CODEC-DERIVED INFORMATION PRIORS FOR VIDEO TOKEN SPARSIFICATION IN VISION LANGUAGE MODEL INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a codec-guided video token sparsification framework for high-throughput large vision-language model serving. Existing methods either require full visual encoding for frame selection or depend on intermediate features and attentions for token pruning, placing additional model-side computation on the GPU critical path and increasing serving overhead under concurrent requests. To this end, we leverage compressed-domain information exposed by video codecs to jointly perform frame selection and token mask generation before visual encoding. Specifically, frame selection is guided by coding cost and motion magnitude extracted from the bitstream, while token masks are generated from spatial priors based on motion and texture complexity. By resolving visual input selection on the CPU before visual encoding, IPB decouples sparsification from downstream GPU inference and enables CPU–GPU overlap across requests. At high concurrency, IPB achieves 3.2× the aggregate throughput and 38.0× lower TTFT than a combined frame-selection and token-pruning baseline, while maintaining favorable accuracy–efficiency trade-offs across four video benchmarks.

1 INTRODUCTION

Large vision-language models (VLMs) (Bai et al., 2025; Jin et al., 2024a; Li et al., 2024a; Lin et al., 2024; 2023; Liu et al., 2024; Maaz et al., 2024; Zhang et al., 2024) are rapidly expanding from image understanding to video understanding, where input length becomes a fundamental systems bottleneck. In practical video settings, the number of candidate frames and visual tokens grows with video duration, increasing both visual-encoding cost and the memory and compute burden of downstream multimodal inference. Efficient video VLM serving therefore requires visual-token reduction whose selection overhead remains outside the GPU critical path as concurrency increases, while preserving informative visual evidence (Hao et al., 2025; Wen et al., 2025).

Existing video sparsification methods mainly follow two directions: frame selection and token pruning. Frame selection methods (Liu et al., 2025; Sun et al., 2025; Tang et al., 2025; Zhu et al., 2025; Zou et al., 2025) score candidate frames using pretrained vision-language encoders (Li et al., 2022; Radford et al., 2021; Zhai et al., 2023) or auxiliary VLMs, while token-pruning methods (Bagrov et al., 2025; Chen et al., 2024; Shang et al., 2025; Tao et al., 2025) derive pruning masks from intermediate features or attention maps during inference. Although effective, both approaches introduce additional model-side computation on the *GPU critical path*. Under concurrent serving, selection overhead competes with VLM inference for GPU resources and accumulates across requests, with a larger cost for long-video inputs. As illustrated in Figs. 1a and 1b, the computation cost further grows with video length, making model-side selection increasingly expensive for long-video serving. Practical video sparsification therefore requires visual input selection outside the GPU critical path.

One practical way to perform visual input selection outside the GPU critical path is to exploit the redundancy cues already exposed by codecs. Modern video codecs (Sullivan et al., 2012; Wiegand et al., 2003) store videos in a redundancy-reduced form through *inter-frame prediction*, i.e., motion-vector-based prediction from reference frames. As a result, codec metadata such as frame type, packet size, and motion vectors implicitly reflects where the encoder spends bits to represent non-redundant temporal and spatial changes. Recent studies (Jin et al., 2024b; Sarkar et al., 2026; Tang et al., 2026;

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

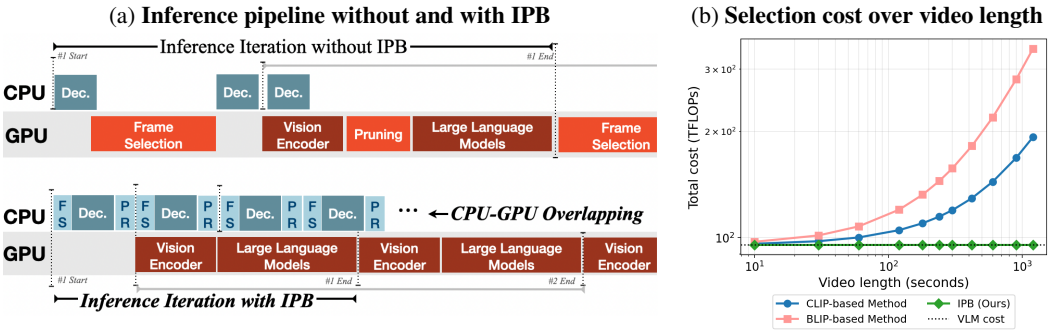


Figure 1: **Motivation for codec-guided visual input selection.** (a) Prior pipelines place selection and pruning on the *GPU critical path*, whereas IPB resolves frame selection (FS) and pruning (PR) before downstream inference and enables CPU–GPU overlap across requests. (b) As video length increases, the cost of model-based selection, such as CLIP (Radford et al., 2021) and BLIP (Li et al., 2022), grows with input duration, while IPB adds negligible overhead relative to VLM inference.

Zou et al., 2026; Wu et al., 2026) have begun to exploit codec-side redundancy cues for VLMs, demonstrating the utility of codec metadata as a structural prior for efficient video understanding.

In this paper, we propose **I**nformation **P**riors from **B**itstream (IPB), a token sparsification framework that derives temporal and spatial selection priors from compressed-domain video signals before GPU-side visual encoding. The proposed framework extracts codec priors from a single bitstream scan to perform frame selection and patch masking before visual encoding. By decoupling visual input selection from GPU-side VLM inference, IPB enables CPU-GPU overlap across requests and improves throughput. Specifically, frame selection is guided by GOP-level codec scores that combine coding cost and motion activity, while patch masks are generated from patch-level motion and texture scores. In a system-level serving evaluation, IPB achieves 3.2× the aggregate throughput and 38.0× lower Time-to-First-Token (TTFT) than a combined frame-selection and token-pruning baseline at high concurrency, while maintaining competitive accuracy across four video benchmarks.

In summary, our main contributions are as follows:

- We propose **Information Priors from Bitstream (IPB)**, a codec-guided video token sparsification framework that jointly performs temporal frame selection and spatial patch masking for video VLM inference without pretrained selection models or additional training.
- We introduce a hierarchical codec-prior selection algorithm that derives temporal support from GOP-level coding cost and motion activity, and generates spatial patch masks from patch-level motion and texture proxies.
- We design IPB to resolve visual input selection purely on CPU, decoupling sparsification from downstream GPU inference and yielding high-concurrency throughput–latency gains, including 3.2× the aggregate throughput and 38.0× lower TTFT than the strongest combined frame-selection and token-pruning baseline.

2 Related Works

Token Sparsification Prior VLM research has often reduced input length through uniform frame sampling, which treats redundant and informative content alike. As shown in Fig. 2(a), recent frame-selection methods, exemplified by BOLT (Liu et al., 2025), rely on pretrained encoders (Li et al., 2022; Radford et al., 2021; Zhai et al., 2023) or auxiliary VLMs to score candidate frames while accounting for temporal coverage (Buch et al., 2025; Sun et al., 2025; Tang et al., 2025; Zhu et al., 2025; Zou et al., 2025). A parallel line of work, illustrated in Fig. 2(b), reduces token-level redundancy by deriving pruning masks from intermediate features or attention maps during inference (Chen et al., 2024; Shang et al., 2025; Tao et al., 2025). EVS (Bagrov et al., 2025) is a representative example, performing inference-time token reduction through model-side scoring of visual tokens. In contrast, as illustrated in Fig. 2(c), IPB resolves both frame selection and patch masking before the GPU critical path, which enables pipelined execution across requests.

Codec-Guided Video Efficiency Methods Recent work has explored the use of codec information in VLMs. Encoder-based approaches (Jin et al., 2024b; Sarkar et al., 2026; Tang et al., 2026) train visual encoders to exploit codec primitives through learned representations. Beyond learned codec

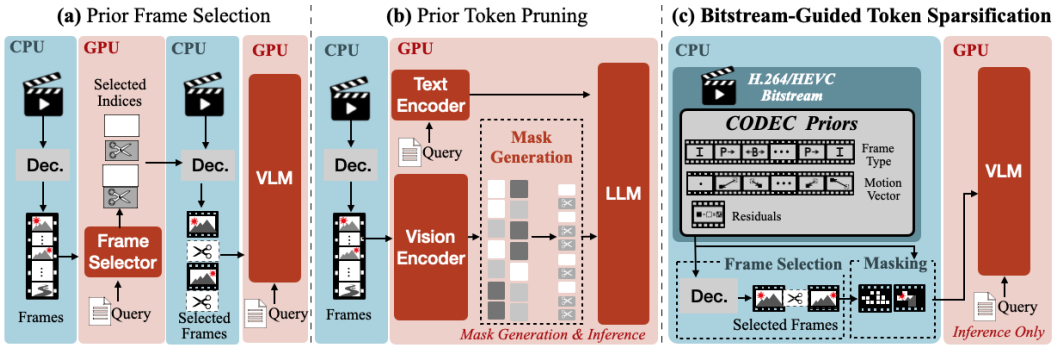


Figure 2: Comparison between prior selection/pruning pipelines and the proposed bitstream-guided framework. (a) Prior frame selection methods (Liu et al., 2025; Tang et al., 2025; Zhu et al., 2025) decode video frames and then perform query-conditioned selection with pre-trained models (Radford et al., 2021; Zhai et al., 2023). (b) Prior token pruning methods (Bagrov et al., 2025; Tao et al., 2025; Shang et al., 2025; Chen et al., 2024) compute pruning masks from model-internal representations, coupling mask generation with GPU-side inference. (c) The proposed IPB derives codec priors directly from the bitstream and performs spatio-temporal mask generation on the CPU, including frame selection and patch masking, before VLM inference.

encoders, Zou et al. (2026) use codec metadata for patch pruning and KV-cache management in streaming VLM serving. In contrast, IPB uses compressed-domain codec statistics to jointly perform frame selection and patch masking before visual encoding, in a training-free and plug-in manner without semantic selectors or trained codec encoders.

3 METHOD

Codec metadata provides fixed, query-agnostic signals, including packet size and motion structure. However, these signals capture compressed-video redundancy rather than query-relevant semantics, making codec-only selection difficult because the selection rule must be inferred from video-internal statistics alone. Codec-only selection is therefore non-trivial for both temporal frame selection and spatial patch selection.

3.1 PRELIMINARY

In this section, we introduce the codec-level concepts required for our method. We assume that the input video bitstreams contain key metadata: frame types, packet sizes, and motion vectors. These attributes are commonly available in standard encoded streams, such as H.264 (Wiegand et al., 2003) and HEVC (Sullivan et al., 2012). Predictive coding reduces temporal redundancy, and our method focuses on compressed-domain signals produced by inter-frame prediction. Let $F_i \in \mathbb{R}^{H\times W\times 3}$ denote the i-th RGB frame, and let $\mathcal{E}$ denote an abstract frame encoder that produces a bitstream b_i. Video frames are categorized into I-, P-, and B-frames according to their prediction structure. I-frames are intra-coded independently, while P- and B-frames are inter-coded through motion-compensated prediction. For an inter-coded frame, let

$$V_i = \{(v_{i,k}, \mathcal{B}_{i,k})\}_{k=1}^{N_i}, \qquad v_{i,k} \in \mathbb{R}^2, \tag{1}$$

where $v_{i,k}$ is the k-th motion vector and $\mathcal{B}_{i,k}$ is the spatial support of its associated prediction block. Given V_i, the codec constructs motion-compensated predictions for P- and B-frames as

$$R_i^{\mathrm{P}} = F_i - \mathcal{P}(F_{\mathcal{R}^-(i)}, V_i), \qquad R_i^{\mathrm{B}} = F_i - \mathcal{P}(F_{\mathcal{R}^\pm(i)}, V_i). \tag{2}$$

where $F_{\mathcal{R}^-(i)}$ and $F_{\mathcal{R}^\pm(i)}$ denote past-only and bidirectional reference frame sets, respectively. The corresponding coding modes are written as

$$b_i^{\mathrm{I}} = \mathcal{E}_{\mathrm{I}}(F_i), \quad b_i^{\mathrm{P}} = \mathcal{E}_{\mathrm{P}}(R_i^{\mathrm{P}}, V_i), \quad b_i^{\mathrm{B}} = \mathcal{E}_{\mathrm{B}}(R_i^{\mathrm{B}}, V_i). \tag{3}$$

The bitstream b_i provides the frame type $\{\mathrm{I}, \mathrm{P}, \mathrm{B}\}$, packet size $|b_i|$, and motion vectors V_i for inter-coded frames; residual complexity is reflected in $|b_i|$. I-frames are inserted at abrupt scene changes or periodic intra-coded anchor points. Let a_g denote the index of the g-th I-frame. We define the g-th Group of Pictures (GOP) as

$$\Gamma_g = \{F_i\}_{i=a_g}^{a_{g+1}-1}. \tag{4}$$

Algorithm 1. IPB Frame Selection

Input: Compressed video $\{b_i\}_{i=1}^{T}$ and frame budget K
Output: Selected frame set $S^\star$

```
Parse {|b_i|, V_i, type_i}_{i=1}^T
Build GOPs {Γ_g}_{g=1}^{N_GOP}
Compute GOP scores {u_g} using eqs. (6) and (7)
Convert {u_g} into frame-level density û_i using eq. (8)
Set τ ← Quantile_q({û_i : û_i > 0})
Build mask Ω = {i | û_i ≥ τ} using eq. (9)
Merge Ω using eq. (10)
Discard short segments using eq. (11)
Compute masses μ_j = Σ_{i∈C_j∩Ω} û_i using eq. (12)
if no segment remains then
    return S* = Uniform({1, ..., T}, K)
end if
S* ← AllocSample(C*, {μ_j}, K)
return S*
```

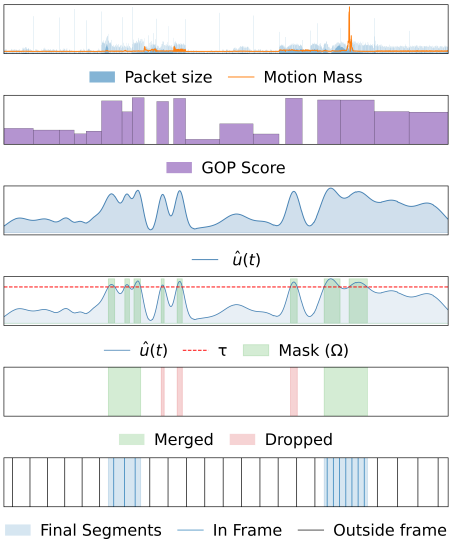


Figure 3: Bitstream-guided temporal frame selection algorithm and the corresponding pipeline. Packet size and motion mass are aggregated into GOP-level codec scores, converted into a frame-level density, thresholded to extract informative temporal support, and post-processed by merging and short-segment removal before final budget allocation.

A GOP serves as an operational temporal unit with a locally coherent scene context and prediction structure. The packet size $|b_i|$ serves as a coding-cost proxy, with larger values reflecting spatio-temporal changes that are more difficult to predict. Our method is built on the hypothesis that the spatio-temporal regions required by VLMs for video understanding are highly correlated with information-theoretic complexity.

3.2 Bitstream-guided frame selection

Frame selection aims to select a subset of K frames from a video with T frames such that the selected subset maximizes the downstream utility of a VLM over a query distribution $\mathcal{Q}$, formally,

$$S^\star = \arg\max_{S\subseteq\{1,\dots,T\}} \mathbb{E}_{Q\sim\mathcal{Q}}\left[\mathcal{U}\left(\mathcal{G}(\{F_i\}_{i\in S}, Q)\right)\right] \quad s.t. \quad |S| = K. \tag{5}$$

where $\mathcal{G}$ denotes the VLM and $\mathcal{U}$ denotes the downstream task utility. Because codec signals are fixed and query-agnostic, we do not rank frames by semantic relevance. Instead, we detect informative temporal segments from compressed-domain statistics and allocate dense coverage to those segments while keeping sparse coverage elsewhere.

GOP Score Our temporal selection starts by estimating informative scene segments from GOP-level codec statistics. A large packet size indicates high coding cost, but bit cost alone is not a reliable indicator of useful visual evidence. For example, fade-in, fade-out, or credit scenes often produce large residuals while containing limited temporal dynamics. We therefore combine packet rate with motion activity to assign a score to each GOP.

Let $\Gamma_g = \{F_i\}_{i=a_g}^{a_{g+1}-1}$ denote the g-th GOP and $L_g = |\Gamma_g|$ its length. Let $V_i = \{v_{i,k}\}_{k=1}^{N_i}$ denote the motion-vector set of frame i. For each Γ_g, we compute the average packet rate and the average motion mass:

$$B_g = \frac{1}{L_g}\sum_{i=a_g}^{a_{g+1}-1} |b_i|, \qquad M_g = \frac{1}{L_g}\sum_{i=a_g}^{a_{g+1}-1} m_i, \quad m_i = \sum_{k=1}^{N_i} |\mathcal{B}_{i,k}|\|v_{i,k}\|_2. \tag{6}$$

Here $|\mathcal{B}_{i,k}|$ is the prediction-block area in pixels. We normalize by positive frame-level medians $s_B = \mathrm{median}_{i:|b_i|>0}\,|b_i|$ and $s_M = \mathrm{median}_{i:m_i>0}\,m_i$, yielding $\widetilde{B}_g = B_g/s_B$ and $\widetilde{M}_g = M_g/s_M$.

An empty positive set uses scale 1; scales are floored at 10^{-9}. The GOP score u_g is then defined as

$$u_g = \underbrace{\log(1+\widetilde{B}_g)}_{\text{Coding score}} \cdot \underbrace{\left(1-\exp(-\beta\widetilde{M}_g)\right)}_{\text{Motion gate}}. \tag{7}$$

The logarithmic term defines an outlier-robust coding score. The exponential term forms a motion gate that suppresses GOPs with weak temporal dynamics. Thus, GOPs with both high compressed-domain complexity and non-trivial motion receive a high score.

Temporal Density Conversion The GOP score u_g is a discrete score over Γ_g. Directly ranking GOPs by u_g tends to chase isolated bitrate spikes or transient motion bursts, such as flash cuts. We convert u_g into a frame-level temporal density:

$$\hat{u}_i = \sum_{g=1}^{N_{\text{GOP}}} u_g \exp\left(-\frac{1}{2}\left(\frac{i-c_g}{\sigma_g}\right)^2\right), \qquad c_g = a_g + \frac{L_g}{2}, \quad \sigma_g = \max(1, \kappa L_g). \tag{8}$$

Here, $\hat{u}_i$ denotes the temporal density at frame index i, and κ controls the spread of each GOP score. The density construction suppresses isolated GOP-level spikes and emphasizes event-like regions with sustained compressed-domain evidence. The subsequent segment detection is therefore performed on temporal regions rather than keyframes or GOPs.

Temporal Masking We extract informative temporal segments by thresholding the frame-level density $\hat{u}$. The threshold τ is the selected quantile of $\{\hat{u}_i : \hat{u}_i > 0\}$, and the high-density mask is

$$\Omega = \{i \mid \hat{u}_i \geq \tau\}. \tag{9}$$

Connected components of Ω define candidate temporal segments $\mathcal{C}^{(0)} = \{C_j^{(0)} = [\xi_j^-, \xi_j^+)\}$. Adjacent candidates are merged when their temporal gap is smaller than Δ_{gap}:

$$[\xi_j^-, \xi_j^+),\ [\xi_{j+1}^-, \xi_{j+1}^+) \longrightarrow [\xi_j^-, \xi_{j+1}^+) \quad \text{if} \quad \xi_{j+1}^- - \xi_j^+ < \Delta_{\text{gap}}. \tag{10}$$

Let $\widetilde{\mathcal{C}}$ denote the merged candidate set. Short segments are discarded:

$$\mathcal{C}^\star = \left\{ C_j \in \widetilde{\mathcal{C}} \,\middle|\, |C_j| \geq \Delta_{\min} \right\}. \tag{11}$$

Segments in $\mathcal{C}^\star$ form event-like temporal regions with sustained compressed-domain evidence.

Budget Allocation The frame budget K is allocated over the surviving segments according to their density mass. For each surviving segment $C_j = [\xi_j^-, \xi_j^+)$, the density mass is

$$\mu_j = \sum_{i \in C_j \cap \Omega} \hat{u}_i. \tag{12}$$

The final selected frame set is obtained as

$$S^\star = \text{AllocSample}\left(\mathcal{C}^\star, \{\mu_j\}_{j=1}^{|\mathcal{C}^\star|}, K\right). \tag{13}$$

High-mass segments receive denser temporal samples, while low-density regions remain sparsely covered. Thus, segment discovery determines informative regions independently of K, and K only controls the final sampling density. Overall pipeline of section 3.2 is explained by algorithm 1. Detailed hyperparameter settings are provided in the Appendix section A.2.

3.3 Bitstream-guided Patch Selection

Temporal frame selection reduces redundancy across frames. We further reduce spatial redundancy within the selected frames by constructing patch masks before visual encoding. Prior compressed-domain patchification schemes (Sarkar et al., 2026; Zou et al., 2026; Tang et al., 2026) use motion vectors and residual cues to localize informative regions. We build on this principle, but replace residual-specific processing with a lightweight Sobel-based proxy for spatial structure. The proxy provides spatial texture cues without reconstructing codec prediction residuals. Given the selected frame set $S^\star$, patch selection constructs a binary mask $\mathcal{M}_i \in \{0,1\}^{H_p \times W_p}$ for each $i \in S^\star$, where $H_p = \lfloor H/p \rfloor$ and $W_p = \lfloor W/p \rfloor$ for patch size p. For each selected frame, we define a Sobel-based texture proxy $R_i = \mathcal{S} * I_i$, where I_i denotes the grayscale image of frame F_i, $\mathcal{S}$ denotes the Sobel

Table 1: **System-level performance comparison.** We compare IPB against Uniform+EVS and BOLT+EVS on Qwen3-VL-8B and AoT on LLaVA-OV-7B. We report mean Time to First Token (TTFT) and aggregate output-token throughput across concurrency levels C and retention ratio r_p. IPB closely matches the serving cost of Uniform+EVS while substantially reducing latency and improving throughput over BOLT+EVS and AoT. OOM denotes out-of-memory failures.

Method	r_p	C=4		C=8		C=16		C=32		C=64	
		TTFT (s) ↓	tok/s ↑	TTFT (s) ↓	tok/s ↑	TTFT (s) ↓	tok/s ↑	TTFT (s) ↓	tok/s ↑	TTFT (s) ↓	tok/s ↑
Qwen3-VL-8B, K=64											
Uniform (no pruning)	1.00	1.59	25.9	1.89	27.2	2.62	28.1	4.20	28.4	7.13	28.6
Uniform+EVS	0.25	**2.02**	62.3	**2.27**	**73.8**	**2.78**	**84.0**	**3.83**	**87.8**	**5.97**	**89.8**
BOLT+EVS	0.25	25.89	16.0	36.43	22.2	65.07	25.0	125.81	26.7	245.07	27.9
IPB (Ours)	0.25	2.02	**62.4**	2.37	73.4	2.97	83.5	4.05	87.5	6.45	89.6
Δ vs. BOLT+EVS		-92.18%	+289.72%	-93.50%	+230.54%	-95.44%	+233.36%	-96.78%	+227.47%	-97.37%	+220.93%
Uniform+EVS	0.50	2.10	**41.6**	**2.42**	**46.3**	**3.04**	**50.0**	**4.34**	**51.2**	**6.87**	**51.9**
BOLT+EVS	0.50	26.31	14.0	35.58	19.1	58.84	24.0	122.25	25.3	237.63	27.9
IPB (Ours)	0.50	**2.10**	41.5	2.50	46.1	3.29	49.8	4.54	51.1	7.21	51.8
Δ vs. BOLT+EVS		-92.04%	+196.89%	-92.97%	+141.12%	-94.41%	+107.03%	-96.29%	+101.83%	-96.97%	+85.83%
Uniform+EVS	0.75	**2.15**	31.1	**2.52**	**33.7**	**3.32**	**35.5**	**4.79**	**36.1**	**8.05**	**36.5**
BOLT+EVS	0.75	25.56	13.4	36.91	16.1	46.27	22.6	104.74	23.6	176.92	24.5
IPB (Ours)	0.75	2.16	**31.2**	2.68	33.6	3.54	35.5	5.17	36.1	8.64	36.2
Δ vs. BOLT+EVS		-91.56%	+132.60%	-92.74%	+108.38%	-92.35%	+57.01%	-95.06%	+52.93%	-95.12%	+47.69%
LLaVA-OV-7B, K = 32											
Uniform (no pruning)	1.00	5.92	41.7	9.94	44.7	17.83	46.1	26.94	46.7	35.34	46.3
AoT	0.27	1.95	117.8	3.04	142.7	5.22	158.9	OOM		OOM	
IPB (Ours)	0.27	**1.13**	**154.3**	**1.83**	**186.3**	**3.14**	**210.2**	**5.50**	**227.5**	**8.34**	**233.2**
Δ vs. AoT		-41.95%	+30.93%	-39.62%	+30.55%	-39.95%	+32.30%	–	–	–	–
AoT	0.47	2.94	82.1	4.93	93.8	8.86	102.0	15.48	106.0	22.91	107.2
IPB (Ours)	0.47	**2.03**	**99.5**	**3.35**	**113.9**	**6.02**	**124.4**	**10.47**	**129.6**	**15.08**	**130.5**
Δ vs. AoT		-30.84%	+21.18%	-32.02%	+21.46%	-32.07%	+22.01%	-32.40%	+22.22%	-34.20%	+21.72%
AoT	0.80	5.66	50.1	9.11	54.4	15.87	56.8	27.38	57.9	34.71	57.8
IPB (Ours)	0.80	**4.99**	**54.3**	**7.54**	**59.3**	**13.11**	**62.0**	**22.69**	**63.3**	**29.07**	**63.1**
Δ vs. AoT		-11.91%	+8.33%	-17.19%	+8.84%	-17.37%	+9.20%	-17.11%	+9.26%	-16.26%	+9.30%

filter, and $*$ denotes convolution. Let $\Pi_{h,w}$ denote the spatial support of patch (h, w). We restrict motion vectors and textures to each patch as $V_{i,h,w} = V_i|_{\Pi_{h,w}}$ and $R_{i,h,w} = R_i|_{\Pi_{h,w}}$. The patch score combines motion and texture for P/B frames and uses texture alone for I-frames, which expose no inter-frame motion vectors:

$$\rho_{i,h,w} = \begin{cases} w_{\text{mv}}\|V_{i,h,w}\|_{2,1} + w_{\text{tex}}\|R_{i,h,w}\|_1, & \text{type}_i \in \{\text{P}, \text{B}\}, \\ w_{\text{tex}}\|R_{i,h,w}\|_1, & \text{type}_i = \text{I}. \end{cases} \tag{14}$$

Here the motion and texture terms sum normalized pixel-level motion magnitudes and Sobel magnitudes within each patch; channel normalization is specified in the Appendix. For every selected frame, we retain $k_p = \lceil r_p H_p W_p \rceil$ patches, where r_p is the requested patch-retention ratio. Let $\Lambda = \{1, \ldots, H_p\} \times \{1, \ldots, W_p\}$ denote the patch-index set. The retained set and mask are

$$\mathcal{K}_i^\star = \arg \max_{\substack{\mathcal{K} \subseteq \Lambda \\ |\mathcal{K}| = k_p}} \sum_{(h,w) \in \mathcal{K}} \rho_{i,h,w}, \mathcal{M}_{i,h,w} \quad = \mathbf{1}[(h, w) \in \mathcal{K}_i^\star], \qquad i \in S^\star. \tag{15}$$

Patch masking therefore operates as a training-free, plug-in pre-encoder module that complements temporal frame selection within a unified visual input selection pipeline. Detailed construction of ρ, R, and settings is provided in the Appendix.

4 Experiments

4.1 Experimental Settings

We evaluate IPB in two settings. First, we evaluate serving latency and throughput under concurrent requests. On Qwen3-VL-8B (Bai et al., 2025), using a single H100 NVL (96 GB) GPU, we compare IPB with BOLT (Liu et al., 2025) frame selection followed by EVS (Bagrov et al., 2025) token pruning and uniform frame sampling followed by EVS. On LLaVA-OV-7B (Li et al., 2024a), using a single H100 (80 GB) GPU, we compare IPB with AoT (Li et al., 2026) under approximately matched visual-token budgets. Since joint frame-selection and token-pruning baselines are not standardized, BOLT+EVS provides a strong composed baseline by combining competitive frame selection with efficient token pruning.

Table 2: **Accuracy–compute trade-off on Qwen3-VL-4B.** Accuracy (%) and total pipeline TFLOPs are reported, with changes relative to Uniform+EVS at the same visual-token budget r_p with $K = 64$. Accuracy differences are in percentage points; FLOPs differences are relative percentages. Green indicates improvement and red degradation. Best values in each group are **boldfaced**.

Method	r_p	Video-MME		LongVideoBench		MVBench		EgoSchema-subset	
		Acc.↑ (Δpp)	TFLOPs↓(Δ%)	Acc.↑ (Δpp)	TFLOPs↓(Δ%)	Acc.↑ (Δpp)	TFLOPs↓(Δ%)	Acc.↑ (Δpp)	TFLOPs↓(Δ%)
Uniform+EVS	0.25	59.52(0.0)	99.6(0.0)	57.67(0.0)	104.2(0.0)	63.83(0.0)	54.7(0.0)	63.40(0.0)	50.7(0.0)
BOLT+EVS	0.25	**62.85**(+3.3)	263.0(+164.0)	**58.94**(+1.3)	221.3(+112.5)	54.50(-9.3)	57.0(+4.1)	64.40(+1.0)	79.5(+56.8)
IPB (Ours)	0.25	58.74(-0.8)	**55.5**(-44.3)	56.99(-0.7)	**58.0**(-44.3)	**65.53**(+1.7)	**30.6**(-44.0)	**64.80**(+1.4)	**28.4**(-44.0)
Uniform+EVS	0.375	59.85(0.0)	119.7(0.0)	57.82(0.0)	125.1(0.0)	64.47(0.0)	65.7(0.0)	**65.40**(0.0)	60.9(0.0)
BOLT+EVS	0.375	**63.30**(+3.4)	283.1(+136.6)	**61.03**(+3.2)	242.3(+93.6)	56.27(-8.2)	67.9(+3.4)	64.80(-0.6)	89.7(+47.3)
IPB (Ours)	0.375	62.00(+2.1)	**82.9**(-30.7)	59.98(+2.2)	**86.7**(-30.7)	**67.15**(+2.7)	**45.6**(-30.6)	65.00(-0.4)	**42.3**(-30.5)
Uniform+EVS	0.5	61.81(0.0)	139.7(0.0)	60.81(0.0)	146.1(0.0)	65.65(0.0)	76.6(0.0)	65.60(0.0)	71.0(0.0)
BOLT+EVS	0.5	**64.22**(+2.4)	303.1(+117.0)	**62.38**(+1.6)	263.2(+80.2)	56.77(-8.9)	78.8(+2.9)	65.40(-0.2)	99.8(+40.6)
IPB (Ours)	0.5	63.74(+1.9)	**110.3**(-21.0)	60.21(-0.6)	**115.3**(-21.0)	**67.23**(+1.6)	**60.5**(-21.0)	**67.60**(+2.0)	**56.1**(-20.9)
Uniform+EVS	0.625	62.30(0.0)	159.7(0.0)	60.51(0.0)	167.1(0.0)	65.75(0.0)	87.5(0.0)	66.20(0.0)	81.1(0.0)
BOLT+EVS	0.625	**64.85**(+2.6)	323.1(+102.3)	**62.53**(+2.0)	284.2(+70.1)	58.38(-7.4)	89.8(+2.6)	66.00(-0.2)	109.9(+35.5)
IPB (Ours)	0.625	63.89(+1.6)	**137.7**(-13.8)	62.15(+1.6)	**144.0**(-13.8)	**67.28**(+1.5)	**75.5**(-13.8)	**67.40**(+1.2)	**70.0**(-13.7)
Uniform+EVS	0.75	63.41(0.0)	179.8(0.0)	60.96(0.0)	188.0(0.0)	66.38(0.0)	98.5(0.0)	66.80(0.0)	91.3(0.0)
BOLT+EVS	0.75	**65.26**(+1.9)	343.2(+90.9)	**62.68**(+1.7)	305.2(+62.3)	58.95(-7.4)	100.7(+2.3)	66.40(-0.4)	120.1(+31.6)
IPB (Ours)	0.75	63.33(-0.1)	**165.1**(-8.2)	61.85(+0.9)	**172.6**(-8.2)	**67.62**(+1.2)	**90.5**(-8.1)	**67.40**(+0.6)	**83.8**(-8.1)

For the Qwen3-VL-8B stress test, each request uses the same synthetic 30-minute video. We set the visual input budget using a 1-fps reference, while IPB performs frame selection on the original codec timeline and retains 64 frames for downstream inference. We fix the input sequence length to 300 and the output sequence length to 128, and vary concurrency and retained ratio to measure end-to-end throughput and latency under increasing system load. All experiments are conducted using lmms-eval (Zhang et al., 2025) and vLLM (Kwon et al., 2023). Second, we evaluate end-to-end accuracy on Video-MME (Fu et al., 2025), MVBench (Li et al., 2024b), LongVideoBench (Wu et al., 2024), and the EgoSchema subset (Mangalam et al., 2023). We further evaluate frame-selection performance separately on the aforementioned benchmarks, additionally including ActivityNet-QA (Yu et al., 2019) with multiple video VLM backbones, including LLaVA-OV (Li et al., 2024a), LLaVA-Video (Zhang et al., 2024), and Qwen3-VL (Bai et al., 2025). For frame-selection comparison, we use uniform sampling and prior baselines including BOLT (Liu et al., 2025), AKS (Tang et al., 2025), and FOCUS (Zhu et al., 2025) under matched frame budgets. Since these prior methods do not report results on ActivityNet-QA (Yu et al., 2019) or MVBench (Li et al., 2024b), we provide reproduced results on those benchmarks where available.

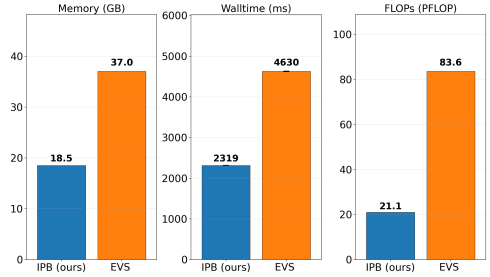


Figure 4: Comparison of computational cost of the vision encoder between IPB and EVS (Bagrov et al., 2025). IPB reduces resource usage across all metrics: peak activation memory, walltime, and FLOPs.

4.2 Results

System Efficiency Table 1 reports our main results. In these experiments, we vary concurrency C and the retained ratio r_p to investigate system throughput and mean Time to First Token (TTFT).

Table 1 demonstrates the serving advantage of IPB under increasing concurrency, while Table 2 reports the end-to-end accuracy–compute trade-off. At r_p=0.25 and C=64, IPB reduces TTFT by a factor of 38.0 relative to BOLT+EVS on Qwen3-VL-8B. On LLaVA-OV-7B, IPB also reduces TTFT by 40% relative to AoT at r_p=0.27 and C=16 under approximately matched visual-token budgets.

This gap is structural. Conventional model-based scoring and attention-based pruning are typically situated on the GPU critical path, making their overhead additive to VLM inference and sensitive to increased concurrency. In contrast, IPB shifts selection to the CPU prior to the vision encoder (fig. 1a), enabling the selection process to overlap with downstream GPU inference. This decoupling reduces selection overhead on the GPU critical path and provides an advantage in the end-to-end trade-off between accuracy and compute (table 2).

Table 3: **Comparison of our proposed IPB with frame-selection models.** The comparison in this table is restricted to frame selection. All reported metrics are accuracy. Methods are grouped by additional GPU-side computation. The best result in each group is **boldfaced**.

Method	#Frame	GPU Sel. FLOPs ↓	GPU Scoring	ActivityNet QA	EgoSchema Subset	MV Bench	LongVideo Bench	Video-MME			
								Overall	Short	Medium	Long
LLaVA-OV-7B (Li et al., 2024a)											
Uniform	32	0	✗	56.6	62.2	**58.2**	56.3	58.7	70.6	56.2	49.2
IPB (Ours)	32	0	✗	**59.0**	**63.0**	58.0	**56.6**	**59.8**	**72.0**	**57.7**	**49.8**
BOLT	32	4.9T	✓	**58.6**	**64.0**	53.7	59.6	**59.9**	**70.1**	**60.0**	49.6
AKS	32	12.4T	✓	–	–	**56.9**	59.3	58.4	–	–	–
FOCUS	32	14.8T	✓	57.9	58.2	54.1	**60.7**	58.9	69.3	55.8	**51.6**
LLaVA-Video-7B (Zhang et al., 2024)											
Uniform	64	0	✗	**65.9**	**60.8**	60.15	**60.7**	63.6	75.7	**61.8**	53.2
IPB (Ours)	64	0	✗	**65.9**	60.4	**61.3**	60.4	**63.9**	**76.3**	61.4	**53.8**
BOLT	64	4.9T	✓	64.6	**60.8**	55.2	61.3	64.5	75.2	63.4	54.8
AKS	64	12.4T	✓	–	–	**58.7**	62.7	65.3	–	–	–
FOCUS	64	28.0T	✓	**65.2**	58.4	56.5	**63.5**	**65.4**	**76.5**	**63.5**	**56.1**
Qwen3-VL-4B (Bai et al., 2025)											
Uniform	64	0	✗	59.7	**69.4**	**67.9**	61.3	65.1	76.1	**64.8**	54.3
IPB (Ours)	64	0	✗	**62.3**	68.4	67.4	**62.4**	**65.5**	**77.8**	64.0	**54.7**

At matched visual-token budgets, IPB requires less total compute than Uniform+EVS and BOLT+EVS across all evaluated reduced-budget settings. Relative to Uniform+EVS, IPB generally maintains or improves accuracy, with consistent gains on MVBench and gains at most budgets on Video-MME, LongVideoBench, and EgoSchema-subset (Table 2). In contrast, BOLT+EVS consistently increases total FLOPs due to its additional model-side frame-selection cost, while achieving higher accuracy in several settings.

AoT additionally encounters OOM at the evaluated settings with $r_p \leq 0.40$ and $C \geq 32$ under the default serving configuration. Figure 4 further isolates the benefit of pre-encoder sparsification. Because IPB resolves token selection before visual encoding, it reduces peak activation memory, encoder walltime, and FLOPs relative to EVS, whose pruning requires model-side visual processing.

Table 4: Per-task behavior of IPB frame selection.

Task	LLaVA-OV-7B ($K = 32$)			Qwen3-VL-4B ($K = 64$)		
	Uni.	Ours	Δ	Uni.	Ours	Δ
MVBench						
Action Localization	**59.0**	58.0	-1.0	56.5	**57.5**	+1.0
Action Sequence	71.0	**71.5**	+0.5	79.5	79.5	0.0
Object Shuffle	**35.5**	35.0	-0.5	**40.0**	35.5	-4.5
Video-MME						
Action Reasoning	54.4	**56.1**	+1.8	58.2	**59.6**	+1.4
Temporal Reasoning	41.8	**42.9**	+1.1	44.6	**47.5**	+2.8
Attribute Perception	**73.0**	70.7	-2.3	78.4	**78.8**	+0.5
LongVideoBench						
Event before/after Text	47.9	**52.1**	+4.1	60.3	**61.6**	+1.4
Event before/after Event	60.6	**66.0**	+5.3	62.8	**67.0**	+4.3
Object before/after Object	**50.0**	47.0	-3.0	**66.7**	65.2	-1.5

Benchmark Evaluation We separately evaluate frame selection, a key component of IPB, under matched frame budgets (Tables 3, 4). IPB achieves competitive accuracy against prior frame-selection methods without GPU-side scoring and improves over uniform sampling across several benchmarks. The per-task breakdown in Table 4 highlights representative tasks, including gains on event-centric reasoning and losses on object shuffling. These examples illustrate the task-dependent effect of codec priors; full per-task results appear in Appendix Table 12.

Visualization The qualitative results are consistent with fig. 5: the codec-derived density $\hat{u}(t)$ concentrates the temporal budget on motion-rich, event-relevant segments rather than spreading it uniformly, and the resulting patch masks further align with visually informative regions within those selected frames. As shown in the figure, IPB follows both the salient temporal intervals and the important spatial evidence more faithfully, whereas uniform sampling is often too coarse to capture the localized cues on which event-centric questions depend.

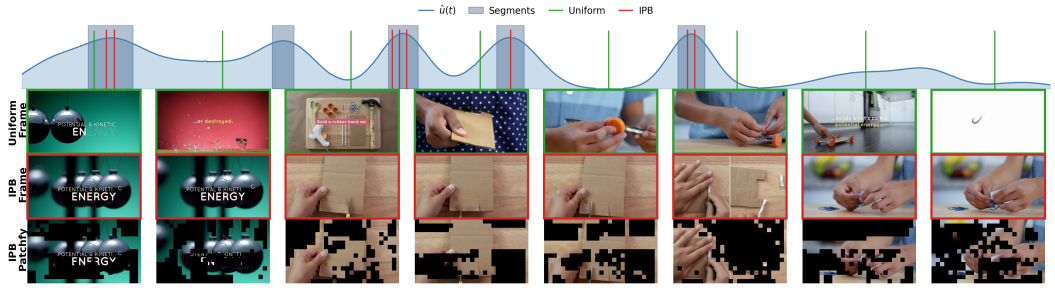


Figure 5: **Visualization of IPB frame selection and patchification.** IPB concentrates frame samples on informative temporal segments identified by the $\hat{u}(t)$ while patch masking preserves spatially informative regions within the selected frames. Uniform sampling is shown for comparison.

Table 5: **Component ablation.** LLaVA-OV-7B on Video-MME, $K = 32$.

Method	Avg.	Spatial Perception	OCR Problem	Counting Problem
Uniform	<u>58.7</u>	57.4	60.4	36.9
IPB (Ours)	**59.8**	**68.5**	<u>61.9</u>	38.4
Coding Score	57.9	59.3	59.7	35.1
Motion Gate	54.6	50.0	60.4	33.6
I-frame	56.6	57.4	**62.6**	**39.9**
PktSize	56.6	<u>61.1</u>	**62.6**	<u>38.8</u>

Table 6: **Encoding robustness.** Qwen3-VL-4B on Video-MME short, $K = 32$.

Encoding	CRF	GOP	Uniform	IPB	Δ	Fallback
Original	–	–	74.0	75.3	+1.3	–
H.264	18	60	72.8	72.9	+0.1	–
H.264	23	60	74.1	74.6	+0.5	–
H.264	28	60	74.2	75.0	+0.8	–
H.264	23	30	73.6	73.6	0.0	299/300
H.264	23	120	74.3	75.3	+1.0	–
H.265	23	120	74.7	75.4	+0.7	–

5 Discussion

Ablation Study Table 5 examines several plausible CPU-side alternatives to the full IPB utility. Starting from compressed-domain cues alone, we consider relying on a single signal such as coding score, motion gate in eq. (7), I-frame anchoring eq. (3), and packet size ($|b_i|$). I-frame and Packet-size based selection methods remain competitive on specific categories (OCR Problem), confirming that each cue is locally useful. However, the overall accuracy is inconsistent across tasks, reflecting the narrow bias. The full IPB formulation is robust because it combines coding and motion information into a shared prior instead of relying on a single compressed-domain signal. Bold and underlined entries mark the best and second-best values; The full breakdown appears in Appendix Table 13. The one-at-a-time sensitivity study in Appendix Table 14 reports ablation accuracies within 1.60 and 0.69 percentage points of the main-table IPB results on Video-MME and LongVideoBench, respectively.

Limitation IPB is query-agnostic and relies on codec-side redundancy rather than semantic relevance, making it less reliable when query-critical evidence is weakly reflected in coding cost or motion. Since codec statistics also vary with the encoding configuration, we evaluate robustness by re-encoding the same Video-MME short videos while varying the codec, Constant Rate Factor (CRF), and GOP length. Table 6 reports accuracy (%) on 300 videos (900 questions); Δ is IPB minus Uniform in percentage points. It shows that IPB matches or exceeds uniform sampling across the evaluated encoding configurations. With H.264 and GOP 30, 299 videos fall back to uniform under the two-second GOP eligibility rule. The result indicates sensitivity to GOP structure, while encoding effects remain coupled with the selector configuration.

6 Conclusion

We introduced **Information Priors from Bitstream (IPB)**, a training-free codec-guided sparsification framework that performs temporal frame selection and spatial patch masking before GPU-side visual encoding. By deriving visual input selection directly from compressed-domain signals, IPB avoids additional model-side scoring, removes sparsification from the GPU critical path, and enables CPU–GPU overlap across requests. Experiments show that IPB preserves competitive accuracy across diverse video benchmarks while achieving 3.2× higher aggregate throughput and 38.0× lower TTFT at high concurrency. More broadly, our findings indicate that codec-derived priors provide a practical foundation for deployable long-video VLM inference, where efficiency depends not only on reducing tokens, but on resolving selection before expensive downstream computation.

AI USE STATEMENT

Generative AI tools were used as assistive tools for literature search, language refinement, LaTeX formatting, and feedback on manuscript organization, presentation, and consistency of notation. All AI-assisted suggestions were reviewed by the authors, and all experimental measurements, numerical results, implementations, and scientific claims were independently reviewed and validated by the authors. Generative AI was not used to generate experimental datasets or experimental measurements. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies efficient video understanding using established public benchmark datasets and does not involve human-subject experiments, the collection of sensitive personal data, or deployment in high-risk decision-making settings.

REPRODUCIBILITY STATEMENT

The main text describes the proposed method, datasets, baselines, hardware, and evaluation protocol. The appendix provides detailed hyperparameter settings, implementation details, and additional experimental results. Section A.7 provides Python-style listings of the core frame- and patch-selection procedures, with hyperparameter values in table 7.

REFERENCES

Natan Bagrov, Eugene Khvedchenia, Borys Tymchenko, Shay Aharon, Lior Kadoch, Tomer Keren, Ofri Masad, Yonatan Geifman, Ran Zilberstein, Tuomas Rintamaki, et al. Efficient video sampling: Pruning temporally redundant tokens for faster vlm inference. *arXiv preprint arXiv:2510.14624*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Shyamal Buch, Arsha Nagrani, Anurag Arnab, and Cordelia Schmid. Flexible frame selection for efficient video reasoning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 29071–29082, 2025.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2024.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24108–24118, 2025.

Xinyue Hao, Gen Li, Shreyank N Gowda, Robert B Fisher, Jonathan Huang, Anurag Arnab, and Laura Sevilla-Lara. Principles of visual tokens for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 21254–21264, 2025.

Peng Jin, Ryuichi Takanobu, Wancai Zhang, Xiaochun Cao, and Li Yuan. Chat-univi: Unified visual representation empowers large language models with image and video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13700–13710, 2024a.

Yang Jin, Zhicheng Sun, Kun Xu, Liwei Chen, Hao Jiang, Quzhe Huang, Chengru Song, Yuliang Liu, Di Zhang, Yang Song, et al. Video-lavit: Unified video-language pre-training with decoupled visual-motional tokenization. *arXiv preprint arXiv:2402.03161*, 2024b.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024a.

Jinlong Li, Liyuan Jiang, Haonan Zhang, and Nicu Sebe. Token reduction via local and global contexts optimization for efficient video large language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22195–22206, 2024b.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 5971–5984, 2024.

Kevin Lin, Faisal Ahmed, Linjie Li, Chung-Ching Lin, Ehsan Azarnasab, Zhengyuan Yang, Jianfeng Wang, Lin Liang, Zicheng Liu, Yumao Lu, et al. Mm-vid: Advancing video understanding with gpt-4v(ision). *arXiv preprint arXiv:2310.19773*, 2023.

Shuming Liu, Chen-Lin Zhang, Chen Zhao, and Bernard Ghanem. End-to-end temporal action detection with 1b parameters across 1000 frames. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18591–18601, 2024.

Shuming Liu, Chen Zhao, Tianqi Xu, and Bernard Ghanem. Bolt: Boost large vision-language model without training for long-form video understanding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 3318–3327, 2025.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12585–12602, 2024.

Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. *Advances in Neural Information Processing Systems*, 36:46212–46244, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Sayan Deb Sarkar, Rémi Pautrat, Ondrej Miksik, Marc Pollefeys, Iro Armeni, Mahdi Rad, and Mihai Dusmanu. Cope-videolm: Codec primitives for efficient video language models. *arXiv e-prints*, pp. arXiv–2602, 2026.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22857–22867, 2025.

Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

Hui Sun, Shiyin Lu, Huanyu Wang, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Ming Li. Mdp3: A training-free approach for list-wise frame selection in video-llms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 24090–24101, 2025.

Feilong Tang, Xiang An, Yunyao Yan, Yin Xie, Bin Qin, Kaicheng Yang, Yifei Shen, Yuanhan Zhang, Chunyuan Li, Shikun Feng, et al. Onevision-encoder: Codec-aligned sparsity as a foundational principle for multimodal intelligence. *arXiv preprint arXiv:2602.08683*, 2026.

Xi Tang, Jihao Qiu, Lingxi Xie, Yunjie Tian, Jianbin Jiao, and Qixiang Ye. Adaptive keyframe sampling for long video understanding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 29118–29128, 2025.

Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. Dycoke: Dynamic compression of tokens for fast video large language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 18992–19001, 2025.

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. Stop looking for "important tokens" in multimodal language models: Duplication matters more. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9972–9991, 2025.

Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7): 560–576, 2003.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. *Advances in Neural Information Processing Systems*, 37:28828–28857, 2024.

Zhuojie Wu, Shijie Wang, and Xin Yu. Metom: Metadata-guided token merging for efficient video llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10441–10450, June 2026.

Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 9127–9134, 2019.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11975–11986, 2023.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, et al. Lmms-eval: Reality check on the evaluation of large multimodal models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 881–916, 2025.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Llava-video: Video instruction tuning with synthetic data. *arXiv preprint arXiv:2410.02713*, 2024.

Zirui Zhu, Hailun Xu, Yang Luo, Yong Liu, Kanchan Sarkar, Zhenheng Yang, and Yang You. Focus: Efficient keyframe selection for long video understanding. *arXiv preprint arXiv:2510.27280*, 2025.

Yuanhao Zou, Shengji Jin, Andong Deng, Youpeng Zhao, Jun Wang, and Chen Chen. Air: Enabling adaptive, iterative, and reasoning-based frame selection for video question answering. *arXiv preprint arXiv:2510.04428*, 2025.

Yulin Zou, Yan Chen, Wenyan Chen, JooYoung Park, Shivaraman Nitin, Luo Tao, Francisco Romero, and Dmitrii Ustiugov. Codecsight: Leveraging video codec signals for efficient streaming vlm inference. *arXiv e-prints*, pp. arXiv–2604, 2026.

A APPENDIX

The appendix includes hyperparameter settings, additional discussions extended from the main paper, and further experimental results. Python-style listings in section A.7 provide the core frame- and patch-selection procedures; hyperparameter values are specified in table 7.

A.1 NOTATION

The notation table summarizes the quantities used in frame selection, patch selection, and serving evaluation.

Table 7: **Notation and default settings.** Frame selection, patch selection, and serving evaluation.

Symbol	Definition	Default or note
Input and budgets		
T, K	Candidate and selected frame counts	$\lvert S^\star\rvert = K \le T$; frame retention is K/T.
$H_p, W_p;\ k_p$	Patch-grid size; retained patches per frame	$k_p = \lceil r_p H_p W_p \rceil$.
r_p	Per-frame patch-retention ratio (eq. (15))	Same top-k_p rule for I/P/B; distinct from K/T.
F	Retained frames in the AoT serving comparison	Frame retention: $F/32$; serving retention labels denote approximately matched overall visual-token budget ratios.
Frame selection		
Γ_g, L_g	GOP g and its frame count	$L_g = \lvert\Gamma_g\rvert$.
$\lvert b_i\rvert;\ m_i$	Frame packet size; motion mass	Bytes; block-area-weighted motion magnitudes.
B_g, M_g	GOP-average packet size and motion mass	Normalized by positive frame-level medians.
$u_g, \hat{u}_i$	GOP score and frame-level temporal density	Equations (7) and (8).
$\Omega;\ C_j, \mu_j$	High-density support; segment and its mass	$\Omega = \{i : \hat{u}_i \ge \tau\}$; $\mu_j = \sum_{i \in C_j \cap \Omega} \hat{u}_i$.
$S^\star$	Selected frame-index set	Budget allocation in eq. (13).
Patch selection		
$(h, w), \Pi_{h,w}$	Patch coordinate and pixel support	$H_p \times W_p$ grid; patch size $p = 16$ pixels.
$V_{i,h,w}, R_{i,h,w}$	Patch motion vectors and Sobel texture	Scores: $\lVert V_{i,h,w}\rVert_{2,1}$ and $\lVert R_{i,h,w}\rVert_1$.
$\rho_{i,h,w}$	Patch score	Equation (14); I-frames use texture only.
$\mathcal{K}_i^\star, \mathcal{M}_i$	Retained patch set and binary mask	Top-k_p selection on each selected frame.
Hyperparameters		
`min_gop_sec`	Minimum scored GOP duration	2 s; shorter GOPs receive score zero.
β, κ	Motion-gate slope and density spread	2, 0.5.
q, τ	Adaptive quantile and density threshold	q: 0.45/0.85/0.45; $\tau = \mathrm{Quantile}_q(\hat{u}_i > 0)$.
b	Duration-based reference length (frames)	$b = \max(1, \lfloor D_{\mathrm{sec}} \rfloor)$; $D_{\mathrm{sec}} = T/\mathrm{fps}$.
$\Delta_{\mathrm{gap}}, \Delta_{\min}$	Merge-gap and minimum-segment thresholds	$3b$, b; independent of K.
d_m	In-/out-of-segment sampling-density ratio	2.5.
$w_{\mathrm{mv}}, w_{\mathrm{tex}}$	Motion and texture fusion weights	1, 1 for P/B fusion; I-frames use texture only.
`percentile`	Channel-normalization percentile	95 for motion and texture.
`mv_block`, `mv_unit_div`	Codec grid stride and vector-unit divisor	16 pixels, 4; componentwise median compensation.
System evaluation		
C	Concurrent requests	Serving concurrency, distinct from segment C_j.
TTFT	Time from request to first output token	Mean latency in seconds.
Throughput	Aggregate generated tokens per second	Across all requests; not per-user decode rate.

A.2 HYPERPARAMETER DETAILS

In this section, we report the coefficients, thresholds, and default values used by the IPB pipeline. The main paper specifies the algorithm symbolically; here we list numerical settings.

A.2.1 FRAME SELECTOR

GOP Utility GOPs shorter than $\max(1, \lfloor 2\,\mathrm{fps} \rfloor)$ frames receive zero utility (`min_gop_sec`=2.0). We instantiate eq. (7) with motion-gate slope $\beta = 2.0$. The $\log(1 + \widetilde{B}_g)$ term keeps the coding-cost channel from dominating high-bitrate I-anchored GOPs, and the $(1 - e^{-\beta \widetilde{M}_g})$ saturating gate prevents pure-static GOPs (text overlays, slate frames) from accumulating bits without earning a high score. We use $\kappa = 0.5$ throughout (fixed across all videos and all K) in the density smoothing (eq. (8)).

Gaussian Support In eq. (8), each GOP's Gaussian contribution is evaluated only on integer frame indices in $[\max(0, \lfloor c_g - 3\sigma_g \rfloor), \min(T, \lfloor c_g + 3\sigma_g \rfloor))$ and is zero outside that interval. The upper endpoint is excluded, matching algorithm 2.

CV-adaptive τ The quantile q that sets τ in eq. (9) adapts to the coefficient of variation $\mathrm{cv} = \mathrm{std}(\{u_g : u_g > 0\})/\,\mathrm{mean}(\{u_g : u_g > 0\})$ of the positive GOP scores via the following non-monotone schedule:

$$q = \begin{cases} 0.45 & \text{if } \mathrm{cv} < 0.3 \quad \text{(low spread; small } \tau \text{ keeps plateau structure resolvable)} \\ 0.85 & \text{if } 0.3 \le \mathrm{cv} < 0.6 \quad \text{(mid spread; large } \tau \text{ for selective segmentation)} \\ 0.45 & \text{if } \mathrm{cv} \ge 0.6 \quad \text{(high spread; small } \tau \text{ retains broader temporal support)} \end{cases}$$

The threshold is $\tau = \mathrm{Quantile}_q(\{\hat{u}_i : \hat{u}_i > 0\})$. This heuristic uses a lower quantile in the low- and high-CV regimes and a higher quantile in the mid-CV regime. A higher threshold narrows the retained temporal support; it does not by itself discard the highest-density peaks. If there are no positive GOP scores or no positive density values, uniform sampling is used before computing the coefficient of variation or the quantile. Uniform sampling is also used if no segment survives post-processing.

Segment Threshold We define a duration-scaled reference length in frames,

$$D_{\mathrm{sec}} = T/\mathrm{fps}, \qquad b = \max(1, \lfloor D_{\mathrm{sec}} \rfloor), \qquad \Delta_{\min} = b, \qquad \Delta_{\mathrm{gap}} = 3b,$$

where D_{sec} is the video duration in seconds. The reference length b is a frame count, not a one-second video interval. The thresholds depend on video duration and are independent of the frame budget K.

Segment-aware Budget Allocation We instantiate $\mathrm{AllocSample}$ as follows. Let $L_{\mathrm{seg}} = \sum_{C_j \in \mathcal{C}^\star} |C_j|$ and $L_{\mathrm{out}} = T - L_{\mathrm{seg}}$. Uniform sampling is used if either region has zero length. With density multiplier $d_m = 2.5$, the target in-segment frame budget is

$$K_{\mathrm{seg}} = \max\left(|\mathcal{C}^\star|, \min\left(K - 1, \left\lfloor \frac{d_m L_{\mathrm{seg}} K}{d_m L_{\mathrm{seg}} + L_{\mathrm{out}}} \right\rceil\right)\right).$$

Per-segment budgets are independently rounded in proportion to the masses μ_j from eq. (12) and capped by segment length:

$$k_j = \min\left(|C_j|, \max\left(1, \left\lfloor \frac{K_{\mathrm{seg}} \mu_j}{\sum_{j'} \mu_{j'}} \right\rceil\right)\right).$$

Each segment $C_j = [\xi_j^-, \xi_j^+)$ is sampled using k_j uniformly spaced midpoint indices. Independent rounding means that $\sum_j k_j$ need not equal K_{seg}. Any positive remainder $K - \sum_j k_j$ is allocated to the out-of-segment gaps proportionally to gap length, with integer counts adjusted to match the remaining budget. The gaps are sampled using the same midpoint rule. Finally, selected indices are deduplicated and sorted. Missing slots are filled with unused frame indices in ascending order, and excess indices are removed by retaining the first K indices in temporal order. The final output contains exactly K unique frame indices.

A.3 Sobel Filter as Residual Proxy

Texture channel In this section, $R_{i,h,w}$ denotes the Sobel-based texture proxy used in eq. (14). The texture channel of eq. (14) (*i.e.* $\|R_{i,h,w}\|_1$) is realized as a Sobel magnitude on the decoded luma plane rather than from the codec residual itself. The luma channel is obtained from the decoded BGR frame via the OpenCV BT.601 convention $Y = 0.114\,B + 0.587\,G + 0.299\,R$, and the Sobel response is computed with the standard 3×3 separable kernels (`cv2.Sobel, ksize=3`):

$$\mathcal{S}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \qquad \mathcal{S}_y = \mathcal{S}_x^\top.$$

The per-pixel edge magnitude $E(x,y) = \sqrt{(\mathcal{S}_x * Y)^2 + (\mathcal{S}_y * Y)^2}$ is percentile-normalized at the 95-th percentile, clipped to $[0,1]$, and patch-summed into the $H_p \times W_p$ grid.

Fusion (eq. (14)). For P/B frames, we instantiate eq. (14) with $w_{\mathrm{mv}} = w_{\mathrm{tex}} = 1.0$, assigning equal weights to the two channels. For these P/B frames, the implementation uses the normalized weighted-average form $\rho_{i,h,w} = (w_{\mathrm{mv}}\|V_{i,h,w}\|_{2,1} + w_{\mathrm{tex}}\|R_{i,h,w}\|_1)/(w_{\mathrm{mv}} + w_{\mathrm{tex}})$, which is equivalent to eq. (14) for the subsequent top-k selection up to a positive scale factor. I-frames use the Sobel texture term only, following eq. (14).

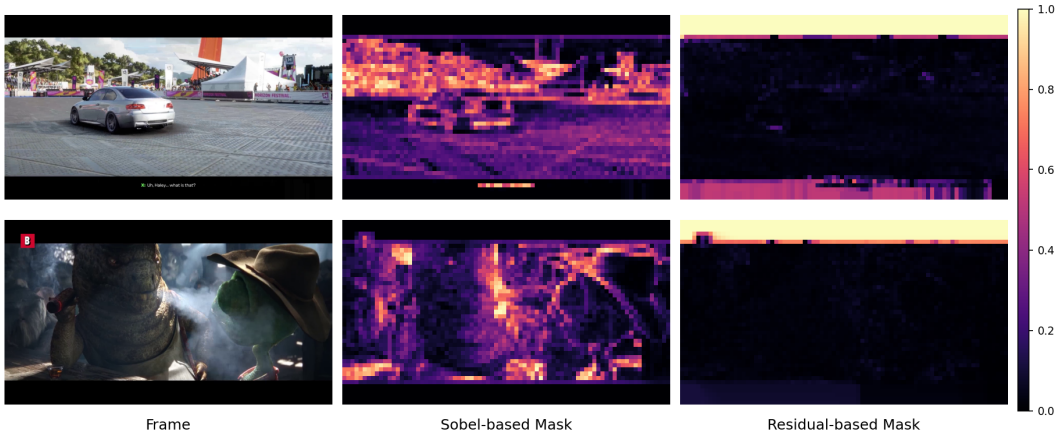


Figure 6: **Qualitative differences between Sobel- and residual-based saliency maps.** Each row is one frame; columns show the decoded RGB frame, the Sobel saliency map, and the residual saliency map. Maps are normalized to relative magnitude $[0, 1]$ and share the colorbar at right.

Sobel as a robust surrogate for the residual signal. Our spatial mask combines motion-vector mass with a per-patch spatial saliency term. In principle, the most direct choice for this spatial term is the codec residual itself, which captures the coding error remaining after motion compensation and therefore reflects the signal for which the encoder allocates rate–distortion budget. In practice, however, we use the Sobel gradient magnitude on the decoded luma plane as a runtime surrogate (fig. 6). Reliable extraction of per-block residuals requires either a custom hook into `libavcodec` or an additional decode pass, which substantially increases per-frame overhead for HEVC and introduces a brittle dependency on a specific decoder implementation. Across mixed H.264/HEVC sources and re-encoded videos, we also observed silent residual-extraction failures that collapse the codec mask to a uniform fallback. The Sobel-based surrogate avoids these issues while providing a lightweight texture cue for patch selection. We compare the flattened, fused patch-score maps using Spearman rank correlation before top-k selection. Agreement is lower on B-frames than on P-frames.

Table 8: Spearman correlation between Sobel- and residual-based fused patch-score rankings before top-k selection (r_p-independent). Both scores include the same motion term.

	Mean	Median	P-frame	B-frame
Spearman ρ	0.671	0.798	0.716	0.634

Correlation with residual-based ranking. We measure agreement between the patch rankings obtained with Sobel- and residual-based texture channels. We evaluate 100 VideoMME videos with 16 uniformly sampled P/B frames per video, yielding 1,600 sampled frames, of which 1,565 provide valid rank correlations. Holding all other components of the masking pipeline fixed—including motion-vector fusion, percentile normalization, and weighted combination—we compute Sobel-based and residual-based score maps side by side. As reported in Table 8, the two fused score maps achieve a mean Spearman rank correlation of ρ=0.671 (median 0.798), with stronger agreement on P-frames (ρ=0.716) than on B-frames (ρ=0.634). These results show agreement between the fused rankings under the evaluated settings; they do not isolate the correlation between the texture channels or establish identical top-k selections. Figure 6 illustrates qualitative differences between the two saliency maps.

A.4 Full Serving Sweeps

Tables 9 and 10 list every point of the Qwen3-VL-8B stress test summarized in the top panel of table 1, adding the uniform-sampling+EVS pipeline, the r_p=1 (no pruning) rows for all three frame selectors, and the per-user decode rate. Table 11 lists the complete AoT (Li et al., 2026) comparison on LLaVA-OV-7B summarized in the bottom panel of table 1 over all six approximately token-matched retention levels. Here the serving retention labels denote approximately matched overall

visual-token budget ratios, distinct from the per-frame patch-retention parameter r_p in eq. (15); in the AoT comparison, IPB retains F of 32 frames while AoT reduces tokens from all 32.

Table 9: **Full concurrency sweep for Table 1** (Qwen3-VL-8B, H100 NVL 96 GB, K=64, 30-minute 720p source, ISL 300 / OSL 128). Mean TTFT in seconds and aggregate output-token throughput for every method, retention ratio r_p, and concurrency C. At r_p=1 no token pruning is applied, so the rows differ only in the frame selector. Within each r_p group the best value per column is **boldfaced**. †54 of 192 requests failed (client-side 900 s timeout); statistics are over the 138 completed requests.

Method	r_p	Mean TTFT (s) ↓						Throughput (tok/s) ↑					
		C=1	C=4	C=8	C=16	C=32	C=64	C=1	C=4	C=8	C=16	C=32	C=64
Uniform	1.00	1.37	**1.59**	**1.89**	**2.62**	**4.20**	**7.13**	18.8	25.9	27.2	28.1	28.4	28.6
BOLT	1.00	22.44	24.89	35.05	45.69	73.11	134.94†	4.6	12.6	15.6	19.2	20.2	18.7
IPB (Ours)	1.00	**1.36**	1.60	2.03	2.73	4.41	7.44	**18.9**	**26.1**	**27.3**	**28.2**	**28.6**	**28.8**
Δ vs. BOLT		-93.95%	-93.56%	-94.21%	-94.03%	-93.96%	-94.49%	+310.18%	+107.70%	+75.33%	+46.82%	+41.62%	+53.74%
Uniform+EVS	0.25	**1.92**	**2.02**	**2.27**	**2.78**	**3.83**	**5.97**	29.8	62.3	**73.8**	**84.0**	**87.8**	**89.8**
BOLT+EVS	0.25	23.15	25.89	36.43	65.07	125.81	245.07	5.0	16.0	22.2	25.0	26.7	27.9
IPB (Ours)	0.25	1.92	2.02	2.37	2.97	4.05	6.45	**29.9**	**62.4**	73.4	83.5	87.5	89.6
Δ vs. BOLT+EVS		-91.72%	-92.18%	-93.50%	-95.44%	-96.78%	-97.37%	+494.50%	+289.72%	+230.54%	+233.36%	+227.47%	+220.93%
Uniform+EVS	0.50	**1.95**	2.10	**2.42**	**3.04**	**4.34**	**6.87**	**23.9**	**41.6**	**46.3**	**50.0**	**51.2**	**51.9**
BOLT+EVS	0.50	23.00	26.31	35.58	58.84	122.25	237.63	4.8	14.0	19.1	24.0	25.3	27.9
IPB (Ours)	0.50	1.97	**2.10**	2.50	3.29	4.54	7.21	23.8	41.5	46.1	49.8	51.1	51.8
Δ vs. BOLT+EVS		-91.45%	-92.04%	-92.97%	-94.41%	-96.29%	-96.97%	+390.49%	+196.89%	+141.12%	+107.03%	+101.83%	+85.83%
Uniform+EVS	0.75	**1.96**	**2.15**	**2.52**	**3.32**	**4.79**	**8.05**	**19.9**	31.1	**33.7**	**35.5**	**36.1**	**36.5**
BOLT+EVS	0.75	23.02	25.56	36.91	46.27	104.74	176.92	4.7	13.4	16.1	22.6	23.6	24.5
IPB (Ours)	0.75	1.96	2.16	2.68	3.54	5.17	8.64	19.9	**31.2**	33.6	35.5	36.1	36.2
Δ vs. BOLT+EVS		-91.48%	-91.56%	-92.74%	-92.35%	-95.06%	-95.12%	+327.07%	+132.60%	+108.38%	+57.01%	+52.93%	+47.69%

Table 10: **Per-user decode rate** (tokens/s per request, $= 1000/\text{mean TPOT}_{\text{ms}}$) for the sweep in Table 9. The rate describes decoding after the first token; aggregate throughput in Table 9 is total generated tokens divided by benchmark wall-clock time.

Method	r_p	C=1	C=4	C=8	C=16	C=32	C=64
Uniform	1.00	23.7	7.2	3.7	1.8	0.9	0.5
BOLT	1.00	23.9	9.0	4.9	2.3	1.1	0.5
IPB (Ours)	1.00	23.7	7.3	3.7	1.9	0.9	0.5
Δ vs. BOLT		-0.66%	-19.74%	-24.70%	-18.72%	-11.40%	-6.77%
Uniform+EVS	0.25	54.3	21.5	11.5	6.1	3.1	1.6
BOLT+EVS	0.25	55.0	21.6	16.9	12.9	14.8	14.1
IPB (Ours)	0.25	54.2	21.6	11.5	6.1	3.1	1.6
Δ vs. BOLT+EVS		-1.30%	-0.30%	-31.60%	-52.60%	-79.01%	-88.90%
Uniform+EVS	0.50	37.6	12.9	6.7	3.5	1.7	0.9
BOLT+EVS	0.50	37.8	12.6	7.8	6.4	5.7	6.8
IPB (Ours)	0.50	37.6	13.0	6.7	3.5	1.7	0.9
Δ vs. BOLT+EVS		-0.53%	+2.56%	-13.84%	-45.38%	-69.45%	-87.19%
Uniform+EVS	0.75	28.8	9.2	4.7	2.4	1.2	0.6
BOLT+EVS	0.75	28.9	11.1	5.4	3.2	2.1	0.9
IPB (Ours)	0.75	28.8	9.3	4.7	2.4	1.2	0.6
Δ vs. BOLT+EVS		-0.48%	-16.33%	-12.40%	-24.27%	-42.36%	-32.05%

A.5 Additional Results

The full per-task results in table 12 and full component ablation in table 13 extend the main-text comparisons. These tables show the task-dependent gains and losses of IPB and the ablation results across all Video-MME categories.

Robustness to encoding settings. Table 6 reports the encoding sweep. With H.264 and GOP 30, 299 of 300 videos use uniform fallback. This result should be interpreted together with the two-second minimum scored GOP duration in section A.2.1; it does not isolate encoding effects from the selector's GOP eligibility rule.

Table 11: **Full concurrency sweep for Table 1** (LLaVA-OV-7B, H100 80 GB). For each retention level r_p the token budget is matched: IPB keeps F of 32 frames (196 tokens each), AoT keeps all 32 frames and merges to a comparable token count. OOM: out of GPU memory at vLLM's default budget (`gpu_memory_utilization`=0.9; the same points also fail at 0.85). Within each r_p group the better value per column is **boldfaced**.

Method	r_p	Mean TTFT (s) ↓						Throughput (tok/s) ↑					
		C=1	C=4	C=8	C=16	C=32	C=64	C=1	C=4	C=8	C=16	C=32	C=64
Uniform (no pruning)	1.00	3.03	5.92	9.94	17.83	26.94	35.34	27.9	41.7	44.7	46.1	46.7	46.3
AoT	0.27	1.64	1.95	3.04	5.22	OOM	OOM	48.2	117.8	142.7	158.9	OOM	OOM
IPB (Ours)	0.27	**0.99**	**1.13**	**1.83**	**3.14**	**5.50**	**8.34**	**63.7**	**154.3**	**186.3**	**210.2**	**227.5**	**233.2**
Δ vs. AoT		-40.01%	-41.95%	-39.62%	-39.95%	–	–	+32.20%	+30.93%	+30.55%	+32.30%	–	–
AoT	0.40	1.80	2.51	4.33	7.49	OOM	OOM	44.0	93.4	107.6	118.8	OOM	OOM
IPB (Ours)	0.40	**1.26**	**1.70**	**2.90**	**5.18**	**8.78**	**12.90**	**53.9**	**114.3**	**133.0**	**146.2**	**153.5**	**155.0**
Δ vs. AoT		-29.99%	-32.44%	-33.09%	-30.91%	–	–	+22.45%	+22.34%	+23.58%	+23.07%	–	–
AoT	0.47	1.92	2.94	4.93	8.86	15.48	22.91	41.5	82.1	93.8	102.0	106.0	107.2
IPB (Ours)	0.47	**1.42**	**2.03**	**3.35**	**6.02**	**10.47**	**15.08**	**49.6**	**99.5**	**113.9**	**124.4**	**129.6**	**130.5**
Δ vs. AoT		-26.15%	-30.84%	-32.02%	-32.07%	-32.40%	-34.20%	+19.36%	+21.18%	+21.46%	+22.01%	+22.22%	+21.72%
AoT	0.53	7.12	9.39	11.61	15.03	22.27	36.22	15.4	39.9	56.0	71.7	82.5	89.8
IPB (Ours)	0.53	**1.60**	**2.44**	**4.18**	**6.96**	**12.36**	**16.69**	**45.6**	**88.1**	**99.2**	**107.4**	**110.9**	**111.5**
Δ vs. AoT		-77.55%	-74.02%	-63.95%	-53.73%	-44.50%	-53.91%	+195.65%	+121.05%	+77.18%	+49.73%	+34.37%	+24.12%
AoT	0.67	2.36	4.51	7.31	12.58	21.89	27.66	35.1	60.1	66.4	70.1	71.9	71.8
IPB (Ours)	0.67	**1.95**	**3.44**	**5.53**	**9.43**	**16.33**	**20.96**	**39.3**	**70.0**	**77.0**	**81.8**	**84.4**	**84.3**
Δ vs. AoT		-17.37%	-23.78%	-24.34%	-25.09%	-25.41%	-24.23%	+12.21%	+16.49%	+16.02%	+16.79%	+17.42%	+17.45%
AoT	0.80	2.72	5.66	9.11	15.87	27.38	34.71	31.1	50.1	54.4	56.8	57.9	57.8
IPB (Ours)	0.80	**2.42**	**4.99**	**7.54**	**13.11**	**22.69**	**29.07**	**33.3**	**54.3**	**59.3**	**62.0**	**63.3**	**63.1**
Δ vs. AoT		-10.79%	-11.91%	-17.19%	-17.37%	-17.11%	-16.26%	+7.10%	+8.33%	+8.84%	+9.20%	+9.26%	+9.30%

A.6 Additional Ablations and Implementation Details

Hyperparameter sensitivity. Table 14 reports the frame-selection sensitivity analysis. We vary one frame-selection parameter at a time while keeping the remaining parameters fixed. Across the evaluated settings, the ablation accuracies differ from the main-table IPB results by at most 1.60 percentage points on Video-MME and 0.69 on LongVideoBench. Fixed quantiles are evaluated separately from the adaptive schedule.

Position encoding and token alignment. Table 15 summarizes the backbone-specific handling. For Qwen, mask alignment respects temporal tubelets and complete spatial merge blocks. Retained tokens preserve their original temporal and spatial position indices when gathered into shorter tensors. The LLaVA-OV path gathers tokens after spatial pooling and follows the model's one-dimensional sequence positions. Mask construction before inference is distinct from token removal: the LLaVA-OV gather described here is after spatial pooling, whereas the Qwen path uses pre-ViT pruning. Tubelet and merge-block alignment determines the realized token count from the requested patch masks.

Table 12: Full Table — Per-task codec-prior advantage across the three benchmarks.

Task	LLaVA-OV-7B (K=32)			LLaVA-Video-7B (K=64)			Qwen3-VL-4B (K=32)			Qwen3-VL-4B (K=64)		
	Uni.	Ours	Δ	Uni.	Ours	Δ	Uni.	Ours	Δ	Uni.	Ours	Δ
MVBench												
Moving Direction	20.5	**23.5**	+3.0	35.5	**36.0**	+0.5	57.5	**58.0**	+0.5	51.0	**52.5**	+1.5
Action Localization	**59.0**	58.0	-1.0	61.0	**66.0**	+5.0	55.0	**57.5**	+2.5	56.5	**57.5**	+1.0
Episodic Reasoning	52.5	**54.0**	+1.5	53.5	**55.0**	+1.5	50.5	**51.0**	+0.5	**53.0**	52.5	-0.5
Object Existence	58.0	**59.5**	+1.5	60.0	60.0	0.0	88.5	**89.0**	+0.5	85.5	**86.0**	+0.5
Action Sequence	71.0	**71.5**	+0.5	70.5	**77.5**	+7.0	77.5	77.5	0.0	79.5	79.5	0.0
Counterfactual Inference	**50.0**	48.5	-1.5	50.0	**59.0**	+9.0	70.5	**71.5**	+1.0	**72.0**	69.0	-3.0
Unexpected Action	78.0	**80.5**	+2.5	**81.5**	79.5	-2.0	**81.5**	81.0	-0.5	80.0	**82.0**	+2.0
Object Interaction	**83.0**	82.0	-1.0	**84.5**	84.0	-0.5	82.5	**83.5**	+1.0	82.5	**84.0**	+1.5
Moving Attribute	71.5	**73.5**	+2.0	71.0	**71.5**	+0.5	**93.5**	91.5	-2.0	92.0	92.0	0.0
Fine-grained Action	**48.0**	46.5	-1.5	48.0	**49.0**	+1.0	50.5	**51.0**	+0.5	**50.0**	49.5	-0.5
Action Prediction	**57.0**	55.0	-2.0	62.0	**64.0**	+2.0	**65.0**	60.5	-4.5	66.5	**67.5**	+1.0
Fine-grained Pose	**55.5**	55.0	-0.5	54.5	**60.0**	+5.5	**71.5**	71.0	-0.5	**71.0**	69.0	-2.0
State Change	**52.5**	50.5	-2.0	54.0	**57.5**	+3.5	**57.5**	56.5	-1.0	59.5	59.5	0.0
Action Antonym	**78.0**	76.5	-1.5	**76.0**	75.5	-0.5	75.0	**77.0**	+2.0	**78.0**	76.5	-1.5
Egocentric Navigation	**35.0**	34.0	-1.0	30.5	**31.0**	+0.5	**35.5**	34.5	-1.0	**33.0**	32.5	-0.5
Character Order	**72.0**	68.5	-3.5	74.5	**75.0**	+0.5	**82.0**	80.5	-1.5	82.0	82.0	0.0
Moving Count	**47.5**	45.0	-2.5	44.0	**45.0**	+1.0	69.0	69.0	0.0	**73.0**	69.0	-4.0
Object Shuffle	**35.5**	35.0	-0.5	41.5	**42.0**	+0.5	**40.0**	38.0	-2.0	**40.0**	35.5	-4.5
Action Count	46.5	**49.0**	+2.5	**57.0**	45.0	-12.0	**60.0**	59.0	-1.0	**60.0**	59.0	-1.0
Scene Transition	93.0	93.0	0.0	**93.5**	92.5	-1.0	92.5	92.5	0.0	93.0	93.0	0.0
Average	**58.2**	58.0	-0.2	60.15	**61.3**	+1.1	**67.8**	67.5	-0.3	**67.9**	67.4	-0.5
Video-MME												
OCR Problems	60.4	**61.9**	+1.4	65.5	**69.8**	+4.3	69.8	**71.2**	+1.4	72.7	**75.5**	+2.9
Action Reasoning	54.4	**56.1**	+1.8	53.7	**57.9**	+4.2	58.2	**58.9**	+0.7	58.2	**59.6**	+1.4
Temporal Reasoning	41.8	**42.9**	+1.1	48.0	**50.3**	+2.3	43.5	**45.2**	+1.7	44.6	**47.5**	+2.8
Object Recognition	65.5	**68.1**	+2.5	68.9	68.9	0.0	68.6	**68.9**	+0.3	72.0	**74.0**	+2.0
Counting Problem	36.9	**38.4**	+1.5	43.7	**46.6**	+3.0	**44.8**	42.2	-2.6	44.8	**45.5**	+0.7
Object Reasoning	57.0	**57.5**	+0.4	**61.7**	60.1	-1.5	58.4	**60.4**	+2.0	60.6	**61.2**	+0.7
Temporal Perception	**61.8**	60.0	-1.8	72.7	**78.2**	+5.5	**74.5**	72.7	-1.8	72.7	**78.2**	+5.5
Spatial Perception	57.4	**68.5**	+11.1	**66.7**	63.0	-3.7	70.4	**74.1**	+3.7	**75.9**	70.4	-5.6
Information Synopsis	73.4	**75.2**	+1.9	75.5	**76.5**	+0.9	**76.8**	76.5	-0.3	**78.9**	76.5	-2.5
Action Recognition	55.0	**55.3**	+0.3	**64.2**	60.4	-3.8	62.9	**63.6**	+0.6	**66.8**	64.9	-1.9
Attribute Perception	**73.0**	70.7	-2.3	**79.3**	77.5	-1.8	**77.9**	77.0	-0.9	78.4	**78.8**	+0.5
Spatial Reasoning	80.4	80.4	0.0	**87.5**	82.1	-5.4	75.0	**76.8**	+1.8	**75.0**	73.2	-1.8
Average	58.7	**59.8**	+1.1	63.6	**63.9**	+0.3	63.2	**63.6**	+0.4	65.1	**65.5**	+0.4
LongVideoBench (val_v)												
Event before/after Text	47.9	**52.1**	+4.1	53.4	**56.2**	+2.7	60.3	**61.6**	+1.4	60.3	**61.6**	+1.4
Text-referred Object Attribute Change	43.9	**53.7**	+9.8	**50.0**	48.8	-1.2	53.7	**57.3**	+3.7	56.1	**57.3**	+1.2
Event before/after Event	60.6	**66.0**	+5.3	73.4	73.4	0.0	62.8	**64.9**	+2.1	62.8	**67.0**	+4.3
Text-referred Object Existence	59.2	**61.8**	+2.6	59.2	**60.5**	+1.3	**57.9**	55.3	-2.6	63.2	**65.8**	+2.6
Scene-referred Event	69.9	**71.0**	+1.1	**73.1**	71.0	-2.2	66.7	**68.8**	+2.2	64.5	**65.6**	+1.1
Object before/after Text	**52.7**	45.9	-6.8	58.1	**60.8**	+2.7	54.1	**56.8**	+2.7	55.4	**56.8**	+1.4
Sequence of Scenes	27.8	**35.1**	+7.2	36.1	**38.1**	+2.1	**44.3**	40.2	-4.1	43.3	43.3	0.0
Text-referred Event	**60.0**	56.9	-3.1	60.0	**63.1**	+3.1	63.1	63.1	0.0	63.1	**66.2**	+3.1
Scene-referred Object Tracking	**64.2**	63.0	-1.2	69.1	**74.1**	+4.9	**67.9**	64.2	-3.7	63.0	**65.4**	+2.5
Scene-referred Object Existence	52.8	**55.6**	+2.8	63.9	**66.7**	+2.8	**62.5**	59.7	-2.8	**65.3**	63.9	-1.4
Scene-referred Object Attribute Change	48.6	48.6	0.0	**59.7**	56.9	-2.8	61.1	**63.9**	+2.8	62.5	**63.9**	+1.4
Object-referred Event	62.1	**66.7**	+4.6	**70.1**	66.7	-3.4	67.8	**69.0**	+1.1	**71.3**	69.0	-2.3
Scene-referred Object Attribute	**76.1**	73.9	-2.3	**77.3**	73.9	-3.4	**78.4**	77.3	-1.1	78.4	**81.8**	+3.4
Event-referred Object	**76.9**	72.3	-4.6	**70.8**	69.2	-1.5	**64.6**	61.5	-3.1	63.1	**64.6**	+1.5
Text-referred Object Attribute	**57.0**	53.2	-3.8	**58.2**	57.0	-1.3	**68.4**	63.3	-5.1	67.1	**68.4**	+1.3
Text-referred Object Tracking	**38.4**	34.2	-4.1	**39.7**	37.0	-2.7	**41.1**	39.7	-1.4	**37.0**	34.2	-2.7
Object before/after Object	**50.0**	47.0	-3.0	**57.6**	51.5	-6.1	**60.6**	57.6	-3.0	**66.7**	65.2	-1.5
Average	56.3	**56.6**	+0.3	**60.7**	60.4	-0.3	**61.0**	60.4	-0.6	61.3	**62.4**	+1.0

Table 13: Full Table — Ablation study of our IPB frame selection on LLaVA-OV-7B Video-MME (K=32). Per-task breakdown across all 12 Video-MME categories.

Method	Avg.	Action Reasoning	Action Recognition	Attribute Perception	Counting Problem	Information Synopsis	OCR Problems	Object Reasoning	Object Recognition	Spatial Perception	Spatial Reasoning	Temporal Perception	Temporal Reasoning
Uniform	58.7	54.4	55.0	**73.0**	36.9	73.4	60.4	57.0	65.5	57.4	80.4	**61.8**	41.8
IPB (Ours)	**59.8**	**56.1**	55.3	70.7	38.4	**75.2**	61.9	**57.5**	**68.1**	**68.5**	**80.4**	60.0	**42.9**
Coding score	57.9	54.4	**55.9**	68.0	35.1	73.4	59.7	55.7	65.5	59.3	80.4	58.2	41.2
Motion gate	54.6	51.6	53.0	65.3	33.6	67.8	60.4	49.6	60.7	50.0	75.0	47.3	38.4
I-frame	56.6	55.1	55.0	68.0	**39.9**	68.1	**62.6**	51.8	63.0	57.4	78.6	50.9	40.7
PktSize	56.6	53.7	53.4	68.0	38.8	67.5	**62.6**	52.6	64.7	61.1	80.4	50.9	41.8

Table 14: **One-at-a-time hyperparameter ablations.** Accuracy (%) with LLaVA-OV-7B, $K = 32$, and frame selection only. Each ablation row varies one parameter with the others fixed. The main-table IPB result is included for comparison.

Parameter	Value	Video-MME	LongVideoBench
IPB (Table 3)		59.8	56.6
β	1	58.20	56.69
	4	58.60	56.47
κ	0.25	58.80	57.29
	1	58.50	56.10
q	0.45 (fixed)	59.20	56.62
	0.85 (fixed)	59.10	56.25
Δ_{gap}/b	1	59.00	56.55
	5	59.10	57.29
$\Delta_{\min}/b$	0.5	59.10	57.07
	2	59.10	57.07
d_m	1.5	59.60	57.07
	3.5	58.70	56.10

Defaults are $\beta = 2$, $\kappa = 0.5$, $d_m = 2.5$, $\Delta_{\min} = b$, and $\Delta_{\text{gap}} = 3b$, where $b = \max(1, \lfloor D_{\text{sec}} \rfloor)$ frames. The two threshold rows report multipliers Δ_{gap}/b and $\Delta_{\min}/b$. The default q is adaptive (0.45/0.85/0.45 according to the coefficient of variation); fixed-q rows replace this schedule.

Table 15: **Position encoding and token alignment.** Implementation details accompanying the pruning results; no additional evaluation is introduced.

Component	Handling
Qwen temporal tubelets	Masks from each two-frame tubelet are OR-pooled onto the temporal token grid.
Qwen spatial merge	Selection operates on complete 2×2 spatial merge blocks, ranked by their retained-patch counts; partial merge blocks are not passed to the merger.
Original positions	Full-grid MRoPE positions are computed first. Retention gathers preserve the original temporal and spatial (t, h, w) indices rather than renumbering the retained positions.
Packed tokens	Retained tokens are gathered into shorter dense tensors in the merge-compatible order; cumulative sequence lengths are rebuilt for the retained tokens in each temporal patch.
LLaVA-OV path	Tokens are gathered after spatial pooling and use the model's one-dimensional sequence positions. This path is distinct from Qwen's pre-ViT/MRoPE handling.

A.7 Core Implementation

Algorithms 2 and 3 give NumPy-style core operations on prepared inputs, using the hyperparameters in table 7 and section A.2.1. In the listings, `packet` and `motion` denote $|b_i|$ and m_i; `density_mult` denotes d_m, `keep_ratio` denotes r_p, and `w_tex` denotes w_{tex}. The `cv_breaks` and `quantiles` arrays specify the CV-adaptive schedule in section A.2.1. Frame inputs are in display order, with block-area-weighted motion mass and $1 \leq K \leq T$.

The patch listing uses decoded grayscale images and codec motion records in decoder order. Vector assignment uses the last record at each grid cell; the score uses the past-reference channels. Channel normalization precedes patch summation, and partial boundary patches are cropped. I-frame motion is zero, leaving Sobel texture to determine the ranking. Returned masks do not perform model-specific token removal; temporal tubelets, spatial merging and position alignment follow table 15. The listings assume valid prepared inputs and omit decoding, storage, CLI, missing-input recovery and serving orchestration.

Algorithm 2 Code: IPB Frame Selection

```python
import numpy as np

def runs(mask):  # Sorted half-open True intervals [start, end).
    edges = np.diff(np.r_[False, mask, False].astype(int))
    return list(zip(np.flatnonzero(edges == 1), np.flatnonzero(edges == -1)))

def uniform(a, b, n):  # Midpoint samples; an empty budget returns [].
    return [min(b - 1, a + int((j + 0.5) * (b - a) / n)) for j in range(n)]

def frame_select(types, packet, motion, fps, K, *, beta, kappa,
                 density_mult, min_gop_sec, cv_breaks, quantiles):
    # Inputs: display order, finite nonnegative masses, fps > 0, 1 <= K <= T.
    T = len(types); packet = np.asarray(packet, dtype=np.float64)
    motion = np.asarray(motion, dtype=np.float64)
    starts = [0] + [i for i in range(1, T) if types[i] == "I"]
    gops = list(zip(starts, starts[1:] + [T]))
    bp = max(np.median(packet[packet > 0]) if np.any(packet > 0) else 1., 1e-9)
    mp = max(np.median(motion[motion > 0]) if np.any(motion > 0) else 1., 1e-9)
    scores = []
    for a, z in gops:
        R = float(packet[a:z].sum()) / (bp * (z - a))
        M = float(motion[a:z].mean()) / mp
        score = np.log1p(R) * (1 - np.exp(-beta * M))
        scores.append(score if z - a >= max(1, int(min_gop_sec * fps)) else 0.)
    u = np.zeros(T, dtype=np.float64)
    for score, (a, z) in zip(scores, gops):
        if score <= 0: continue
        center = (a + z) / 2; sigma = max(1., kappa * (z - a))
        t = np.arange(max(0, int(center - 3*sigma)), min(T, int(center + 3*sigma)))
        u[t] += score * np.exp(-0.5 * ((t - center) / sigma)**2)
    if u.max() == 0: return uniform(0, T, K)
    positive = np.asarray(scores)[np.asarray(scores) > 0]
    cv = 0. if len(positive) < 2 else positive.std() / max(positive.mean(), 1e-9)
    q = quantiles[0] if cv < cv_breaks[0] else (
        quantiles[1] if cv < cv_breaks[1] else quantiles[2])
    active = u >= np.quantile(u[u > 0], q)
    b = max(1, int(T / fps)); delta_min, delta_gap = b, 3*b
    segments = []
    for a, z in runs(active):
        mass = float(u[a:z].sum())
        if segments and a - segments[-1][1] < delta_gap:
            segments[-1][1] = z; segments[-1][2] += mass  # Exclude gap mass.
        else: segments.append([a, z, mass])
    segments = [(a, z, m) for a, z, m in segments if z-a >= delta_min and m > 0]
    if not segments: return uniform(0, T, K)
    L = sum(z-a for a, z, _ in segments); outside = T-L
    if outside <= 0 or L <= 0: return uniform(0, T, K)
    density = K / (L*density_mult + outside)
    inside = max(len(segments), min(round(L*density*density_mult), K-1))
    mass_sum = sum(m for _, _, m in segments); selected = []
    for a, z, m in segments:
        n = min(z-a, max(1, round(inside*m/mass_sum)))
        selected.extend(uniform(a, z, n))
    remaining = K-len(selected)  # Independent rounding may leave no outside slot.
    gaps = []; previous = 0
    for a, z, _ in segments:
        if a > previous: gaps.append((previous, a))
        previous = z
    if previous < T: gaps.append((previous, T))
    if remaining > 0 and gaps:
        length = sum(z-a for a, z in gaps)
        counts = [max(0, round(remaining*(z-a)/length)) for a, z in gaps]
        while sum(counts) > remaining: counts[int(np.argmax(counts))] -= 1
        while sum(counts) < remaining: counts[int(np.argmin(counts))] += 1
        for (a, z), n in zip(gaps, counts): selected.extend(uniform(a, z, n))
    selected = sorted(set(selected))
    if len(selected) < K:
        used = set(selected)
        for i in range(T):
            if len(selected) >= K: break
            if i not in used: selected.append(i); used.add(i)
        selected.sort()
    return selected[:K]  # Match the reference's final truncation order.
```

Algorithm 3 Code: IPB Patch Selection

```python
import numpy as np

def normalize(x, percentile):
    finite = x[np.isfinite(x)]
    scale = max(float(np.percentile(finite, percentile)), 1e-6) if finite.size else 1.
    return np.clip(x / scale, 0., 1.).astype(np.float32)

def patch_sum(x, p):
    h, w = x.shape[0] // p, x.shape[1] // p
    return x[:h*p, :w*p].reshape(h, p, w, p).sum(axis=(1, 3)).astype(np.float32)

def sobel(gray):  # 3x3 Sobel, REFLECT_101 boundary, no gradient rescaling.
    y = np.pad(np.asarray(gray, dtype=np.float32), 1, mode="reflect")
    gx = (y[:-2, 2:] - y[:-2, :-2] + 2*(y[1:-1, 2:] - y[1:-1, :-2])
          + y[2:, 2:] - y[2:, :-2])
    gy = (y[2:, :-2] - y[:-2, :-2] + 2*(y[2:, 1:-1] - y[:-2, 1:-1])
          + y[2:, 2:] - y[:-2, 2:])
    return np.sqrt(gx*gx + gy*gy).astype(np.float32)

def motion_grid(records, H, W, mv_block):
    # Structured MV records in decoder order; L0 channels only enter the score.
    h, w = max(1, H // mv_block), max(1, W // mv_block)
    grid = np.zeros((h, w, 2), dtype=np.int32)
    for r in records:
        if r["source"] > 0: continue  # L1 is stored separately, unused by scoring.
        x = np.clip(int(r["dst_x"]) // mv_block, 0, w-1)
        y = np.clip(int(r["dst_y"]) // mv_block, 0, h-1)
        grid[y, x] = (r["motion_x"], r["motion_y"])  # Last write wins.
    return grid

def patch_select(gray, types, mv_records, patch_size, keep_ratio, *,
                 mv_block, mv_unit_div, percentile, w_mv, w_tex):
    # Valid decoded inputs; images >= one patch; 0 < keep_ratio <= 1.
    masks = []
    for image, kind, records in zip(gray, types, mv_records):
        H, W = image.shape; p = patch_size
        texture = patch_sum(normalize(sobel(image), percentile), p)
        motion = np.zeros_like(texture)
        if kind != "I":
            grid = motion_grid(records, H, W, mv_block)
            vx = grid[..., 0].astype(np.float32) / mv_unit_div
            vy = grid[..., 1].astype(np.float32) / mv_unit_div
            vx -= np.median(vx); vy -= np.median(vy)
            energy = normalize(np.sqrt(vx*vx + vy*vy), percentile)
            # Nearest-neighbor expansion from the codec grid to source pixels.
            rows = np.floor(np.arange(H) * (energy.shape[0] / H)).astype(int)
            cols = np.floor(np.arange(W) * (energy.shape[1] / W)).astype(int)
            motion = patch_sum(energy[rows[:, None], cols[None, :]], p)
        # I-frames have zero motion: texture alone determines their ranking.
        score = ((w_mv*motion + w_tex*texture) / max(w_mv+w_tex, 1e-6))
        flat = score.astype(np.float32).ravel(); n = len(flat)
        k = min(n, int(np.ceil(n*keep_ratio)))  # Eq. (15); upper bound n = Hp*Wp.
        mask = np.zeros(n, dtype=bool)
        mask[np.argpartition(-flat, k-1)[:k]] = True
        masks.append(mask.reshape(texture.shape))
    return np.stack(masks)  # Mask construction only; token gathering is separate.
```