序可能不同于它们的显示顺序。

在处理这种情况时，我们需要考虑到解码时间戳（DTS）和显示时间戳（PTS）的区别：

解码时间戳（DTS）：DTS是指媒体帧（音频或视频）应该何时被解码。在一些复杂的视频编码格式中，例如使用了B帧的编码格式，帧的解码顺序可能不同于它们的显示顺序。在这种情况下，DTS可以帮助解码器确定何时解码每一帧。

显示时间戳（PTS）：PTS则是指媒体帧应该何时被显示。在播放视频时，你需要根据PTS来控制每一帧的显示时间，以保证视频的播放速度与视频的帧率相匹配，也可以保证音频和视频的同步。

在处理这种情况时，我们需要采取以下步骤：

读取每一帧的DTS和PTS：当我们从媒体文件中读取每一帧时，我们需要获取到这一帧的DTS和PTS。这两个时间戳将告诉我们这一帧应该在何时被解码和显示。

根据DTS控制每一帧的解码：我们需要根据每一帧的DTS来控制解码的顺序。具体来说，我们需要先解码DTS较早的帧，然后再解码DTS较晚的帧。

根据PTS控制每一帧的显示：我们需要根据每一帧的PTS来控制显示的时间。具体来说，我们需要等待到PTS指定的时间后再显示这一帧。

这种处理策略的优点是它可以准确地控制每一帧的解码和显示时间，从而保证视频的播放质量。然而，它也有一些局限性。首先，它需要为每一帧计算显示延迟，这可能会增加处理的复杂性。其次，它假设每一帧的解码时间可以忽略不计，这在处理高清或者复杂编码格式的视频时可能并不成立。因此，当我们面对这些挑战时，我们可能需要采用更复杂的时间戳处理策略

这部分的内容主要参考了以下的文献：

1. Table 1: The detailed process of the video feature decoding.

该文章知识点与官方知识档案匹配，可进一步学习相关知识

C技能树 > 首页 > 概览 178455 人正在系统中学习

ffmpeg视频解码demo 12-26
裁剪编译ffmpeg并用于实现安卓端**解码**demo的源码。

FFmpeg时间戳详解 yinshipin007的博文 883
当然，不含B帧的**视频**，其DTS和PTS是相同的。AVStream.time_base是AVPacket中pts和dts的时间单位，输入流与输出流中time_base按如下...

ffmpeg：通过视频PTS获取当前帧所在的时间 ETalien的博文 1944
FFmpeg通过PTS获取当前帧所在的时间 / 输入对应流，获取帧 AVStream *stream=pFormatCtx->streams[packet_stream_index]; // **解码** avcodec_decode...

ffmpeg转码时对变帧率和固定帧率的处理 zww_sap111的专栏 7650
http://www.roscoo.net/a/201107/14663.html 一般fps在代码里这样表示 Fps = den/num 如果den = 15, num=1, 则fps = 15。如果帧率固定，pts*fps 就表...

FFmpeg视频解码VS2015工程 最新发布 08-02
FFmpeg**视频解码**的简洁示例，可打开**视频文件**、**解码视频帧**、YUV转RGB，并显示**视频**图像。已封装为CVideoDecoder类，代码用C++编写，VS2015编...

ffmpeg转码MPEG2-TS的音视频同步机制分析 fanbird2008的专栏 1566
http://blog.chinaunix.net/uid-26000296-id-3483782.html 一、FFmpeg忽略了adaptation_field()数据 FFmpeg忽略了包含PCR值的adaptation_field数据; ...

ffmpeg 添加时间戳 12-24
Windows下安装 ffmpeg 添加**时间戳** 简单教程 图像处理

如何用FFmpeg读取内存的PS/TS流，并分离出视频和音频 相见不如怀念 1536
之前做一个项目遇到一个问题：从网络中收到PS/TS流，需要从中分离出**视频**和**音频**，但是FFmpeg只支持标准的几种输入流协议（RTP/HTTP/RTSP/MM...

mp4视频提取时间戳 W_E_DAY的博文 698
MP4**视频提取时间戳**

ffmpeg 之 时间戳 qq_38883139的博文 1357
看到好多人对**时间戳**这个概念不明白啊,简单说一下我的理解 第一,**时间戳**是什么 **时间戳**就是一个能够表示一个事物发生时间的东西,她有单位,比如秒,毫...

FFmpeg解码并播放视频 10-11
在Android ndk中使用FFmpeg**解码**并**播放视频**，全部是自己手写的，完全可用。如有问题大家多沟通交流。

基于ffmpeg对摄像头采集的视频加上时间戳水印 07-13
基于ffmpeg调用摄像头并通过sdl显示，加上了**时间戳**水印。其中摄像头名称记得改成自己的

雷霄骅——FFmpeg视频解码器 12-28
雷神的《基于FFmpeg + SDL的**视频播放器**的制作》课程的**视频**中的3部分代码，整体上传审核不通过，我也不知道为什么CSDN会把雷神的开源代码去设...

C++基于FFmpeg的音频解码和播放 12-17
C++基于FFmpeg的**音频解码**和**播放**，使用VS2010编码，本程序实现了**音频的解码和播放**。可供学习和参考

使用ffmpeg解码视频渲染到sdl窗口 07-22
使用ffmpeg**解码视频**并渲染**视频**到窗口，网上是有不少例子的，但是大部分例子的细节都不是很完善，比如资源释放、flush**解码**缓存、多线程优化等都没...

FFmpeg时间戳 TyearLin的专栏 191
例如，如果一个**视频帧**的dts是40，pts是160，其time_base是1/1000秒，那么可以计算出此**视频帧**的**解码**时刻是40毫秒(40/1000)，显示时刻是160毫秒(1...

ffmpeg qq_42331499的博文 2525
1 简单使用需要现在ffmpeg，然后配置环境变量 2 基本命令行参数介绍 -re 使用正常速度推流 -stream_loop -1 循环推流 -vcodec copy或者libx264 **视频**转...

FFmpeg转码一帧(时间戳) 好过天的专栏 2101
写在前面 **时间戳** 时间基 帧 包 I B P SPS PPS 场 封装格式 编码格式 音频 **视频** 如果上面的这些关键字有些不了解的需要先查资料弄懂 **ffmpeg**转码一帧 转...

通过时间格式的字符串解析时间格式 小飞的博文 343
话不多说，在工作时有一个需求要求我通过一个时间格式的字符串解析出时间格式，然后我在网上找找看有没有什么工具可以用，但是也没发现什么这类...

ffmpeg监控推流带时间戳 08-01
要实现FFmpeg**监控推流带时间戳**，可以按照以下步骤进行操作。首先，需要安装FFmpeg，在命令行中输入相关命令安装程序，并确保可以在系统上正...

“相关推荐”对你有帮助？
😞 非常没帮助 😐 没帮助 😐 一般 😊 有帮助 😄 非常有帮助

关于我们 招贤纳士 商务合作 寻求报道 ☎ 400-660-0108 ✉ kefu@csdn.net 🗨 在线客服 工作时间 8:30-22:00
公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心 家长监护 网络110报警服务
中国互联网举报中心 Chrome商店下载 账号管理规范 版权与免责声明 版权申诉 出版物许可证 营业执照 © 1999-2023北京创新乐知网络技术有限公司

泡沫o0 已关注 1 6 0 专栏目录

泡沫o0

码龄9年

C/C++ 领域优质...

1392

316

933

80万+

原创

总排名

总排名

访问

等级

1万+

3万+

745

175

2836

积分

粉丝

获赞

评论

收藏

私信

已关注

发布首篇原创文章，
原力分+10分，点亮新秀勋章

去发布

搜博主文章

热门文章

WPA_CLI 的介绍：介绍如何使用wpa_cli连接WiFi的方法 10567

【研究Qt webengine 模块编译】linux 交叉编译qt5.12的webengine模块成功的条件 9833

WOL (Wake-On-LAN) 网络唤醒介绍 7890

【Linux C/C++ 延时(延迟)函数比较】介绍Linux系统中常用的延时函数sleep、usleep、nanosleep、select和std::sleep_for()的区别和使用场景 7794

【C/C++ 实用工具】CppCheck：静态代码检测工具，让你的代码更安全 7442

最新评论

【Linux系统编程】Linux 文件系统探究：... 隐→尘烟_Li：你这个排版，是markdown吗？挺好看的

C++ Qt 项目设计：跨平台的文件与视频... 泡沫o0：没有。。平时忙，没时间做，只提供设计思路。

C++ Qt 项目设计：跨平台的文件与视频... 右大臣：请问有没有源码可以学习一下呢...

【C++ 继承】C++继承解密：一步步引领... zwh1298454060：像书一样

【C++ 继承】C++继承解密：一步步引领... zwh1298454060：卧槽，好牛啊

您愿意向朋友推荐“博客详情页”吗？

😞 强烈不推荐

😐 不推荐

😐 一般般

😊 推荐

😄 强烈推荐

最新文章

【职场面试 情商篇】从心理学角度 解析如何回答‘与其它部门合作沟通不一致的问题’？

【职场面试 情商篇】从心理学角度 解析如何回答‘过去工作经历中遇到的挑战’？

【C/C++ 基础知识】深入C++：特殊成员函数的底层原理与规则

2023

10月

09月

08月

07月

122篇

171篇

201篇

55篇

06月

05月

04月

03月

470篇

100篇

144篇

52篇

02月

01月

3篇

2篇

2022年

86篇

2021年

16篇

2020年

13篇

👤 用户头像

📧 消息

📖 目录

📄 文档

📝 笔记

🔍 搜索

🔊 语音

📢 举报

⬆️ 返回