RAPID：面向高效长视频理解的编解码器感知视频解码

Yuna Lee 成均馆大学电气与计算机工程系
yuna6548@skku.edu

Yulhwa Kim*成均馆大学半导体系统工程系
yulhwakim@skku.edu

arXiv:2611.30557v1 [cs.CV] 23 Nov 2026

摘要

近期的视觉语言模型（*VLMs*）能够处理一小时长视频，但大量输入帧会带来显著的推理开销。现有的高效长视频理解方法大多忽视了视频解码，而解码耗时可能超过大语言模型预填充。我们提出 *RAPID*（参考帧与访问点感知解码），这是一种编解码器感知框架，利用压缩比特流中嵌入的信息，同时降低帧解码和帧选择的成本。*RAPID* 包含两个组件。*RAPID-Dec* 利用参考帧信息和随机访问点，跳过不必要的帧，并行解码保留帧，同时确保目标帧的精确重建。在 *H.264/AVC* 视频上，与最快的基线方法相比，其解码加速比最高可达 *3.7*×。*RAPID-Sel* 将关键帧用作视频片段的粗略表示，避免了现有帧选择方法所需的密集候选帧解码与评分。其准确率与现有方法相当，预处理加速比最高可达 *3.0*×。结合起来，与采用最快基线解码器的先进帧选择方法相比，*RAPID* 在一小时长视频上的首 *token* 时间（*TTFT*）加速比达到 *4.9*× 。

1. 引言

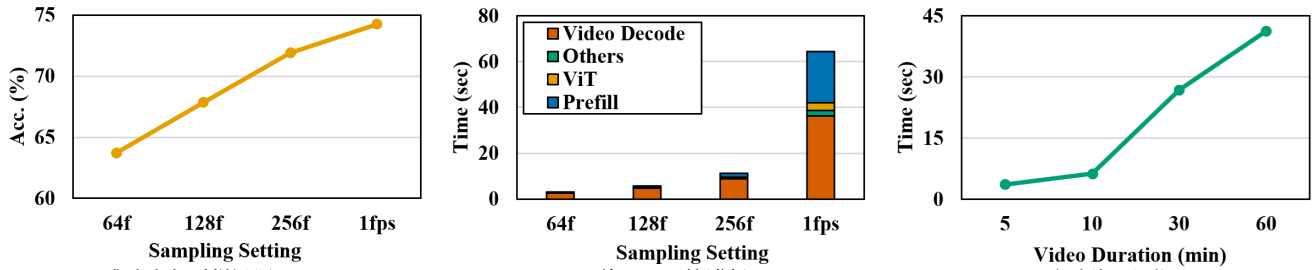
视觉语言模型（视觉语言模型）发展迅速，近期模型的应用范围已从图像和短片扩展到长视频 [1, 2, 10, 32]。随着上下文长度的增加，这些模型可以输入更多帧，从而提高其对长视频的准确率，如图所示。1a。然而，处理长视频会产生巨大的推理开销，其中包括视频解码，即将压缩比特流转换为 RGB 帧。

图 1b 展示了在不同帧采样设置下，Qwen3-VL-8B 处理一小时长视频时的首 token 时间（首 token 时间）分解情况。在所有设置中，视频解码所需时间甚至超过了大语言模型预填充，近期一项研究也报告了这一点 [44]。此外，现有的帧选择方法会先解码并评估密集候选帧，这些帧通常以每秒 1 帧（帧/秒）的速度采样 [5, 27], ，然后再为视觉语言模型选择少量帧。这进一步增

会增加预处理延迟，如图1c所示。现有的视频解码加速器可在 CPU核心 或 GPU [23, 41], 之间并行执行解码，但仍会重建 VLM 从未使用的帧。因此，高效的长视频理解需要同时降低获取输入 VLM 的帧以及选择这些帧的解码成本。

我们发现，压缩视频比特流包含有关帧依赖关系和随机

*通讯作者



(a) 准确率与采样设置　(b) 首 token 时间分解　(c) 1 fps 视频解码延迟

图 1. RAPID 的动机。(a) 不同帧采样设置下 Qwen3-VL-8B 在 MLVU 上的准确率。(b) Qwen3-VL-8B 在不同采样设置下的首 token 时间分解。(c) 不同视频时长下，在不设置帧数上限的情况下以 1 fps 进行视频解码的延迟。在 (a) 和 (b) 中，f 表示均匀采样帧的数量（64、128 或 256），而 fps 表示每秒帧数；对于 1 fps 设置，采样帧数上限为 2,048 帧。所有视频均使用 TorchCodec [29] 进行解码。

访问点的信息，无需解码帧即可获取。此外，视频编码器可以在场景切换处放置关键帧，以提高压缩效率。[18, 30]这些特性为加速解码和帧选择提供了机会，且无需修改视觉语言模型。基于这些观察，我们提出了 RAPID，这是一种编解码器感知框架，由用于解码目标帧的 RAPID-Dec 和用于选择目标帧的 RAPID-Sel 组成。

我们的贡献总结如下：

- 我们提出 RAPID-Dec，它利用比特流中的信息跳过不必要的帧，并并行解码保留帧，无需近似处理；在 H.264/AVC 视频上，其视频解码速度比最快的基线解码器提升了 3.7× 倍。
- 我们提出 RAPID-Sel，将关键帧作为视频片段的粗略代表，以选择查询相关帧。RAPID-Sel 只解码并评估关键帧，而非密集候选帧，在预处理速度提升最高达 3.0× 倍的同时，实现了相当的准确率。
- 我们将这两个组件结合为 RAPID。与采用最快基线解码器的最先进帧选择方法相比，RAPID 在一小时长视频上的首 token 时间提升了 4.9× 倍。

2. 相关工作

2.1. 高效视频解码

QuickCodec [23] 和 FlashCodec [41] 通过将视频划分为多个片段并行解码，加速视觉语言模型的解码。QuickCodec 将每个片段分配给一个 CPU 核心，而 FlashCodec 则将片段分发到多个 GPU 及其 NVDEC 硬件解码器上。然而，这两种方法都会解码既未被采样为视觉语言模型输入、也未被其他帧引用的帧。QuickCodec 会解码每一帧，不受采样间隔影响，因此其优势仅限于密集采样。FlashCodec 则受限于每个 GPU 上 NVDEC 引擎的数量，而且参考帧会占用与模型共享的 GPU 显存。在视频分析中，Smol [13] 解码低分辨率输入并在其上微调深度神经网络，而 CoVA [12]则利用编解码器元数据训练轻量级网络，以选择用于目标检测的帧。然而，这两种方法都依赖近似处理或特定任务的训练。相比之下，RAPID-Dec 无需特定任务训练即可跳过不必要的帧，同时保留精确重建。

2.2. 高效长视频理解

帧选择。 在帧预算有限的情况下，帧选择旨在比均匀采样取得更高的准确率。对于动作识别，SCSampler [14] 利用压缩域特征训练轻量级采样器，以选择显著片段。对于视觉语言模型，Hu 等人 [11] 和 Qin 等人 [19] 训练轻量级外部选择器，以找出查询相关帧，而 Frame-Voyager [39] 则训练一个附加于视觉语言模型的插件模块。然而，这些方法需要带有帧选择监督信号的视频训练数据。相比之下，免训练方法使用预训练轻量级视觉语言模型 [16, 20]为帧打分。AKS [27] 和 WFS-SB [5] 对以 1帧/秒采样的候选帧进行打分。AKS兼顾查询相关性和时间覆盖，而 WFS-SB则将帧预算分配到不同语义片段中。Focus [43]先识别查询相关的时间区域，再从中选出得分最高的帧。然而，这些方法必须对远超最终帧预算数量的候选帧池进行解码和评分，从而带来巨大的预处理开销。

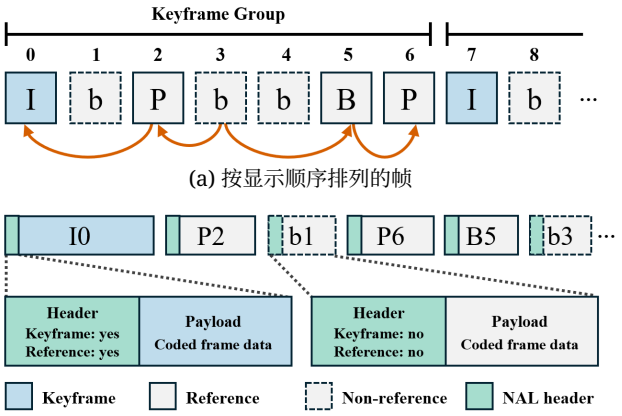
词元压缩和基于编解码器的表示。词元压缩方法会减少帧内或帧间的视觉令牌 [4, 24, 26, 28, 31, 37, 45], ，但它们处理的是解码帧，因此无法降低视频解码延迟。其他方法则利用压缩域表示，减少对 RGB 帧的依赖。CoPE- VideoLM [21] 和 ReMoRa [38]使用编解码器生成的信息（例如运动矢量和残差）替代大多数 RGB 帧，而 CoViAR [35] 则探索了在动作识别中采用类似方法。与 RAPID 不同，这些方法需要使用压缩域输入进行训练。

3. 方法

3.1. 预备知识：视频编解码器

视频编解码器由编码器和解码器组成。编码器将视频压缩为紧凑的比特流，其语法由 H.264/AVC [34] 或 H.265/HEVC [25], 等编码标准定义；当视频呈现给用户或视觉语言模型时，解码器会根据比特流重建帧。对于视觉语言模型，重建后的帧还会进一步转换为 RGB 图像。现有研究利用反映视觉内容的压缩域线索（例如运动矢量和残差），以实现高效的视频理解 [21, 38, 45]。除了这些内容线索，我们还利用比特流语法所揭示的帧依赖关系和随机访问点，跳过不必要的解码并指导帧选择。

编解码器中的帧类型。根据编码所采用的预测方法，帧可分为 I 帧、P帧和 B 帧。I 帧仅使用帧内预测进行编码，因此可以独立于其他帧进行解码。相比之下，P帧和 B 帧还会利用帧间预测。P帧根据之前已解码的帧进行预测，而 B 帧则可以按照显示顺序，根据前后帧进行双向预测。如图 2a所示，第 0 帧是 I 帧，可以独立解码；但第 3 帧只有在第 2 帧和第 5 帧解码后才能解码。由于这种依赖关系，即使只获取单个帧，也需要解码它所依赖的所有



(a) 按显示顺序排列的帧

(b) 按解码顺序排列的比特流

图2. 关键帧组示例。(a) 按显示顺序排列时，每个箭头都从一帧指向其所引用的帧。(b) 在比特流中，帧按解码顺序存储；比特流由 NAL 单元构成，每个 NAL 单元都包含头部和载荷。

帧，因此，在从长视频中稀疏采样帧的视觉语言模型处理流程中，视频解码成为瓶颈。

一些 I 帧还可作为随机访问点（RAP），解码器可从这些点开始解码，而无需任何前序帧。尽管名称各异，所有主流视频编解码器都提供 RAP，以支持诸如查找等功能 [9, 17, 25, 33,34]。H.264/AVC 定义了即时解码刷新（IDR）帧 [34], ，而 H.265/HEVC 定义了帧内随机访问点帧，包括 IDR、干净随机访问帧和断链访问帧 [25]。在本文中，如果比特流中没有后续帧引用该帧之前的任何帧，我们就将 RAP 称为关键帧。场景切换后的帧与切换前的帧内容相似度很低，因此跨场景切换进行帧间预测效率不高 [15, 22]。因此，广泛使用的编码器 [18, 30]默认会在检测到场景切换时插入关键帧。

由于 RAPID 以关键帧为粒度访问比特流，我们将关键帧组定义为从一个关键帧开始，到下一关键帧之前一帧为止的一系列帧，如图2a所示。每个关键帧组都始于一个关键帧，因此可以独立解码。

比特流语法结构。如图2b所示，H.264/AVC 和 H.265/HEVC 等标准编解码器的比特流由一系列网络抽象层（NAL）单元组成。每个 NAL 单元都包含一个头部和一个载荷。头部指示载荷的类型以及帧级属性，例如该帧是否为关键帧，以及是否可被其他帧引用；载荷则承载编码数据。由于头部是位于每个 NAL 单元起始处的定长字段，因此只需解析少量字节即可获取这些信息，而无需解码该帧。

3.2. RAPID 概述

RAPID 由两个组件组成：RAPID-Dec 和 RAPID-Sel，如图3所示。RAPID-Dec 通过在解码保留帧之前，从比特流中过滤掉不必要的帧，加速采用任意帧采样策略的视频解码。目标帧可以通过均匀采样、现有帧选择方法或 RAPID-Sel 选出。RAPID-Sel 利用关键帧粗略识别查询相关场景；与均匀采样相比，它能够选出更多与查询相关的目标帧，同时无需解码密集候选帧。

3.3. RAPID-Dec

用于播放的解码器必须重建每一帧，因为每一帧都会被显示。相比之下，视觉语言模型只使用其采样策略选出的帧。其他帧重建后会被丢弃，或仅作为所选帧的参考。因此，视觉语言模型流水线的视频解码延迟并非由模型使用的帧数决定，而是取决于解码器为获取这些帧必须重建多少帧。RAPID-Dec 利用比特流的语法缩小这一差距。给定目标帧的索引后，RAPID-Dec 会确定哪些帧需要保留以实现精确重建，仅将这些帧的比特流交给解码器，并行解码这些帧。解码得到的目标帧与解码整个视频所得到的帧完全相同。

保留帧。 RAPID-Dec 首先扫描一次比特流，但不解码任何帧，只解析每帧中的少量简短语法元素。此次扫描会确定关键帧的位置，从而确定各组的位置，同时获取每帧的参考标志。我们将无法被引用的帧称为非参考帧。

令 $\mathcal{T}$ 为目标帧集合，G_g 为第 g 组，其关键帧为 k_g。对于至少包含一个目标帧的每一组，令 ℓ_g 为该组中解码顺序上的最后一个目标帧。RAPID-Dec 将 G_g 中需要保留的帧定义为

$$\mathcal{E}_g = \{\, f \in G_g : f \preceq \ell_g,\ \mathrm{ref}(f) \vee f \in \mathcal{T} \cup \{k_g\} \,\}, \quad (1)$$

其中，$f \preceq \ell_g$ 表示在解码顺序中，f 先于或等于 ℓ_g；ref(f) 是 f的 NAL 单元头中的参考标志。因此，在比特流到达解码器之前，会移除三类帧：(i) 不包含目标帧的组中的所有帧；(ii) 解码顺序中每组最后一个目标帧之后的帧，因为没有更早的帧会引用这些帧（每帧都存储在其引用的所有帧之后）；(iii) 不是目标帧的非参考帧。由于剩余帧都不会引用已移除的帧，因此每个目标帧都能像完整视频解码一样精确重建，无需近似处理。请注意，

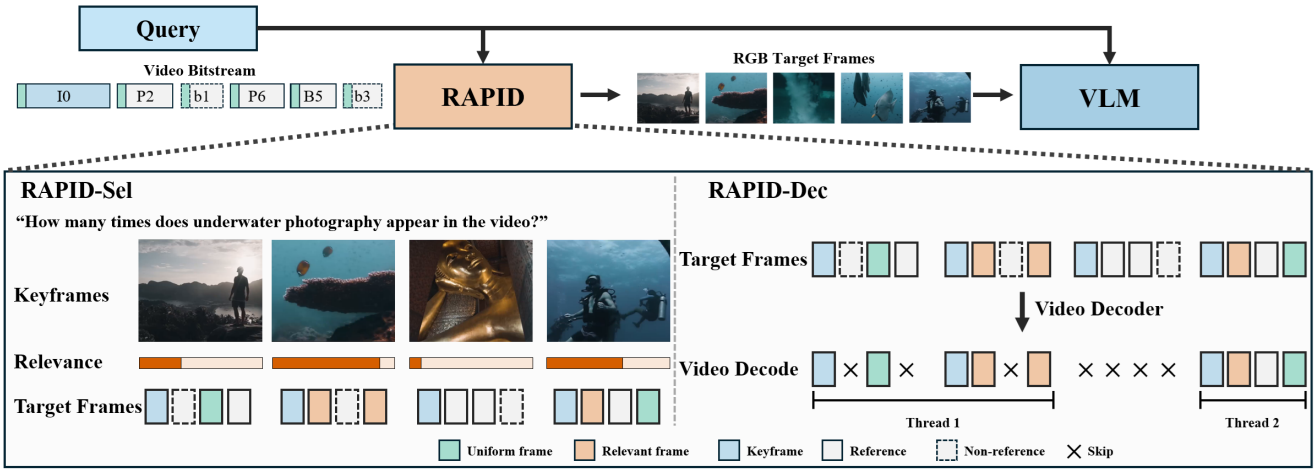


图 3. RAPID 概览。RAPID-Dec（右）对由任意采样策略（包括 RAPID-Sel（左））选出的目标帧进行解码。它会跳过不含目标帧的组、每组最后一个目标帧之后的帧，以及除目标帧之外的非参考帧，同时保持目标帧的精确重建。随后，保留的组会分配到各个线程中进行并行解码。RAPID-Sel 仅对关键帧进行解码，并针对查询对其评分；然后结合均匀采样帧与按相关性分配给各组的相关帧，选择目标帧。

T表 1. 每个视频中非参考帧的平均比例 eo.

Video-MME	LongVideoBench	MLVU
52.7%	51.8%	37.5%

(ii) 依据的是解码顺序而非显示顺序，因为某帧可能会引用显示在它之后的帧。例如，图2a 中的帧 3 引用了帧 5，而帧 5 又引用了帧 6。虽然帧 5 和帧 6 都显示在帧 3 之后，但它们存储在帧 3 之前（图2b）。通过保守地保留可能需要的参考帧，RAPID-Dec 无需进行繁重的显式依赖关系追踪，同时保证目标帧的精确重建。

随着采样变得更加稀疏，(i) 和 (ii) 带来的节省会增加；而 (iii) 即使在密集采样时也适用，因为此时几乎每个组都包含一个目标。为了估算 (iii) 的潜在节省空间，我们测量了 Video-MME 中每个视频的非参考帧比例 [8]，LongVideoBench [36], 和 MLVU [42]。如表 1所示，非参考帧平均占总帧数的 37.5–52.7%，这表明跳过这些帧可以节省大量开销。

FFmpeg [7] 也可以通过其`skipframe` 选项跳过非参考帧，但它无法选择要跳过哪些帧，因此也会跳过作为目标的非参考帧。RAPID-Dec 则只移除并非目标的非参考帧，如式 (1) 所示。

负载均衡并行解码。 随后并行解码筛选后的关键帧组。给定线程预算 T，设 $\mathcal{G}$ 为包含目标的组集合，$N = \min(T, |\mathcal{G}|)$。一种简单的并行解码方法是将 $\mathcal{G}$ 划分为 N 个包含相同数量组的连续片段，并让每个工作线程解码各自的片段。然而，各组的解码成本并不相同。关键帧组的长度可能不同，而且移除最后一个目标之后的帧后，组的解码成本还取决于其最后一个目标的位置。因此，包含相同数量组的片段可能导致各工作线程的负载不均，延迟则由最慢的工作线程决定。RAPID-Dec 改为用每组保留帧的数量 $w_g = |\mathcal{E}_g|$来衡量其工作负载。它将 $\mathcal{G}$ 划分为至多 N 个总工作负载大致相等的连续片段，在累计工作负载达到 $\frac{1}{N}\sum_{g\in\mathcal{G}} w_g$ 的整数倍处进行切分，并将每个片段分配给一个工作线程。当由于只有少数组包含目标而创建的工作线程少于 T 个时，其余线程将用于各解码器内部的多线程处理，因此线程总数不超过 T。

每个工作线程都拥有一个解复用器和一个解码器。它按顺序访问其片段中的各组，直接跳转到每个关键帧，并仅将保留帧输入其解码器（算法 1）。每个工作线程只将解码后的目标帧转换为 RGB，并直接写入输出中的对应位置，因此后处理也实现了并行化。

3.4. RAPID-Sel

RAPID-Dec 可加速任意目标帧的视频解码，但视觉语言模型的准确率取决于所选帧。若不进行帧选择，通常会均匀采样目标帧；当帧预算不足以覆盖长视频时，这可能会漏掉查询相关场景。查询感知帧选择方法 [5, 27]会先解码密集候选帧，通常以 1帧/秒的速度解码，再使用轻量级视觉语言模型 [16, 20] 根据查询对候选帧进行评分，从而选出目标帧。然而，在长视频中，候选帧数量远超帧预算，解码这些候选帧会使选择过程变慢。RAPID-

算法 1 RAPID-Dec

要求： 比特流 $\mathcal{B}$、目标帧 $\mathcal{T}$、线程预算 T
确保： RGB 中的目标帧 $\{x_t\}_{t\in\mathcal{T}}$
1：扫描 $\mathcal{B}$ 而不解码 $\rightarrow$ 关键帧组 $\{G_g\}$ 包含关键帧 k_g、参考标志
2：$\mathcal{G} \leftarrow \{g : G_g \cap \mathcal{T} \neq \emptyset\}$，依次
3: **对于所有** $g \in \mathcal{G}$ **执行**
4: $\ell_g \leftarrow$ 解码顺序中的最后一个目标 G_g
5: $\mathcal{E}_g \leftarrow$公式（1）
6: $w_g \leftarrow |\mathcal{E}_g|$
7: **结束循环**
8: $N \leftarrow \min(T, |\mathcal{G}|)$
9：将 $\mathcal{G}$ 划分为 $M \le N$ 个连续片段 $\{S_i\}$，其中 $\sum_{g\in S_i} w_g \approx \frac{1}{N}\sum_{g\in\mathcal{G}} w_g$
10: **对所有** S_i **并行执行**
11: 打开一个使用 $\lfloor T/M \rfloor$ 个线程的解码器
12: **对所有** $g \in S_i$ **执行**
13: 跳转到 k_g 并解码 $\mathcal{E}_g$
14: 将解码后的 RGB 目标写入输出
15: **结束循环**
16: **结束循环**

Sel 旨在以最小的延迟开销提升准确率，胜过均匀采样。

由于关键帧通常插入在场景切换处（第 3.1 节），关键帧 k_g 可以作为内容的粗略代 G_g表。此外，关键帧不引用其他帧，因此可以并行解码。RAPID-Sel 因此仅解码 K组的关键帧，并测量每个组 G_g 相对于文本查询 q 的相关性，所依据的是图文评分器（如 CLIP）输出的匹配 logit，

$$s_g = \tau \cos\big(\phi_I(k_g),\, \phi_T(q)\big), \tag{2}$$

其中，ϕ_I 和 ϕ_T 分别是评分器的图像编码器和文本编码器，τ 是其 logit 缩放系数。

给定帧预算 B 和比例 $\rho \in [0,1]$，RAPID-Sel 将其划分为均匀预算 $B_u = \text{round}(\rho B)$ 和相关性预算 $B_r = B - B_u$。均匀采样帧 B_u 在整个视频中均匀采样，用于保留针对整个视频的查询所需的时间覆盖范围。相关帧 B_r则根据由 logits 给出的各组概率，逐一分配给组 g^*:

$$p_g = \frac{e^{s_g}}{\sum_{h=1}^{K} e^{s_h}}, \qquad g^* = \arg\max_g \frac{p_g}{c_g + 1}, \tag{3}$$

其中，p_g 是 k_g 在K 关键帧中与 q 匹配的概率，c_g 是已分配给 G_g的相关帧数。该规则称为德洪特法 [3]，，按大致比例分配帧至 p_g。由于组的概率会根据其已拥有的帧数进行折减，只有当某一组的概率占据主导时，帧才会集中分配给该组；否则，帧会分散到多个组中。当某组在其持续时间内每秒拥有一个选定帧时，该组即为满组，因为间隔不到一秒的帧很可能是冗余的。满组会从公式 最大值函数（3）中的选择中排除。

最后，通过最小化连续选中帧之间间隔的平方和，将 c_g 相关帧均匀地分布在各组中。选中的帧构成 RAPID-Dec 的目标集 $\mathcal{T}$，其中的关键帧沿用评分步骤中选出的关键帧。不使用 RAPID-Sel 时，$\mathcal{T}$由均匀采样或现有帧选择方法确定。

4. 实验

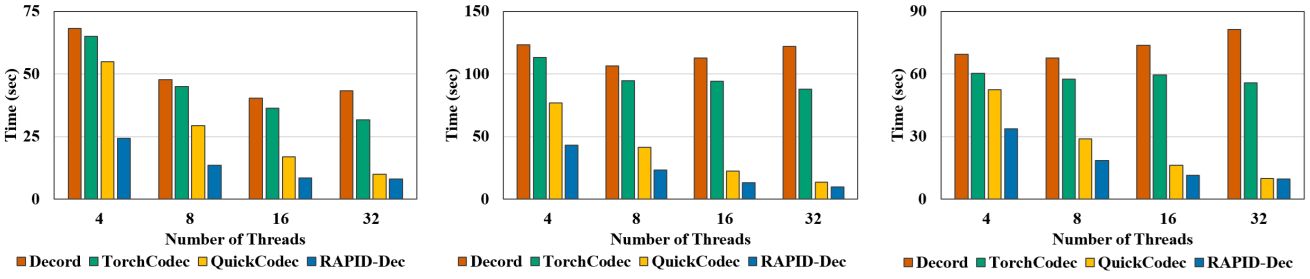
4.1. 实验设置

环境。 我们在 Intel Xeon Gold 6530 CPU 上测量视频解码延迟，处理器时钟频率固定为 2.1 GHz；视觉语言模型和 CLIP 评分器则运行在 NVIDIA RTX 5880 Ada GPU 上。除非另有说明，线程预算设为 16，且每个解码器分配相同数量的线程。报告的延迟涵盖从打开视频文件到获取 RGB 格式目标帧的整个过程，包括 RAPID-Dec 的码流扫描。当以 1帧/秒采样帧作为视觉语言模型的输入时，我们按照 Qwen3-VL [1], 的评估设置，将帧数上限设为 2,048；而用于帧选择的候选帧则以 1帧/秒解码，不设上限。所有准确率评估均使用 lmms-eval 框架 [40]进行。RAPID-Sel 的比率 ρ 默认设为 0.5。

基线方法。 在视频解码方面，我们将 RAPID-Dec 与三种解码后端进行比较：Decord [6], TorchCodec [29], 和 QuickCodec [23]。Decord 和 TorchCodec 是视觉语言模型评估中广泛使用的视频加载库，而 QuickCodec 则旨在加速密集帧采样。

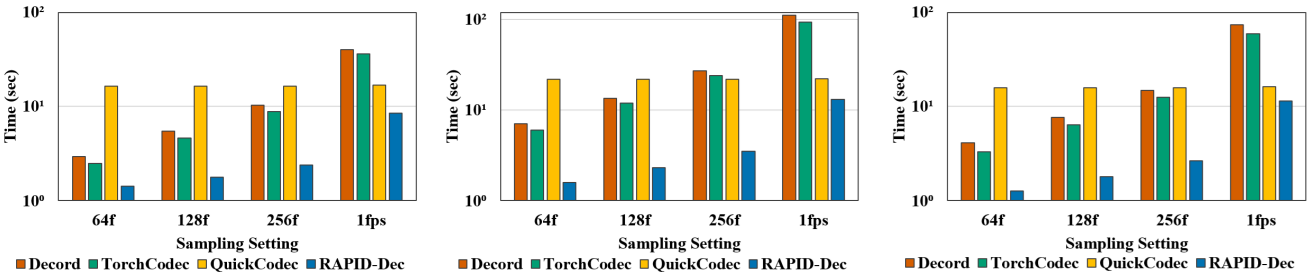
在帧选择方面，我们将 RAPID-Sel 与均匀采样、Top-K 选择、AKS [27], Focus [43],和 WFS-SB [5]进行比较。Top-K、AKS 和 WFS-SB 会对以 1帧/秒采样的候选帧进行评分，而 Top-K 只选择相关性分数最高的帧。相比之下，Focus 采用由粗到细的两阶段选择策略。

模型与基准测试。我们在三个长视频理解基准测试上评估 Qwen3-VL-8B [1]和 GLM-4.6V-Flash [10]: Video-MME [8], 、LongVideoBench (LVB) [36], 和 MLVU [42], 。使用的数据集分别为不含字幕的 Video-MME、LongVideoBench 的验证集以及 MLVU 的开发集。对于每种帧选择方法，我们使用 CLIP 视觉 Transformer-B/32 [20] 计算查询与候选帧之间的相关性。



(a) H.264/AVC (b) H.265/HEVC (c) VP9

图 4. 不同线程数下，1帧/秒采样（最多 2,048 帧）的视频解码延迟。



(a) H.264/AVC (b) H.265/HEVC (c) VP9

图 5. 16个线程下不同帧采样设置的解码延迟。纵轴采用对数刻度。

4.2. RAPID-Dec 的视频解码效率

首先，我们单独评估 RAPID-Dec，在不同预算下均匀采样目标帧，以便将视频解码与目标帧的选择分离开来。

并行性的影响。图4 展示了在来自 Video-MME 的一小时长视频上，使用不同线程数进行 1帧/秒采样时的视频解码延迟 [8]。为评估 RAPID-Dec 的通用性，我们将评估扩展至 H.265/HEVC（一种较新的标准编解码器）和 VP9 [17],（一种开源视频编解码器），两者均由原始 H.264/AVC 视频重新编码得到。每项结果均基于四个一小时长视频、每个视频运行五次后取平均值。对于每种编解码器和线程数，RAPID-Dec 都实现了最低延迟。在默认的 16个线程下，RAPID-Dec 比两个广泛使用的库中速度更快的 TorchCodec 快 43×、72×和 52× 倍，比 QuickCodec 在 H.264/AVC、H.265/HEVC 和 VP9 上分别快 20×、17×和 14× 倍。Decord 和 TorchCodec 的扩展性较差，可能是因为它们的线程仅在单个解码器内部进行并行，而帧间依赖可能限制了并行性。

帧采样策略的影响。图5 在相同视频上改变采样设置。对于三种编解码器，RAPID-Dec 在每种设置下再次实现了最低延迟。QuickCodec 会解码每一帧，因此无论采样帧数量如何，其延迟都基本保持不变。因此，它在密集采样下表现良好，但随着采样变得更加稀疏，其与 RAPID-Dec 之间的差距也会扩大。其他解码器的延迟会随着采样帧数量的增加而上升，但 RAPID-Dec 的增长速度更慢。因而，随着采样变得更加密集，RAPID-Dec 相比 TorchCodec 的加速比也随之提高：在 H.264/AVC 上从 1.7× 提升至 4.3×，在 H.265/HEVC 上从 3.8× 提升至 7.2×，在 VP9 上从 2.6× 提升至 5.2×。相较于每种设置下速度最快的基线方法，在 H.264/AVC、H.265/HEVC 和 VP9 上，加速比分别最高可达 3.7×、6.3× 和 4.7×。

每个组件的影响。我们在图5a 所示的设置下，使用表2中的64帧和1帧/秒采样，分析每个组件的贡献。我们首先仅采用并行解码，在具有相同组数的连续片段中解码每一帧，然后逐步加入RAPID-Dec的各个组件。最有效的组件取决于采样设置。在64帧的情况下，最多有64组包含目标，因此跳组带来的加速比最大（7。23×）。在1帧/秒下，几乎每组都包含一个目标，而

表2. RAPID-Dec 的消融实验。图5a中视频的解码延迟（延迟）。各组件依次累加，加速比相对于上一行计算。

	64帧		1帧/秒	
配置	延迟（秒）	加速比	延迟（秒）	加速比
仅并行解码	15.91	–	16.38	–
+ 跳组	2.20	7.23×	15.94	1.03×
+ 截断	1.84	1.20×	13.03	1.22×
+工作负载均衡	1.77	1.04×	12.90	1.01×
+非引用跳过（完整）	**1.45**	1.22×	**8.72**	1.48×

表3。Video-MME、MLVU 和 LongVideoBench（LVB）上的准确率（%）与预处理延迟。预处理指 RAPID-Dec 解码和 CLIP ViT-B/32 评分的每视频延迟，该数值是在每个基准数据集 10% 的视频上运行三次后取平均得到的。

			Qwen3-VL-8B				GLM-4.6V-Flash			
方法	#帧数	预处理（秒）↓	Video-MME	MLVU	LVB	Avg.	Video-MME	MLVU	LVB	Avg.
均匀	32	0.60	60.26	60.69	55.95	58.97	57.15	58.68	55.27	57.03
Top-K	32	3.41	61.56	68.81	57.89	62.75	59.70	67.08	60.13	62.30
AKS	32	3.41	61.44	65.84	56.99	61.42	59.26	61.70	57.67	59.54
Focus	32	2.36	61.04	67.40	58.27	62.24	58.30	64.76	60.21	61.09
WFS-SB	32	3.41	62.78	66.81	58.26	62.62	60.93	64.33	58.49	61.25
RAPID-Sel	32	1.12	61.48	67.71	57.67	62.29	60.30	65.65	59.54	61.83
均匀	64	0.83	63.63	63.72	58.86	62.07	59.96	62.14	58.34	60.15
Top-K	64	3.41	63.81	70.50	59.91	64.74	62.63	68.05	61.78	64.15
AKS	64	3.41	65.70	69.93	60.36	65.33	62.41	66.19	60.81	63.14
Focus	64	2.39	64.26	70.07	60.73	65.02	61.04	66.58	62.08	63.23
WFS-SB	64	3.41	65.26	69.89	59.91	65.02	62.93	66.22	60.36	63.17
RAPID-Sel	64	1.30	65.41	70.05	60.58	65.35	62.44	66.71	61.71	63.62

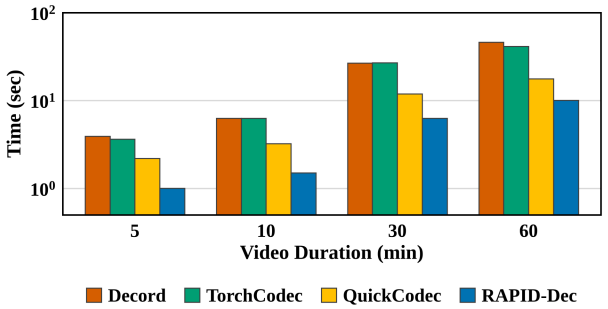


图 6. 使用16个线程对 Video-MME 中 H.264/AVC 视频以1帧/秒采样的帧进行解码的延迟。

非参考帧跳过带来的加速比最大（1。48×），因为它是唯一一个能够移除每组最后一个目标之前的帧的组件。因此，跳组对于稀疏采样最为有效，而非参考帧跳过对于密集采样最为有效；不过，在这两种设置下，每个组件都会降低延迟，使完整的RAPID-Dec在64帧和1帧/秒下分别比单独使用并行解码快11。0×和1。88×。

跨视频时长的候选采样。许多帧选择方法以 1帧/秒的速度解码候选帧，因此候选项数量会随视频时长线性增长。图6 展示了时长为 5 至 60 分钟的 H.264/AVC 视频的解码延迟。RAPID-Dec 在所有时长下都实现了最低延迟，相较于 TorchCodec 和 QuickCodec 的加速比分别为 3.6–4。3×和 1.8–2。2×对于时长达一小时的视频，RAPID-Dec 耗时 9.8 秒，而 TorchCodec 和 Decord 分别耗时 40.5 秒和 45.1 秒。由于 RAPID-Dec 不受目标帧选择的影响，现有帧选择方法可以直接从中受益。

4.3. RAPID 的准确率和效率 Sel

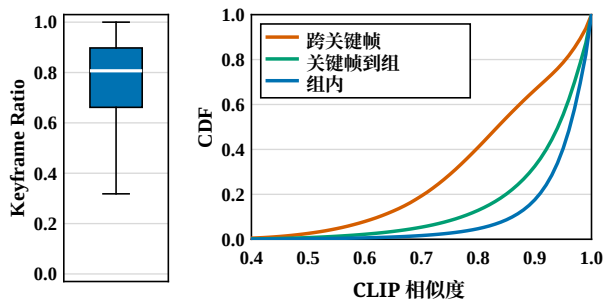
接下来，我们在相同帧预算下，将 RAPID-Sel 与现有帧选择方法进行比较；所有方法均使用 RAPID-Dec，以便将帧选择与解码器选择的影响区分开来。

主要结果。 如表 3所示，与均匀采样相比，所有帧选择方法都提高了平均准确率，这证实了面向查询的帧选择对长视频的益处。使用 32 帧和 64 帧时，RAPID-Sel 在 Qwen3-VL-8B 上分别提升 3.32 和 3.28 个百分点，在 GLM-4.6V-Flash 上分别提升 4.80 和 3.47 个百分点。使用 Qwen3-VL-8B 和 64 帧时，RAPID-Sel 的平均准确率最高；在其他三种设置下，其准确率也与最佳方法相差不超过 0.53 个百分点。同时，使用 32 帧和 64 帧时，RAPID-Sel 的预处理耗时仅为 1.12 秒和 1.30 秒，比 Top-K、AKS 和 WFS-SB 快 3。0× 和 2。6× 倍，比 Focus 分别快 2。1× 和 1。8× 倍。这仅比均匀采样多耗时约 0.5 秒。由于所有方法都使用 RAPID-Dec，因此这种加速得益于只对关键帧而非密集候选帧进行解码和评分。

关键帧候选项的影响。为评估基于关键帧的选择方式所带来的收益，我们将每个视频的关键帧替换为数量相同的均匀间隔帧，将每个候选项到下一个候选项之间的帧视为其组，并保持 RAPID-Sel 的其余部分不变。如表 4所示，关键帧候选项取得了略高的平均值

表4. 在使用 Qwen3-VL-8B 和 64帧的 Video-MME 上，RAPID-Sel 中数量相等的均匀候选项与关键帧候选项对比。

候选项	短	中	长	Avg.
均匀	**76.22**	63.67	55.22	65.04
关键帧	75.67	**63.78**	**56.78**	**65.41**



(a) 位置 (b) 相似度

图 7. Video-MME、LongVideoBench 和 MLVU 共 2,098 个视频中的关键帧位置与视觉一致性。(a) 每个视频中包含关键帧的低相似度 1 秒区间所占比例（CLIP 相似度低于 0.73）。(b) 组内相邻 1帧/秒的视频帧之间、跨越关键帧的视频帧之间，以及每个关键帧与其所在组其他视频帧之间的 CLIP 相似度累积分布函数。

准确率（65.41 对 65.04），且差异从短视频中的 -0.55个百分点扩大到长视频中的 $+1.56$个百分点。这表明，基于关键帧的候选项对长视频尤其有益，因为稀疏且均匀的候选项可能会错过相关的视觉变化。

关键帧与视觉一致性。我们使用三个基准测试中 1帧/秒的视频帧的 CLIP ViT-B/32 嵌入，考察关键帧与视觉变化之间的关系。对于每个视频，我们计算其 1帧/秒的视频帧两两之间的余弦相似度，并取中位数，以此表示视频中任意两个时刻的相似程度。所有视频的该值中位数为 0.73，因此，我们将相似度低于 0.73 的相邻 1帧/秒的视频帧之间的 1 秒区间定义为低相似度区间（占所有区间的 7.8%）。如图7a所示，每个视频中这些区间包含关键帧的比例中位数为 0.81，这表明关键帧位于大多数显著视觉变化处。

在图 7b中，跨越关键帧的相邻 1 帧/秒视频帧之间，相似度低于 0.73 的情况比同一组内更常见（24.7% 对 2.2% ），而同一组内有 82.4% 的帧对相似度超过 0.9。 $11\times$ 跨越关键帧的某些帧对仍然高度相似，这可能是因为编码器也会在组达到最大长度时插入关键帧。 [18, 30]为检验关键帧能否代表其所在的组，我们还将其与该组中的其他 1 帧/秒视频帧进行比较。这些相似度与组内帧之间的相似度相近，而低于跨越关键帧的相似度；这可能是因为它们跨越了更长的时间间隔，期间逐渐累积的视觉变化可能会造成影响。这些结果支持将关键帧用作其所在组的粗略代表。

影响 ρ。表5展示了不同均匀采样比例 ρ下 Video-MME 上的准确率。每个测试值均优于均匀采样（$\rho = 1$），平均而言，

表 5。 ρ的影响（Video-MME，Qwen3-VL-8B，64帧）。

ρ	短	中	Long	Avg.
0.45	75.44	63.67	55.22	64.78
0.50	75.67	**63.78**	**56.78**	**65.41**
0.55	75.22	63.22	56.11	64.85
0.60	**76.00**	61.89	55.22	64.37
统一（$\rho = 1$）	74.56	61.89	54.44	63.63

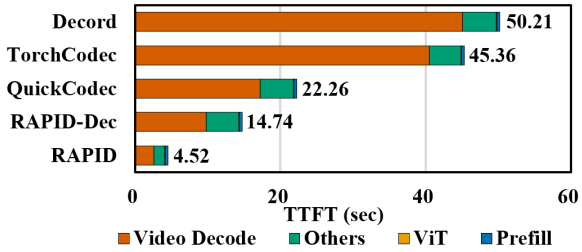


图 8. 在一小时长视频上使用64帧时，Qwen3-VL-8B的首 token 时间分解。前四个柱状条分别使用各解码器搭配AKS。其余柱状条还包括相关性评分和帧选择。

这表明，将部分帧预算分配给与查询相关的组是有益的。平均准确率在我们的默认设置 $\rho = 0.50$ 下达到峰值，比均匀采样高出 1.78 个百分点，这主要得益于中视频和长视频上的提升。当 ρ 从 0.60 降至 0.50 时，相关性预算增加，长视频上的准确率从 55.22 提升至 56.78。将 ρ 降至 0.45 会导致准确率下降，可能是因为剩余的均匀采样帧更少，无法覆盖整个视频。

4.4. RAPID 的端到端效率

图 8 展示了第4.2节一小时长视频上使用64帧时，Qwen3-VL-8B的首 token 时间分解，对比了RAPID与AKS；AKS是此前在Qwen3-VL-8B和64帧设置下精度最高的方法（表3）。使用AKS时，视频解码占据了首 token 时间的主要部分：使用TorchCodec时，45.4秒中有40.5秒用于视频解码。RAPID-Dec将其缩短至9.8秒，相比TorchCodec和QuickCodec分别实现了3。$1\times$ 和1。$5\times$ 倍的首 token 时间加速比，但以1帧/秒解码并对所有候选帧进行评分仍需14.3秒。RAPID-Sel仅解码并评估关键帧，将这项开销降至4.1秒。因此，RAPID将首 token 时间缩短至4.5秒，相比使用RAPID-3×、4。9×和10。$0\times$ 的AKS分别实现了加速比；其精度与AKS相当（65.35%对65.33%）。

5. 结论

我们提出了 RAPID，这是一种面向高效长视频理解的编解码器感知视频解码框架。我们证明，视频解码是首 token 时间的主要瓶颈，尤其对于需要解码密集候选帧的帧选择方法而言。RAPID 通过利用比特流

的两个组件实现这一点。对于任意目标，RAPID-Dec 利用关键帧和参考信息跳过不必要的帧，并在不进行近似的情况下并行解码其余帧，在 H.264/AVC 上的速度最高可达到最快基线的 3.7× 倍。RAPID-Sel 对关键帧而非密集候选帧进行评分，以选择与查询相关的帧，在准确率相当的情况下，预处理速度最高提升 3.0× 倍。结合这两个组件，RAPID 在长达数小时的视频上，相较于采用最快基线解码器的当前最先进帧选择方法，实现了首 token 时间 4.9× 倍的加速。

大语言模型使用声明

我们在准备本文时使用了 Claude（Anthropic）。本文由作者起草，并使用 Claude 润色语法、措辞和句子结构，包括相关工作和实验结果部分。我们还将完整的论文草稿提供给 Claude，请其以审稿人的视角提出反馈；作者独立评估了这些建议，并决定采纳哪些内容。Claude 协助检索相关先前研究，而作者对照原始出版物核实了所有引用文献，并最终决定引用内容。我们还使用 Claude 根据作者提供的规格生成了部分实现代码。作者对所有生成的代码进行了审查、测试，并根据需要加以修改。所有实验均由作者独立设计并开展，结果也由作者分析。Claude 未参与研究构想、假设、实质性主张或结论的形成，也未用于创建图表、示意图或其他视觉材料。合成数据生成、数据集创建或标注均未使用任何大语言模型工具。作者对本文的全部内容承担全部责任。

参考文献

[1] Shuai Bai、Yuxuan Cai、Ruizhe Chen、Keqin Chen、Xionghui Chen、Zesen Cheng、Lianghao Deng、Wei Ding、Chang Gao、Chunjiang Ge 等。Qwen3-VL 技术报告。*arXiv* 预印本 *arXiv:2511.21631*，2025年。[2] Shuai Bai、Keqin Chen、Xuejing Liu、Jialin Wang、Wenbin Ge、Sibo Song、Kai Dang、Peng Wang、Shijie Wang、Jun Tang 等。Qwen2.5-VL 技术报告。*arXiv* 预印本 *arXiv:2502.13923*，2025年。[3] M. L. Balinski 和 H. P. Young。杰斐逊分配法。*SIAMReview*，20(2):278–284，1978年。[4] Daniel Bolya、Cheng-Yang Fu、Xiaoliang Dai、Peizhao Zhang、Christoph Feichtenhofer 和 Judy Hoffman。令牌合并：让你的视觉 Transformer 更快。载于国际学习表征会议（*ICLR*），2023年。[5] Wang Chen、Yuhui Zeng、Yongdong Luo、Tianyu Xie、Lu-ojun Lin、Jiayi Ji、Yan Zhang 和 Xiawu Zheng。通过检测语义边界进行基于小波的帧选择，以实现长视频理解。载于*IEEE* 计算机视觉与模式识别会议（*CVPR*），2026年。[6] 分布式（深度）机器学习社区。Decord：用于深度学习的高效视频加载器。https://github.com/dmlc/decord，2019年。访问日期：2026年11月13日。[7] FFmpeg 开发者。FFmpeg 编解码器文档：编解码器选项。https://ffmpeg.org/ffmpeg-codecs.html，2026年。访问日期：2026年11月15日。[8] Chaoyou Fu、Yuhan Dai、Yongdong Luo、Lei Li、Shuhuai Ren、Renrui Zhang、Zihan Wang、Chenyu Zhou、Yunhang Shen、Mengdan Zhang 等。Video-MME：首个用于视频分析的多模态大语言模型综合评测基准。在 IEEE 计算机视觉与模式识别会议（CVPR），2025年。[9] Jingning Han、Bohan Li、Debargha Mukherjee、Ching-Han Chiang、Adrian Grange、Cheng Chen、Hui Su、Sarah Parker、Sai Deng、Urvang Joshi、Yue Chen、Yunqing Wang、Paul Wilkins、Yaowu Xu 和 James Bankoski。AV1 技术概览。《IEEE 会议录》，109(9)：1435–1462，2021年。[10] Wenyi Hong、Wenmeng Yu、Xiaotao Gu、Guo Wang、Guob-ing Gan、Haomiao Tang、Jiale Cheng、Ji Qi、Junhui Ji、Lihang Pan 等。GLM-4.5V 和 GLM-4.1V-Thinking：通过可扩展强化学习实现通用多模态推理。arXiv 预印本 arXiv:2507.01006，2025年。[11] Kai Hu、Feng Gao、Xiaohan Nie、Peng Zhou、Son Tran、Tal Neiman、Lingyun Wang、Mubarak Shah、Raffay Ha mid、Bing Yin 等。基于多模态大语言模型的视频帧选择，实现高效视频理解。在 IEEE 计算机视觉与模式识别会议（CVPR），2025年。[12] Jinwoo Hwang、Minsu Kim、Daeun Kim、Seungho Nam、Yoonsung Kim、Dohee Kim、Hardik Sharma 和 Jongse Park。CoVA：利用压缩域分析加速视频分析。在 USENIX 年度技术会议（ATC），2022年。[13] Daniel Kang、Ankit Mathur、Teja Veeramacheneni、Peter Bailis 和 Matei Zaharia。联合优化预处理与推理，实现基于深度神经网络的视觉分析。《VLDB 会议录》，14(2)：87–100，2020年。[14] Bruno Korbar、Du Tran 和 Lorenzo Torresani。SCSampler：从视频中采样显著片段，实现高效动作识别。在国际计算机视觉会议（ICCV），2019年。[15] Jeehong Lee、Ilhong Shin 和 Hyunwook Park。H.264 中可变 GOP 长度的自适应帧内分配与码率估计。《IEEE 电路与系统视频技术汇刊》（TCSVT），16(10)：1271–1279，2006年。[16] Junnan Li、Dongxu Li、Caiming Xiong 和 Steven Hoi。BLIP：为统一视觉语言理解与生成引导语言-图像预训练。在国际机器学习会议（ICML），2022年。[17] Debargha Mukherjee、Jim Bankoski、Adrian Grange、Jingn-ing Han、John Koleszar、Paul Wilkins、Yaowu Xu 和 Ronald Bultje。最新开源视频编解码器 VP9——概览与初步结果。在图像编码研讨会（PCS），2013年。[18] MulticoreWare。x265 文档：命令行选项。https://x265.readthedocs.io/en/4.1/cli.html。4.1 版。访问日期：2026年11月15日。[19] Yaxuan Qin、Hefei Li、Wenqi Mu 和 Yancheng He。通过强化学习实现长视频理解的高效帧选择。在 IEEE 计算机视觉与模式识别会议（CVPR），2026年。

[20]Alec Radford、Jong Wook Kim、Chris Hallacy、Aditya Ramesh、Gabriel Goh、Sandhini Agarwal、Girish Sastry、Amanda Askell、Pamela Mishkin、Jack Clark 等。从自然语言监督中学习可迁移视觉模型。在国际机器学习会议（ICML），2021年。[21] Sayan Deb Sarkar、R´emi Pautrat、Ondrej Miksik、Marc Polle-feys、Iro Armeni、Mahdi Rad 和 Mihai Dusmanu。CoPE-VideoLM：利用编解码器基元实现高效视频语言建模。arXiv 预印本 arXiv:2602.13191，2026年。[22] J. Sastre、P. Usach、A. Moya、V. Naranjo 和 J. M. Lopez。低码率 H.264 视频编码的镜头检测方法。在欧洲信号处理会议（EUSIPCO），2006年。[23] Benjamin Schneider、Dongfu Jiang、Chao Du、Tianyu Pang 和 Wenhu Chen。QuickVideo：通过系统与算法协同设计实现实时长视频理解。《机器学习研究汇刊》（TMLR），2026年。[24] Kele Shao、Keda Tao、Can Qin、Haoxuan You、Yang Sui 和 Huan Wang。HoliTom：面向快速视频大语言模型的整体式令牌合并。在神经信息处理系统进展大会（NeurIPS），2025年。[25] Rickard Sj¨oberg、Ying Chen、Akira Fujibayashi、Miska M. Hannuksela、Jonatan Samuelsson、Thiow Keng Tan、Ye-Kui Wang 和 Stephan Wenger。HEVC 高级语法与参考图像管理概览。《IEEE 电路与系统视频技术汇刊》（TCSVT），22：1870，2012年12月，1858–（页码格式原文如此）。[26] Chong Tang、Sannara Ek、Dirk Koch、Robert D. Mullins、Alex S. Weddell 和 Jagmohan Chauhan。SURGE：惊奇度引导的令牌缩减，实现基于视觉语言模型的高效视频理解。在国际学习表征会议（ICLR），2026年。[27] Xi Tang、Jihao Qiu、Lingxi Xie、Yunjie Tian、Jianbin Jiao 和 Qixiang Ye。面向长视频理解的自适应关键帧采样。在 IEEE 计算机视觉与模式识别会议（CVPR），2025年。[28] Keda Tao、Can Qin、Haoxuan You、Yang Sui 和 Huan Wang。DyCoke：面向快速视频大语言模型的动态令牌压缩。在 IEEE 计算机视觉与模式识别会议（CVPR），2025年。[29] TorchCodec贡献者。TorchCodec：适用于 PyTorch 的媒体解码与编码。https://github.com/meta-pytorch/torchcodec，2024年。0.15.0版。访问日期：2026年11月13日。[30] VideoLAN。x264：免费的 H.264/AVC 编码器。https://www.videolan.org/developers/x264.html。访问日期：2026年11月15日。[31] Hesong Wang、Xin Jin、Lu Lu、Chenhaowen Li、Jian Chen、Qiang Liu 和 Huan Wang。EarlyTom：提前进行令牌压缩，实现快速视频理解。在 IEEE 计算机视觉与模式识别会议（CVPR），2026年。[32] Weiyun Wang、Zhangwei Gao、Lixin Gu、Hengjun Pu、Long Cui、Xingguang Wei、Zhaoyang Liu、Linglin Jing、Shenglong Ye、Jie Shao 等。InternVL3.5：在通用性、推理能力和效率方面推进开源多模态模型。arXiv 预印本 arXiv:2508.18265，2025年。[33] Ye-Kui Wang、Robert Skupin、Miska M. Hannuksela、Sachin Deshpande、Hendry、Virginie Drugeon、Rickard Sj¨oberg，Byeongdoo Choi、Vadim Seregin、Yago Sanchez、Jill M. Boyce、Wade Wan 和 Gary J. Sullivan。多功能视频编码（VVC）标准的高级语法。*IEEE* 电路与系统视频技术汇刊（*TCSVT*），31(10):3779– 3800，2021。[34] T. Wiegand、G.J. Sullivan、G. Bjontegaard 和 A. Luthra。H.264/AVC 视频编码标准概述。*IEEE* 电路与系统视频技术汇刊（*TCSVT*），13(7):560– 576，2003。[35] Chao-Yuan Wu、Manzil Zaheer、Hexiang Hu、R Manmatha、Alexander J Smola 和 Philipp Kr¨ahenb¨uhl。压缩视频动作识别。发表于*IEEE* 计算机视觉与模式识别会议（*CVPR*），2018。[36] Haoning Wu、Dongxu Li、Bei Chen 和 Junnan Li。LongVideoBench：长上下文交错视频-语言理解基准。发表于 *Adv. Neural In- form.Process.Syst.(NeurIPS)*，2024年。[37] Senqiao Yang、Yukang Chen、Zhuotao Tian、Chengyao Wang、Jingyao Li、Bei Yu 和 Jiaya Jia。VisionZip：视觉语言模型中，越长越好，但并非必要。发表于*IEEE* 计算机视觉与模式识别会议（*CVPR*），2025年。[38] Daichi Yashima、Shuhei Kurita、Yusuke Oda 和 Komei Sugiura。ReMoRa：基于精细化运动表征的长视频理解多模态大语言模型。发表于 *IEEE* 计算机视觉与模式识别会议（*CVPR*），2026年。[39] Sicheng Yu、Chengkai Jin、Huanyu Wang、Zhenghao Chen、Sheng Jin、Zhongrong Zuo、Xiaolei Xu、Zhenbang Sun、Bingni Zhang、Jiawei Wu 等。Frame-Voyager：学习为视频大语言模型查询帧。发表于国际学习表征会议（*ICLR*），2025年。[40]Kaichen Zhang、Bo Li、Peiyuan Zhang、Fanyi Pu、Joshua Adrian Cahyono、Kairui Hu、Shuai Liu、Yuanhan Zhang、Jingkang Yang、Chunyuan Li 和 Ziwei Liu。LMMs-Eval：大型多模态模型评估的现实检验。发表于计算语言学协会会议论文集：*NAACL*，2025年。[41] Lingxiao Zhao、Haoran Zhou、Yuezhi Che 和 Dazhao Cheng。通过 GPU 内部调度和资源共享，实现解耦式多阶段多模态大语言模型推理。*arXiv* 预印本 *arXiv:2512.17574*，2025年。[42] Junjie Zhou、Yan Shu、Bo Zhao、Boya Wu、Zhengyang Liang、Shitao Xiao、Minghao Qin、Xi Yang、Yongping Xiong、Bo Zhang 等。MLVU：多任务长视频理解基准测试。发表于*IEEE Conf. Comput. Vis. Pat- tern Recog.(CVPR)*，2025年。[43] Zirui Zhu、Hailun Xu、Yang Luo、Yong Liu、Kanchan Sarkar、Zhenheng Yang 和 Yang You。FOCUS：面向长视频理解的高效关键帧选择。发表于国际学习表征会议（*ICLR*），2026年。[44] Sepideh Zohdi 和 Lin Wang。首个 token 之前：视觉语言模型数据预处理基准测试。发表于欧洲机器学习系统研讨会（*EuroMLSys*），2026年。[45] Yulin Zou、Wenyan Chen、Yan Chen、Anya Rajan、JooYoung Park、Shivaraman Nitin、Luo Tao、Francisco Romero 和 Dmitrii Ustiugov。CodecSight：利用视频编解码器信号实现高效的流式视觉语言模型推理。*arXiv* 预印本 *arXiv:2604.06036*，2026年。

RAPID：编解码器感知的视频解码，实现高效的长视频理解

补充材料

在补充材料中，我们提供了更多实现细节、实验、分析和可视化内容。补充材料的组织结构如下：

- 第A 节介绍实现细节，以及 RAPID-Dec 如何在每种编解码器中识别关键帧和非参考帧。
- 第 B 节在不同硬件上以及所有 Video-MME 视频上评估 RAPID-Dec，报告 RAPID-Sel 的候选开销，并将 RAPID 的准确率–首 token 时间权衡与现有方法进行比较。
- 第C节通过可视化示例，对关键帧与视觉变化之间的关系进行补充分析。
- 第 D 节讨论了 RAPID 的局限性和未来方向。

A. 实现细节

A.1. 详细评估设置

帧选择方法。 Top-K、AKS [27], 和 WFS-SB [5] 会对整段视频中以 1帧/秒 采样得到的候选帧进行评分。Top-K 会选择得分最高的帧，而 AKS 和 WFS-SB 则使用其官方实现中的算法和默认超参数来选择帧。我们运行 Focus 的官方实现 [43] 来识别查询相关区域。在这些区域内，选择得分最高的帧。对于 RAPID-Sel，τ 公式（2）中的 $\tau = 100$是预训练 CLIP 模型的 logit 缩放系数。如果短视频中的 1帧/秒的视频帧 少于帧预算，所有帧选择方法（包括 RAPID-Sel）都会退回到均匀采样。

模型输入。 输入视觉语言模型之前，每帧都会在保持宽高比的前提下调整大小，使像素数不超过 65,536，从而让两个模型接收到的像素数量几乎相同。Qwen3-VL-8B [1] 会将连续两帧的每个 32×32 区域映射为一个视觉令牌，因此每条边的长度都会取整为 32 的倍数。GLM-4.6V-Flash [10] 则使用28×28 区域，因此每条边的长度都会取整为最接近的 28 的倍数。

图 4 和 5中的视频。 这四个一小时长视频取自 Video-MME 中最长的 10% 视频 [8]。为覆盖不同的组长度，我们选择了两个关键帧组较长的视频（平均 149 帧和 158 帧）以及两个关键帧组较短的视频（47 帧和 63 帧）。我们使用 FFmpeg [7], 分别通过 libx265 和 libvpx-vp9 将它们重新编码为 H.265/HEVC 和 VP9。每个重新编码的视频都保留原视频的分辨率、帧率和像素格式，并将目标码率设为原视频流的码率。我们在原视频的每个 IDR 帧处强制插入一个帧内帧，防止编码器在场景变化处插入额外的关键帧，并保留 x265 的默认图像组（GOP）设置，该设置允许解码顺序中位于帧内帧之后的帧引用其之前的帧。对于 VP9，我们禁用备用参考帧，因为我们观察到 Decord 在使用多个线程时无法对其进行解码。

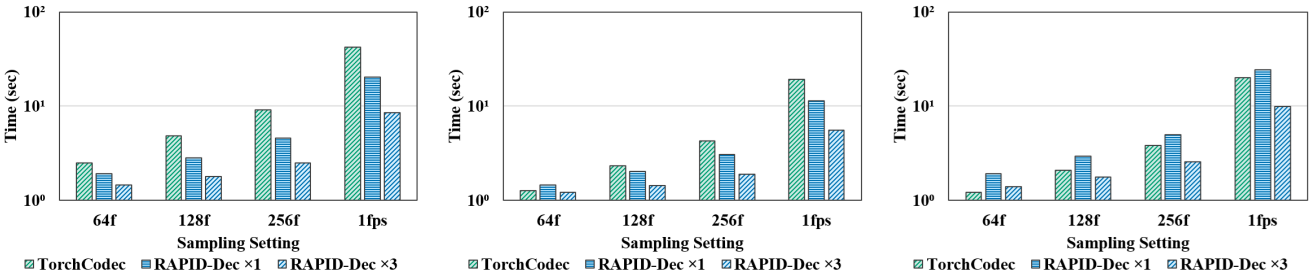
A.2. RAPID-Dec 在各种编解码器上的表现

如第3.1节所述，每种编解码器都定义了自己的比特流语法，其中包括用于指示关键帧和参考帧的语法元素。我们介绍 RAPID-Dec 如何在各类编解码器中识别关键帧和非参考帧。首先介绍 H.264/AVC，在 Video-MME [8], LongVideoBench [36], 和 MLVU [42]中，98.0% 的视频均采用该编码；随后介绍 H.265/HEVC，大多数其余视频均采用该编码。

H.264/AVC。 H.264/AVC 的每个 NAL 单元 [34] 均以一个字节的头部开头（图2b），其中包含 NAL 单元类型，用于指示该 NAL 单元所承载的数据类型，以及 NAL 参考标识。NAL 单元类型为 5 表示该切片属于 IDR 帧，而切片头部中的 NAL 参考标识为 0 则表示该帧为非参考帧。除了 IDR 帧之外，编码器还可以通过补充增强信息（SEI）NAL 单元（NAL 单元类型 6）承载的恢复点消息，将 I 帧标记为随机访问点；这种消息称为恢复点 SEI。RAPID-Dec 将 IDR 帧作为关键帧，也将恢复点 SEI 能够保证从该 I 帧开始解码所得帧与对整个视频进行解码所得帧完全一致的 I 帧作为关键帧，即 recovery frame cnt $= 0$、exact match flag $= 1$ 和 broken link flag $= 0$所指示的 I 帧。

在 H.264/AVC 开放图像组中，解码顺序上位于带有恢复点 SEI 的 I 帧之后、但显示顺序上位于其之前的 B 帧，可能会引用前一个组。我们的实现不支持此类码流。不过，x264 [30]，默认禁用开放图像组，而且基准测试视频中没有此类帧。

H.265/HEVC。 与 H.264/AVC 类似，H.265/HEVC 比特流 [25] 由 NAL 单元组成，但每个 NAL 单元都以一个双字节头部开头，其中包含 NAL 单元类型和时间层标识。H.265/HEVC 定义了三种类型的帧内随机访问点（IRAP）帧：IDR、清洁随机访问（CRA）和断链访问（BLA）。随机访问跳过的前导（RASL）帧可能在解码顺序中跟随 CRA 或 BLA 帧，并且可能引用前一组中的帧。由于从此类 IRAP 帧开始解码无法重建其 RASL 帧，因此，只有当该 IRAP 帧与解码顺序中的下一个 IRAP 帧之间不存在 RASL 帧时，RAPID-Dec 才会将其作为关键帧。对于



(a) H.264/AVC (b) H.265/HEVC (c) VP9

图9. 使用 NVDEC 进行视频解码的延迟。纵轴采用对数刻度。

非 IRAP 帧，NAL 单元类型同样携带参考信息，因为每种类型都有两个变体，其名称分别以 N 和 R 结尾。N 帧不会被同一时间子层中的后续帧引用，但可能被更高时间子层中的帧引用。因此，只有当 N 帧的时间层标识在视频中最高时，RAPID-Dec 才会将其视为非参考帧。

VP9。 VP9 [17] 是一种开源视频编解码器，其开发独立于 H.264/AVC 和 H.265/HEVC 的标准化工作；我们将其纳入研究，以检验 RAPID-Dec 是否能推广到基于 NAL 的编解码器之外。VP9 的每帧开头不是 NAL 单元，而是一个未压缩的头部，其中的一个比特用于指示该帧是否为关键帧，即能够重置解码状态的 VP9 帧类型。由于这种帧符合我们对关键帧的定义，RAPID-Dec 会将每个关键帧 都视为关键帧，并根据容器为每个数据包存储的关键帧标志识别它们。头部中的帧刷新标志则表明某帧是否可被引用。带有帧刷新标志的帧 $= 0$ 不会被存储为参考帧。不过，与 H.264/AVC 和 H.265/HEVC 中的非参考帧不同，这种帧仍可能影响后续帧的解码，因为下一帧可能会使用它的运动矢量或分割图。因此，仅凭帧刷新标志无法判断某帧是否可以跳过，RAPID-Dec 会将每个 VP9 帧都视为参考帧，所以第 3.3 节中的 (i) 和 (ii) 适用。

B. 扩展评估

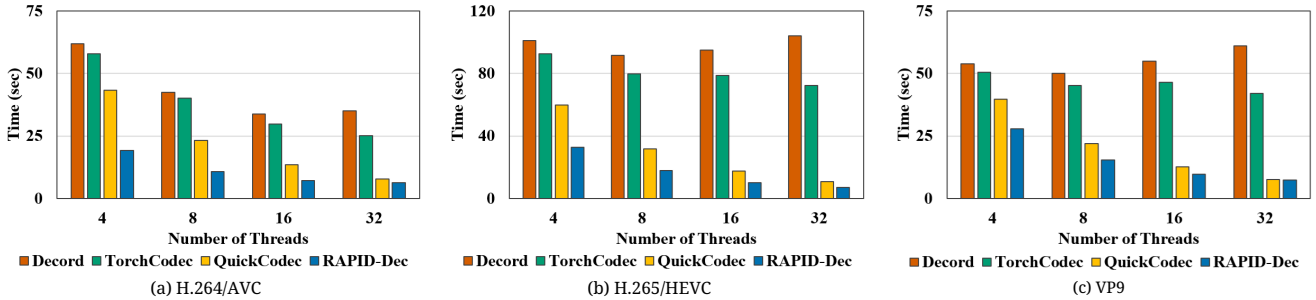
B.1. 不同硬件上的 RAPID-Dec

为验证 RAPID-Dec 的效率不依赖于特定的硬件配置，我们在 NVDEC 和另一种 CPU 上对其进行评估，使用与图4 和图 5中相同的视频。

NVDEC 结果。图9展示了 NVDEC 上的视频解码延迟，该数据是在配备三个 NVDEC 引擎的 RTX 5880 Ada GPU 上测得的。在各项基线方法中，Torch- Codec [29]是唯一官方发行版支持 NVDEC 解码的库，因此我们在此设置下将 RAPID-Dec 与 TorchCodec 进行比较。QuickCodec [23] 仅解码在 CPU 上运行，且 PyPI 发行版的 Decord [6] 未启用 CUDA 构建。TorchCodec 使用单个 NVDEC 解码器解码视频，也就是使用一个解码会话。由于单个解码会话的吞吐量受限于一个 NVDEC 引擎，TorchCodec 实际上只使用了三个引擎中的一个。相比之下，RAPID-Dec 将解码工作划分为多个组，每组都从一个关键帧开始。由于这些组可以独立解码，RAPID-Dec 能够将它们分配到多个解码器上，每个解码器运行各自的解码会话。因此，我们分别评估使用一个解码器的 RAPID-Dec（RAPID-Dec $\times 1$），以及使用三个并行运行的解码器、从而利用全部三个 NVDEC 引擎的 RAPID-Dec（RAPID-Dec $\times 3$）。

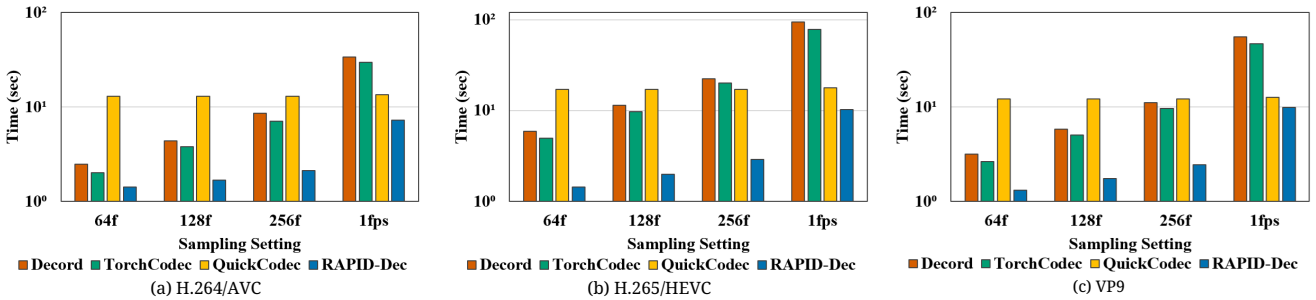
由于 RAPID-Dec 会跳过非参考帧，RAPID-Dec $\times 1$ 在所有设置下均优于 TorchCodec，唯一例外是 64帧下的 H.265/HEVC；并且 RAPID-Dec $\times 3$在每种 H.264/AVC 和 H.265/HEVC 设置下都是最快的，相较于 TorchCodec 最高可实现 4.9×倍加速。如第 A.2节所述，RAPID-Dec 将每个 VP9 帧都视为参考帧，因此非参考帧跳过不适用于 VP9。在缺少这一优势的情况下，RAPID-Dec 在索引和数据包处理方面的 CPU 端开销使 RAPID-Dec $\times 1$在 VP9 上慢于 TorchCodec。同样的开销也会影响 64帧下的 H.265/HEVC，因为此时所需的解码工作较少。使用三个解码器时，并行解码带来的收益超过了这一开销，因此 RAPID-Dec $\times 3$ 从 128帧起在 VP9 上优于 TorchCodec。因此，在三个引擎全部启用的情况下，RAPID-Dec 在除 64 帧下的 VP9 之外的所有设置中，都降低了 NVDEC 上的解码延迟。

不同 CPU 的结果。 除主论文中使用的 Intel Xeon Gold 6530 CPU 外，我们还在 2.8 GHz 的 AMD EPYC 7543 32 核 CPU 上评估了 RAPID-Dec，图 4 和图 5 中的其他设置均保持不变。如图 10 和图 11所示，结果与 Intel CPU 上的趋势一致。在每种设置下，RAPID-Dec 均实现了最低延迟；Decord 和 TorchCodec 随线程数增加的扩展性较差；而 QuickCodec 的延迟在不同采样设置下几乎保持不变。在 1 帧/秒采样和默认的 16个线程下，RAPID-Dec 比最快的基线方法 QuickCodec 快 1。9×、1。7×和 1。3×倍，分别对应



(a) H.264/AVC (b) H.265/HEVC (c) VP9

图10。AMD CPU 上使用不同线程数进行 1帧/秒采样（最多 2,048 帧）的视频解码延迟。



(a) H.264/AVC (b) H.265/HEVC (c) VP9

图11. AMD CPU 使用16个线程时，不同帧采样设置下的视频解码延迟。纵轴采用对数刻度。

H.264/AVC、H.265/HEVC 和 VP9，分别接近在 Intel CPU 上的 2.0×、1.7× 和 4× 倍加速比。在其他采样设置下，使用 16个线程时，对于三种编解码器，在 64、128 和 256 帧下，RAPID-Dec 分别比各设置中最快的基线方法快 1.4–3.5×、2.3–4.9× 和 3.3–5.9× 倍。

B.2. RAPID-Dec的大规模评估

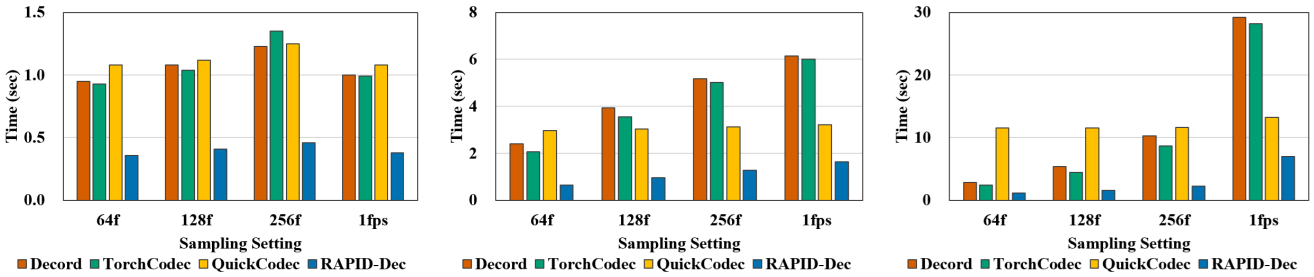
为检验 RAPID-Dec 的加速效果能否推广到这四段一小时长视频之外，我们在 Video-MME 的全部 900 段视频上重复图 5 中的实验。这些视频均采用 H.264/AVC 编码，线程数和帧采样设置也相同。由于结果是对大量视频取平均值，因此每段视频只解码一次。

视频时长的影响。 图12展示了短、中、长视频的延迟，其时长中位数分别约为 1、8 和 40 分钟。在所有三种时长的视频中，RAPID-Dec 在每种采样设置下都实现了最低延迟。由于 QuickCodec 会解码每一帧，随着采样变得更加稀疏，其与 RAPID-Dec 之间的差距会扩大；这一点在长视频上最为明显，因为长视频包含的帧最多。相比之下，在中、长视频上，RAPID-Dec 相对于 TorchCodec 的加速比从 64 帧增加到 256 帧，这是因为更多组包含目标帧，而 RAPID-Dec 会并行解码这些帧。因此，最快的基线方法会随采样设置和视频时长而变化，而 RAPID-Dec 在所有情况下都是最快的。在所有视频上取平均值（图 13a），RAPID-Dec 在每种采样设置下都比最快的基线方法快 1.9–3。7×倍，接近 1.7–3。7× 在图中时长约一小时的 H.264/AVC 视频上。5a。

特殊情况。 我们进一步考察了两个可能限制 RAPID-Dec 效果的视频子集。第一个子集由不含非参考帧的视频组成，占所有视频的 4.4%；对这些视频而言，跳过非参考帧并无效果。如图 13b 所示，RAPID-Dec 在这些视频上的加速比低于在所有视频上的加速比，尤其是在 1帧/秒时，此时跳过非参考帧是效果最显著的组成部分（表 2）。尽管如此，RAPID-Dec 在所有设置下仍能实现最低延迟。第二个子集由组长度中位数最长的 5% 视频组成，其组长度中位数超过 5.75 秒。组越长，可并行解码的组数就越少。不过，编码器会在组达到最大长度时插入关键帧 [18, 30], , 因此即使是这些视频也会被划分为多个组；当只有少数组包含目标帧时，RAPID-Dec 会按照第 3.3 节所述，在每个解码器内部使用剩余线程。因此，RAPID-Dec 在这些视频上的加速比仍接近其在所有视频上的加速比（图 13c）。

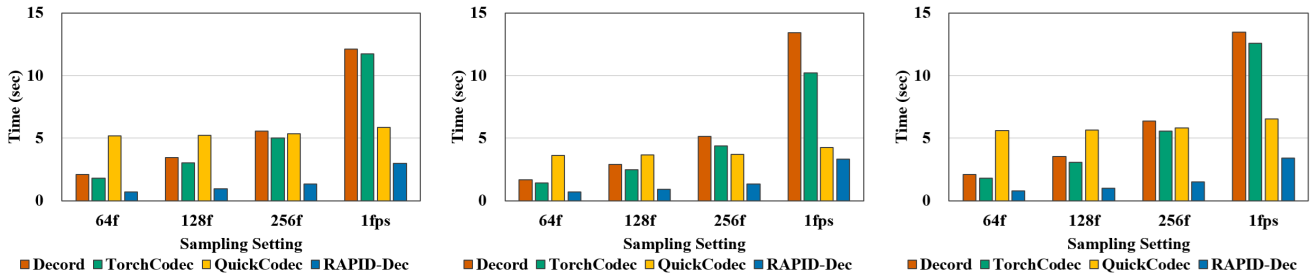
B.3. RAPID-Sel 的候选开销

除表 4中的准确率外，表 6 还报告了 RAPID-Sel 使用均匀候选项和关键帧候选项时的预处理延迟，每个视频的候选项数量相同。延迟的测量方式与表3 相同，测试对象为 10% 的 Video-MME 视频；None 表示不使用候选项的均匀采样。关键帧候选项取得了



(a) 短（<2 分钟） (b) 中（4–15分钟） (c) 长（30–60分钟）

图 12. 使用16个线程时，Video-MME中不同视频时长对应的视频解码延迟。



(a) 所有视频 (b) 不包含非参考帧 （c）长组长度

图13. 使用16个线程时，所有 Video-MME 视频及两个特殊案例的视频解码延迟。

表6. 使用 Qwen3-VL-8B 和 64帧时，RAPID-Sel 在 Video-MME 上均匀候选项与关键帧候选项的准确率（%）和预处理延迟。

候选项	预处理（秒）↓	短	中	Long	Avg.
None	0.75	74.56	61.89	54.44	63.63
均匀	2.87	**76.22**	63.67	55.22	65.04
关键帧	**1.39**	75.67	**63.78**	**56.78**	**65.41**

略高的平均准确率，在长视频上提升最大（+1.56 个百分点），且预处理速度比均匀候选项快 2.1×。相较于不使用候选项的均匀采样，关键帧候选项仅增加 0.64 秒，而均匀候选项增加 2.12 秒。由于两种设置都对相同数量的候选项进行评分，这一差距主要源于候选帧解码。关键帧不引用其他帧，可单独解码；而均匀间隔的候选帧通常需要解码其引用的帧，并从前一个关键帧开始。因此，关键帧是比均匀间隔帧更高效的候选项，同时准确率也略高，尤其是在长视频上。

B.4. 准确率与首 token 时间

表 3比较了相同帧预算下各帧选择方法的预处理延迟。然而，首 token 时间还取决于输入帧的数量，因为帧数越多，视觉语言模型编码并预填充的视觉令牌就越多。A

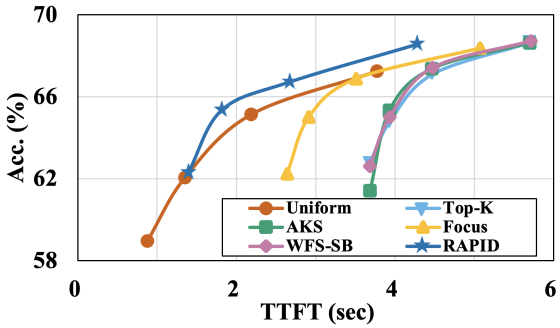


图 14. Qwen3-VL-8B 使用 32 至 256 帧时，三个基准测试的平均准确率与首 token 时间的关系。

因此，选择开销较小的方法能够在相同的首 token 时间内向视觉语言模型输入更多帧，而均匀采样没有选择开销。为在完整流程下比较这些方法，图14 绘出了 Qwen3-VL-8B 在 32、64、128 和 256 帧下的准确率与首 token 时间；每种方法的各个数据点从左到右按帧数递增排列。准确率取 Video-MME、MLVU 和 LongVideoBench 上的平均值；首 token 时间则与表 3 一样，在相同的 10% 视频上测量，所有方法均使用 RAPID-Dec。在准确率相同的情况下，RAPID 位于其他方法的左侧，表明它能以更低的首 token 时间达到相同准确率。在首 token 时间相同的情况下，RAPID 位于其他方法的上方，准确率更高。除 32 帧的情况外，这两种趋势在 RAPID 涵盖的整个范围内均成立；在 32 帧时，

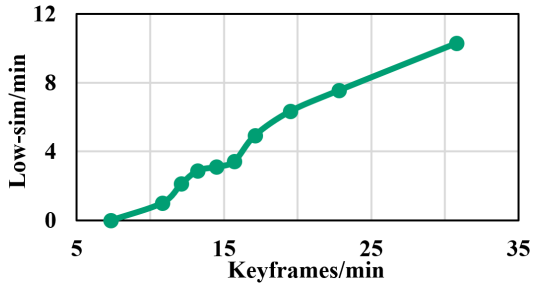


图 15. 按每分钟关键帧数排序的十组视频中，每分钟关键帧和低相似度区间（Low-sim）数量的中位数。

RAPID 在 64 帧时的表现与均匀采样相当。其他帧选择方法的准确率与 RAPID 相近，但较大的选择开销使它们在帧数相同时的首 token 时间高于 RAPID。因此，与使用更多帧的均匀采样以及现有帧选择方法相比，RAPID 在准确率与首 token 时间之间实现了更好的权衡。

C. 分析

C.1. 关键帧与视觉变化

如图 7a所示，在至少包含五个此类区间的视频中，每个视频包含关键帧的低相似度区间比例的中位数为 0.81，因此每个比例都基于足够多的区间计算得出。如果不考虑内容而放置关键帧，那么这一比例应接近所有 1 秒区间中包含关键帧的比例。然而，在相同视频中，后者的中位数仅为 0.26，而实际数值（0.81）3.1× 更高。这进一步表明，关键帧的放置是依据视觉变化，而不是独立于内容进行的。

接下来，我们考察视觉变化更多的视频是否包含更多关键帧。对于每个视频，我们使用与第 4.3 节相同的 0.73 阈值，计算每分钟关键帧数和每分钟低相似度区间数。与比例不同的是，这些计数是除以视频时长，而不是低相似度区间的数量，因此我们也纳入上文排除的视频。我们按照每分钟关键帧数对视频进行排序，将其划分为大小相等的十组，并在图 15中绘制每组的中位数。每分钟低相似度区间数随着每分钟关键帧数的增加而增加，在单个视频上的斯皮尔曼相关系数为 0.61。这表明，视觉变化更多的视频往往包含更多关键帧。

C.2. 关键帧放置示例

上述分析表明，关键帧出现在视觉变化处，且视觉变化越多的视频包含的关键帧也越多。图 16说明，在同一视频中也是如此：对比四个视频中的一个稀疏片段和一个密集片段即可看出这一点。同一视频的两个片段来自相同的编码，因此关键帧数量的差异反映了它们的内容。在这些例子中，密集片段包含的关键帧数量是稀疏片段的 3.1–7。1× 倍。在稀疏片段中，CLIP 相似度始终接近 1，很少低于图 7中用于低相似度区间的 0.73 阈值，而且连续的关键帧通常显示相同的镜头。在密集片段中，相似度较低且波动更大，三个连续的关键帧仅相隔几秒，却显示不同的镜头。

稀疏片段中，人们坐着交谈，动作很少。相比之下，密集片段的内容变化迅速，包括短镜头序列、彩色灯光下的夜间派对、电梯内的打斗，以及战斗场景。即使在稀疏片段中，每隔几秒仍会出现关键帧。在第一个和最后一个视频中，关键帧几乎以固定间隔放置，这可能是因为当一组达到最大长度时，编码器会插入关键帧，正如图中所讨论的，跨越关键帧的高度相似对7b。这类关键帧并不标记视觉变化，但由于每组内部的内容变化很少，它们仍能代表各自的组。因此，RAPID-Sel 评分较高的关键帧在动态片段中较为密集，在静态片段中较为稀疏；在静态片段中，只需少量关键帧就足以代表由相似帧构成的长组。

D. 局限性与未来工作

RAPID-Sel 用关键帧表示每个组，因此只能以较粗略的粒度估计组的相关性。组内逐渐变化的视觉内容——例如，关键帧与组内其他帧之间的相似度略低所表明的情况（图7b）——并未反映在相关性评分中。因此，对于涉及细粒度动作的问题，RAPID-Sel 可能会漏掉所需的帧。比特流中的运动矢量可指示组内内容的移动幅度，将其与关键帧的查询相关性结合起来选择帧，是未来一个有前景的研究方向。此外，我们的 RAPID-Dec 实现支持 H.264/AVC、H.265/HEVC 和 VP9。AV1 [9]和 H.266/VVC 等较新的编解码器 [33]也会在头部中标记关键帧和参考信息，因此可以通过解析这些头部，将 RAPID-Dec 扩展到这些编解码器。



图16。四个视频中一个稀疏分钟（左）和一个密集分钟（右）的关键帧。每张图展示相邻 1帧/秒的视频帧之间的 CLIP 相似度、低相似度区间的阈值 0.73，以及该分钟的关键帧。每张图下方展示该分钟连续的三个关键帧。