A. A4VL Algorithm

The detailed algorithm is shown in Algorithm 1. First, the video is partitioned to at most B blocks. Suppose the event-based partitioning algorithm finally divides the video into b ($\leq B$) blocks. In case it divides the video into more than B blocks, we simply merge similar blocks until there are B blocks in total. Line 4-9 implements sample-clue based exploration, where the perception clues are generated. Line 10-19 performs block-based perception sampling. Then action exploration step starts from line 20. From line 20-26, each agent generates the answer and the reason, and a consensus check is made to decide whether we early exit. Line 27-34 performs the agent pruning with multi-round deliberation. The worst agent is pruned and remaining agents revise their perception clue for the next round.

B. Additional Visualizations

Figure 6 shows the result of event-based partitioning on two cases. Segments are annotated by colored blocks below the video frames. The top case is a video from NeXT-QA, which lasts for 25 seconds. Our event-based partitioning algorithm accurately divides the video into two blocks: in the first block, the girl keeps talking, and in the second block, the girl takes out a paper. The bottom example is from EgoSchema, which lasts for three minutes. Our partitioning algorithm divides the video into six blocks, where the person alternates between reading the book, using the phone, taking out a pen, and looking around. The entire procedure is training free and query agnostic. Figure 7

Figure 6. Visualization of event-based video partitioning.

presents a case in which the majority answer (B) in the first round is incorrect, yet A4VL ultimately returns the correct answer in the final round, supported by a sound explanation from our summarizer. In this example, the green agent corresponds to InternVL3.5-38B [41], the orange agent to InternVL3-78B [41], and the pink agent to QwenVL-2.5-

Algorithm 1: A4VL Algorithm.

Input: Video $V = \{v_1, \ldots, v_n\}$, question Q, options O, agent pool $\mathbb{A} = \{A_i : 1 \leq i \leq m\}$, preview frames N_1, inference frames N_2, max number of blocks B.

Output: Answer $\mathcal{A}$.

1 $\{B_1, \ldots, B_b\} \leftarrow \text{EventPartition}(V, B)$
2 $j \leftarrow 1$
3 **while** $|\mathbb{A}| \geq 1$ **do**
4 **foreach** $A_i \in \mathbb{A}$ **do** Step 1.1: Sample-Clue Based Exploration
5 **if** $j = 1$ **then**
6 $\hat{v}_{A_i} \leftarrow p_1(V, N_1)$
7 $P_{i,1} \leftarrow A_{i,\text{clue}}(\hat{v}_{A_i}, Q, O)$
8 **else**
9 $P_{i,j}$ given
10 **foreach** $A_i \in \mathbb{A}$ **do** Step 1.2: Block-Based Perception Sampling
11 $\mathbf{s}^{(i)} \leftarrow \{Sim(B_k, P_{i,j}) | 1 \leq k \leq b\}$
12 **if** $\max(\mathbf{s}^{(i)}) > \rho$ **then**
13 $\mathbf{s}^{(i)}[\mathbf{s}^{(i)} \leq \rho] = -\infty$
14 $\mathbf{s}^{(i)} \leftarrow \mathbf{s}^{(i)} - \max(\mathbf{s}^{(i)})$
15 $\mathbf{c}^{(i)} \leftarrow N_2 \lfloor SoftMax(\mathbf{s}^{(i)}) \rfloor$
16 $V_{\text{act}}^{(i)} \leftarrow \bigcup_{k=1}^{b} p_2(B_k, c_k^{(i)})$
17 **else**
18 $k^* \leftarrow \arg\max_{k \in \{1,\ldots,b\}} s_k^{(i)}$
19 $V_{\text{act}}^{(i)} \leftarrow p_2(B_{k^*}, N_2)$
20 **foreach** $A_i \in \mathbb{A}$ **do** Step 2.1: Generating Answer & Reason
21 $a_{i,j} \leftarrow A_{i,\text{act}}(V_{\text{act}}^{(i)}, Q, O)$
22 $R_{i,j} \leftarrow A_{i,\text{reason}}(V_{\text{act}}^{(i)}, a_{i,j}, Q)$
23 $S_{a,j} \leftarrow \{a_{i,j} \mid A_i \in \mathbb{A}\}$
24 $S_{r,j} \leftarrow \{R_{i,j} \mid A_i \in \mathbb{A}\}$
25 **if** $\forall i, i' : a_{i,j} = a_{i',j}$ **then**
26 **return** $\mathcal{A} \leftarrow a_{1,j}$
27 **foreach** $A_i \in \mathbb{A}$ **do** Step 2.2: Pruning with Deliberation
28 $(s_{i,\mathbb{A}_1}, \ldots, s_{i,\mathbb{A}_{|\mathbb{A}|}}) \leftarrow A_{i,\text{eval}}(Q, S_{a,j}, S_{r,j})$
29 **foreach** $A_i \in \mathbb{A}$ **do**
30 $s_{A_i} \leftarrow \sum_{A_k \in \mathbb{A}} s_{k,\mathbb{A}_i}$
31 $A_{\min} \leftarrow \arg\min_{A \in \mathbb{A}} s_A$
32 $\mathbb{A} \leftarrow \mathbb{A} \setminus \{A_{\min}\}$
33 **foreach** $A_i \in \mathbb{A}$ **do**
34 $P_{i,j+1} \leftarrow A_{i,\text{refine}}(P_{i,j}, S_{a,j}, S_{r,j}, A_{\min}, Q, O)$
35 $j \leftarrow j + 1$
36 **return** $\mathcal{A} \leftarrow a_{i^\star,j-1}$ // $A_{i^\star}$ is the last remaining agent

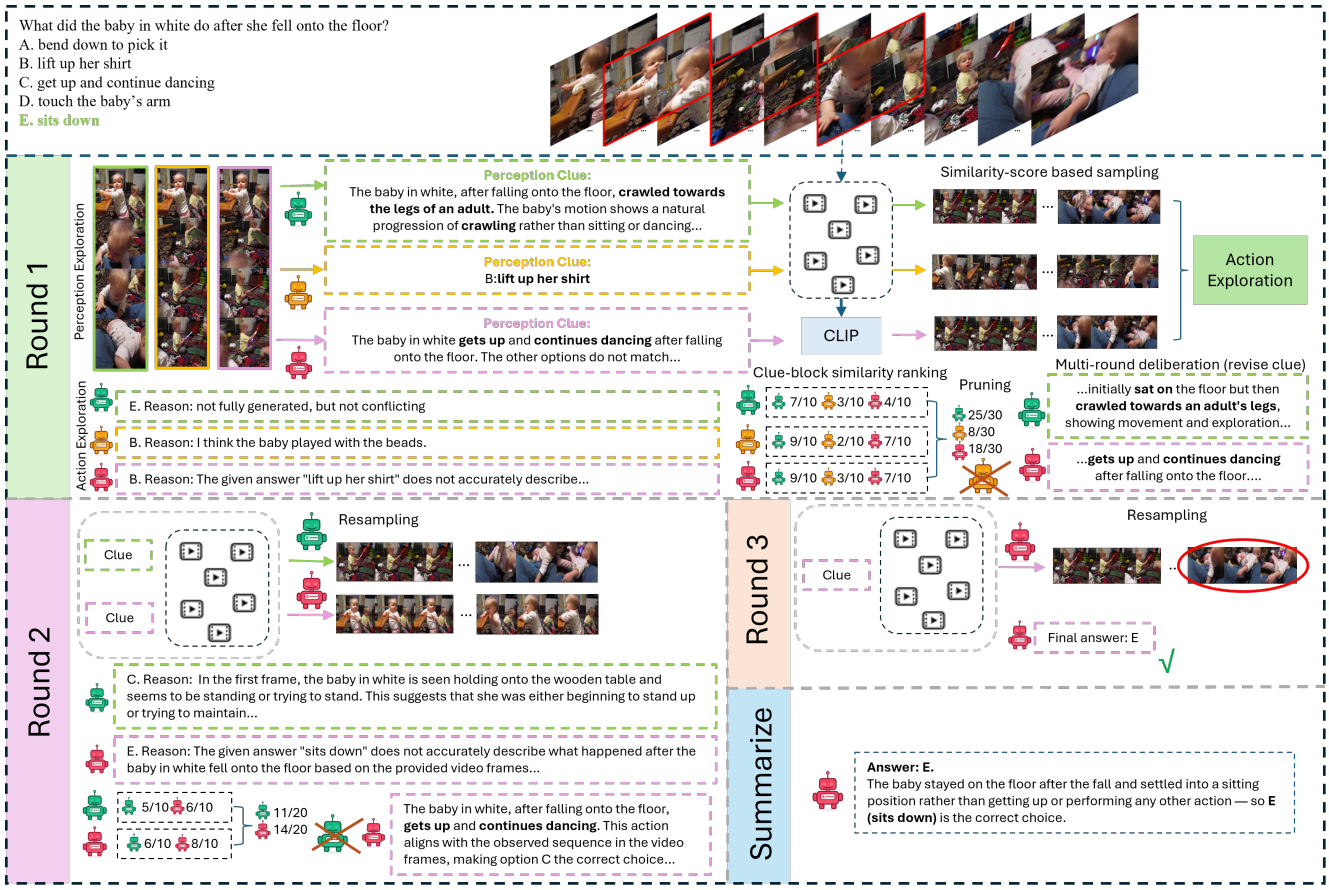

Figure 7. Example where the majority is wrong in the first round, but finally A4VL produces the correct answer. The example is taken from NeXT-QA.

72B [38]. We observe that, in the first round, only the green agent produces the correct answer, while the other two receive significantly lower scores. Notably, the green agent is not the model with the largest number of parameters. However, it exhibits instability and becomes incorrect in the second round, resulting in its removal. In contrast, the pink agent becomes correct in the second round and remains correct through the third round, leading to the correct final answer. These visualizations complement Figure 2 to 4 in the main paper by illustrating how event-based partitioning and multi-round collaboration behave on real videos.

C. Failure Cases

Although A4VL demonstrates superior performance over existing models, it can still fail in certain cases. For example, when all agents make an incorrect prediction in the first round, the early-exit mechanism forces the final answer to be wrong. Figure 8 shows such a failure case: the final answer is incorrectly chosen as option D because all agents in the first round output D. This error arises from several factors: (i) it is difficult for the perception frames to accurately capture the moments after the baby smiles; (ii) although the green agent generates the perception clue "lay still," CLIP is an image–text similarity model and thus struggles to capture temporal movements such as "still," leading to the selection of an incorrect block, while the other two agents, whose perception clues are themselves incorrect, also choose wrong blocks; and (iii) the question references the video position "at the end," but CLIP cannot measure the temporal alignment between frames and text. Addressing such cases is a challenging but important direction for future work, and potential remedies include integrating more advanced video grounding models and routing complex cases to these models when detected, as well as incorporating neuro-symbolic techniques for precise video processing, such as explicitly cropping the end segment of the video.

It is also possible that the majority vote in the first round is correct, while the final answer is wrong. As shown in Figure 9, we illustrate such a case from Video-MME (without subtitle). The green, orange, and pink agents represent InternVL3-78B [41], QwenVL-2.5-72B [38], and LLaVA-Video-72B-Qwen2 [54], respectively. This is a challenging

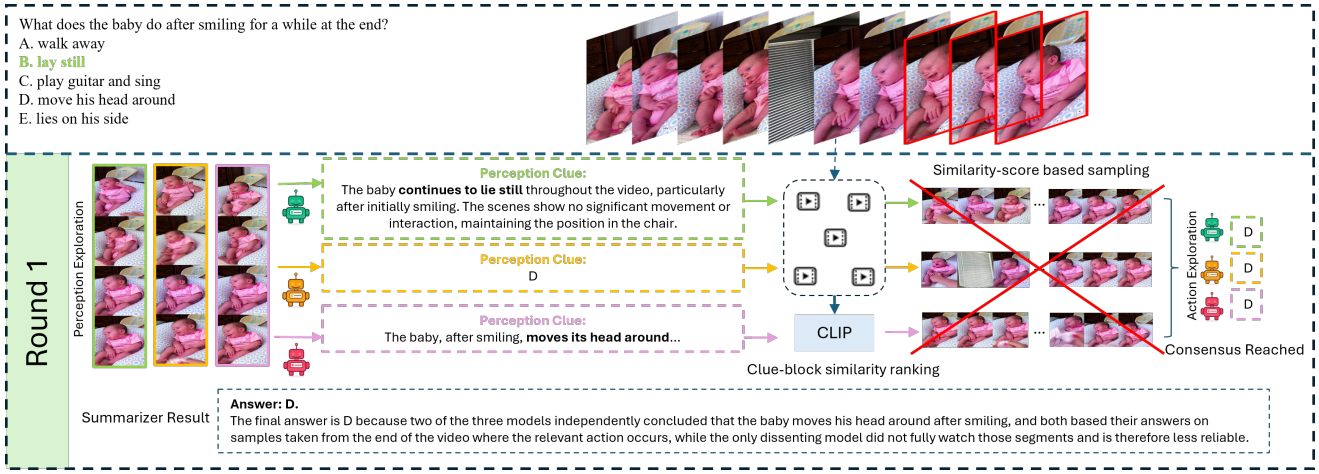

Figure 8. Failure case where all agents achieve a wrong consensus in the first round. The example is taken from NeXT-QA.

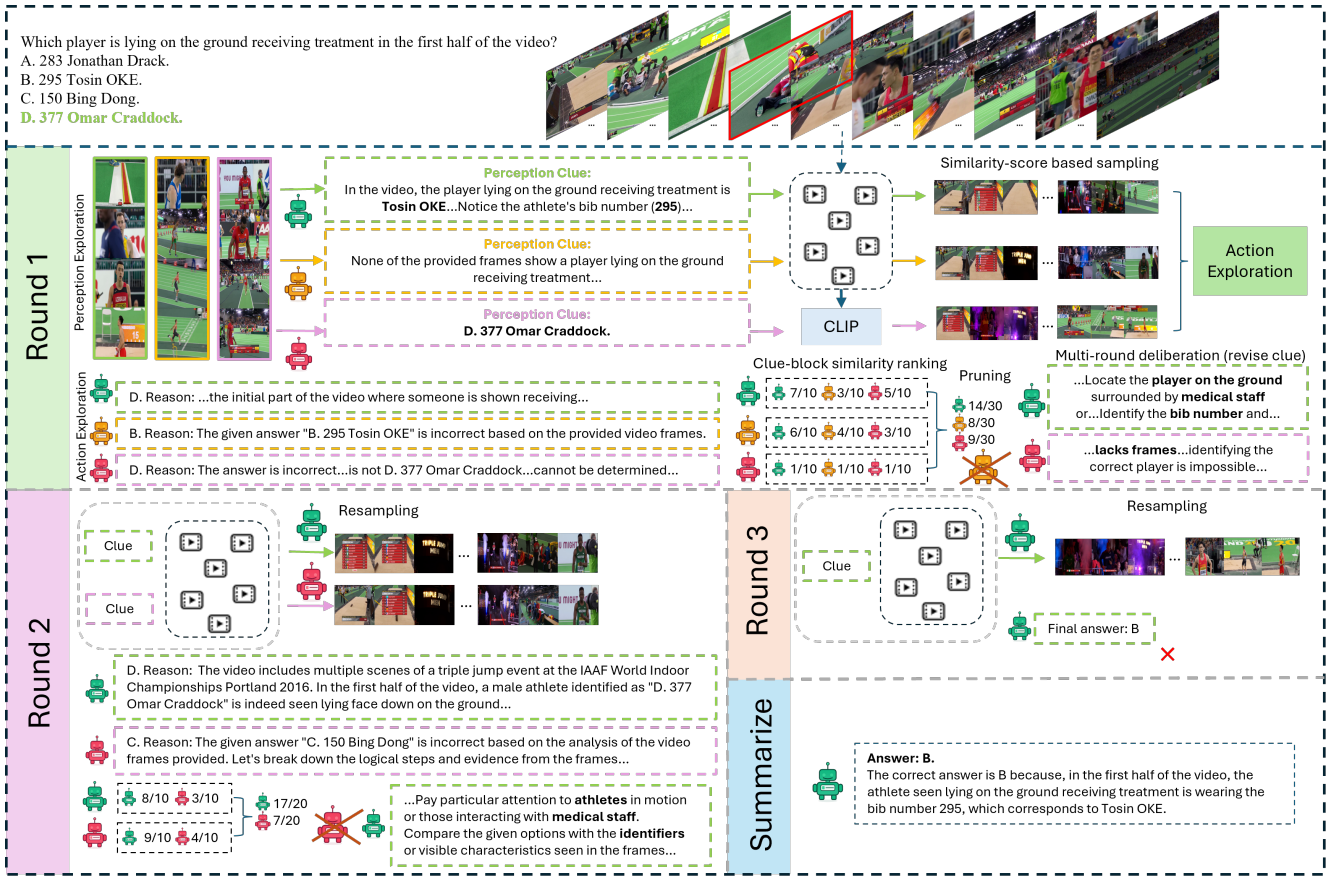

Figure 9. Failure case where the majority is correct in the first round, but finally the final answer gets wrong. This example is taken from Video-MME (w/o sub).

example because the entire video is about 20 minutes long, while the keyframes last only a few seconds. The perception frames provide no useful clues, and none of the sampled action frames contain the keyframe where a player is lying on the ground receiving treatment. As a result, each agent's behavior is highly unstable when reasoning over these irrelevant frames. In the first round, two agents choose answer D, likely because that player happens to appear most fre-

quently and is often centered in the view. However, their predictions shift in rounds 2 and 3, eventually converging on the wrong answer B. Another difficulty is that the question explicitly restricts the focus to the first half of the video. If this constraint were reliably detected and only the first half were used in the multi-round reasoning process (e.g., via the neural-symbolic approach mentioned earlier), the outcome might improve, though sampling the true keyframes would remain non-trivial. For such questions, an efficient framework based on fine-grained watching is therefore needed.

D. Event-Based Partitioning

Given a video V, our goal is to obtain a small set of temporal boundaries that partition V into at most B visually coherent blocks $\{B_1, \ldots, B_b\}$, where $b \leq B$. We follow a training-free, query-agnostic pipeline: (i) sampled decoding and feature extraction, (ii) construction of a fused novelty signal on time, (iii) multi-scale pooling and PELT-based segmentation on the 1D signal, and (iv) embedding-based heads (SSM and KTS) whose outputs are combined with non-maximum suppression (NMS) and a cap of at most $B-1$ cuts.

Sampled frames and notation. Let V have T frames $\{I_1, \ldots, I_T\}$ with timestamps $0 \leq t_1 < \cdots < t_T = L$. We uniformly sample at most F frames (in practice $F = 200$) and denote their indices by

$$\mathcal{S} = \{i_1 < i_2 < \cdots < i_N\}, \quad N \leq F,$$

with corresponding images I_{i_r} and timestamps t_{i_r}. For readability, we abbreviate $I_r := I_{i_r}$ and $t_r := t_{i_r}$ when referring to the sampled frames.

Color, sharpness, motion, and embeddings. For each sampled frame I_r, we compute four types of features.

HSV color histograms. we detect color changes across frames using histogram distances in HSV space. Specifically, we convert I_r from BGR to HSV, obtaining $H_r^{\text{img}}(x)$, $S_r^{\text{img}}(x)$, and $V_r^{\text{img}}(x)$ for pixel locations $x \in \Omega$ (image grid). We discretize each channel into K bins (we use $K = 32$) and compute histograms

$$h_r^H \in \mathbb{R}^K, \quad h_r^S \in \mathbb{R}^K, \quad h_r^V \in \mathbb{R}^K,$$

followed by a joint ℓ_1 normalization:

$$\tilde{h}_r^H = \frac{h_r^H}{S_r}, \quad \tilde{h}_r^S = \frac{h_r^S}{S_r}, \quad \tilde{h}_r^V = \frac{h_r^V}{S_r},$$

with

$$S_r = \sum_{k=1}^{K} h_r^H[k] + \sum_{k=1}^{K} h_r^S[k] + \sum_{k=1}^{K} h_r^V[k] + \varepsilon,$$

where $\varepsilon > 0$ is a small constant for numerical stability. The concatenated color descriptor is

$$c_r = \left[\tilde{h}_r^H; \tilde{h}_r^S; \tilde{h}_r^V\right] \in \mathbb{R}^{3K}.$$

Sharpness. we detect focus changes using the variance of a Laplacian filter. We convert I_r to gray-scale $G_r(x)$ and apply a discrete Laplacian filter $\Delta G_r(x)$ (e.g., the standard 3×3 kernel). The sharpness score is the variance of the Laplacian:

$$\mu_r^{\Delta} = \frac{1}{|\Omega|} \sum_{x \in \Omega} \Delta G_r(x), \qquad s_r = \frac{1}{|\Omega|} \sum_{x \in \Omega} \left(\Delta G_r(x) - \mu_r^{\Delta}\right)^2.$$

Motion. we measure frame-to-frame intensity differences as a proxy for motion magnitude. For $r \geq 2$, we define a robust frame-to-frame motion magnitude using the median absolute difference between consecutive gray-scale images:

$$d_r^{\text{mot}} = \text{median}_{x \in \Omega} \left|G_r(x) - G_{r-1}(x)\right|, \qquad m_r = \frac{1}{255}\, d_r^{\text{mot}},$$

and set $m_1 = 0$.

DINOv2 embeddings. we use cosine distance in feature space to capture semantic changes: we pass a resized RGB version of I_r through a DINOv2 encoder [27] and obtain a feature vector

$$e_r \in \mathbb{R}^D.$$

We use ℓ_2-normalized embeddings $\hat{e}_r = e_r / (\|e_r\|_2 + \varepsilon)$ when computing cosine similarities.

Per-step novelty from consecutive frames. We now define novelty cues on the temporal midpoints

$$\tau_r = \frac{t_{r-1} + t_r}{2}, \qquad r = 2, \ldots, N.$$

Color novelty. For each channel we use cosine distance between histograms; for hue, for example:

$$d_r^H = 1 - \cos\left(\tilde{h}_r^H, \tilde{h}_{r-1}^H\right) = 1 - \frac{\langle \tilde{h}_r^H, \tilde{h}_{r-1}^H \rangle}{\|\tilde{h}_r^H\|_2 \, \|\tilde{h}_{r-1}^H\|_2 + \varepsilon},$$

and analogously d_r^S and d_r^V. We combine them as

$$c_r^{\text{nov}} = \alpha\, d_r^H + \beta\, d_r^S + \gamma\, d_r^V.$$

where $\alpha = 0.55, \beta = 0.35$ and $\gamma = 0.10$ in our settings.

Sharpness and motion novelty. We use absolute temporal differences:

$$u_r^{\text{shp}} = \left|s_r - s_{r-1}\right|, \qquad u_r^{\text{mot}} = \left|m_r - m_{r-1}\right|.$$

Embedding novelty with EMA. We first compute a forward exponential moving average (EMA) of embeddings:

$$\tilde{e}_1 = e_1, \qquad \tilde{e}_r = \delta \tilde{e}_{r-1} + (1 - \delta) e_r,$$

with $\delta = 0.9$. For each $r \geq 2$ we define two cosine distances:

$$d_r^{\text{prev}} = 1 - \cos(\hat{e}_r, \hat{e}_{r-1}), \qquad d_r^{\text{ema}} = 1 - \cos(\hat{e}_r, \widehat{\bar{e}}_{r-1}),$$

where $\widehat{\bar{e}}_{r-1}$ is the ℓ_2-normalized EMA. The embedding novelty is

$$u_r^{\text{emb}} = \tfrac{1}{2} d_r^{\text{prev}} + \tfrac{1}{2} d_r^{\text{ema}}.$$

Robust normalization and fusion. For any scalar sequence x_r ($r = 2, \ldots, N$) we denote its *robust z-score* by

$$\text{med}(x) = \text{median}_r\, x_r,$$
$$\text{MAD}(x) = \text{median}_r\, \big|x_r - \text{med}(x)\big|,$$
$$z_r(x) = \frac{x_r - \text{med}(x)}{1.4826\ \text{MAD}(x) + \varepsilon}.$$

The constant $1.4826 = 1/\Phi^{-1}(0.75)$, where Φ is the standard normal CDF, so that $1.4826\,\text{MAD}$ is a consistent estimator of the standard deviation for Gaussian data.

We additionally apply a short moving average of window w (we use $w = 5$) to reduce noise:

$$\bar{z}_r(x) = \frac{1}{|\mathcal{W}_r|} \sum_{u \in \mathcal{W}_r} z_u(x),$$

where $\mathcal{W}_r = \{u : |u - r| \leq \lfloor w/2 \rfloor\}$ truncated at the sequence boundaries. We finally clip values to $[-4, 4]$.

We apply this procedure separately to the color, motion, sharpness, and embedding novelties:

$$z_r^{\text{col}} = \bar{z}_r\big(c^{\text{nov}}\big), \quad z_r^{\text{mot}} = \bar{z}_r\big(u^{\text{mot}}\big),$$
$$z_r^{\text{shp}} = \bar{z}_r\big(u^{\text{shp}}\big), \quad z_r^{\text{emb}} = \bar{z}_r\big(u^{\text{emb}}\big).$$

We then compute cue-specific empirical standard deviations

$$\sigma_{\text{col}} = \text{std}_r\left(z_r^{\text{col}}\right), \quad \sigma_{\text{mot}} = \text{std}_r\left(z_r^{\text{mot}}\right)$$
$$\sigma_{\text{emb}} = \text{std}_r\left(z_r^{\text{emb}}\right), \quad \sigma_{\text{shp}} = \text{std}_r\left(z_r^{\text{shp}}\right)$$

and combine them with fixed base weights

$$\beta_{\text{col}} = 0.20, \quad \beta_{\text{mot}} = 0.30, \quad \beta_{\text{emb}} = 0.35, \quad \beta_{\text{shp}} = 0.05.$$

The final fusion weights are

$$w_c = \frac{\beta_c \sigma_c}{\sum_{c' \in \{\text{col,mot,emb,shp}\}} \beta_{c'} \sigma_{c'}}.$$

The fused novelty signal on midpoints τ_r is

$$s(\tau_r) = w_{\text{col}}\, z_r^{\text{col}} + w_{\text{mot}}\, z_r^{\text{mot}} + w_{\text{emb}}\, z_r^{\text{emb}} + w_{\text{shp}}\, z_r^{\text{shp}}.$$

We apply another short moving average in time (again with window 5) to obtain a smooth 1D signal, which we still denote by $s(\tau_r)$.

Multi-scale pooling and PELT heads. To handle varying frame rates and video lengths, we pool $s(\tau_r)$ onto several temporal grids. For a grid resolution $\Delta > 0$ (we use $\Delta \in \{0.25, 0.5, 1.0\}$ seconds), we define bin edges

$$u_0 = \tau_2, \quad u_J = \tau_N, \quad u_j = u_0 + j\Delta,$$

and bins $[u_j, u_{j+1})$ for $j = 0, \ldots, J-1$. The pooled signal is

$$s_j^{(\Delta)} = \max\big\{s(\tau_r)\ :\ \tau_r \in [u_j, u_{j+1})\big\},$$

with bin centers $t_j^{(\Delta)} = (u_j + u_{j+1})/2$.

On each pooled sequence $\{s_j^{(\Delta)}\}_{j=0}^{J-1}$ we apply PELT [12] with a standard ℓ_2 cost and a penalty λ swept over a small range. Denoting by $\mathcal{K}^{(\Delta)}$ the predicted change-point indices for a given λ, the breakpoints correspond to bin centers $\{t_k^{(\Delta)} : k \in \mathcal{K}^{(\Delta)}\}$. We only keep candidates that yield segments of length at least

$$\ell_{\min} = \max\left(\ell_0,\ \frac{L}{15}\right),$$

where ℓ_0 is a user-defined minimum (e.g., 2 s). For each candidate index k we estimate a simple boundary strength via the difference of local averages:

$$\gamma_k^{(\Delta)} = \left| \frac{1}{W} \sum_{j=k}^{k+W-1} s_j^{(\Delta)} - \frac{1}{W} \sum_{j=k-W}^{k-1} s_j^{(\Delta)} \right|,$$

with a small window W (e.g., $W \approx \ell_{\min}/\Delta$). We discard boundaries with $\gamma_k^{(\Delta)}$ below a quantile threshold and collect the remaining breakpoints from all grid levels into a set $\mathcal{T}_{\text{PELT}}$.

Embedding-based SSM and KTS heads. We additionally operate directly on the embedding sequence. We first resample embeddings $\{e_r\}$ to a grid $\{\tilde{t}_q\}$ at roughly 1 Hz and obtain averaged embeddings $\tilde{e}_q$ in each bin.

SSM-based novelty. We L_2-normalize $\tilde{e}_q$ to $\hat{e}_q$ and build the Gram matrix

$$S_{pq} = \langle \hat{e}_p, \hat{e}_q \rangle.$$

We then follow Foote's self-similarity matrix (SSM) novelty detector [6]. For each center index c we extract a $(2w) \times (2w)$ block around (c, c), partition it into four $w \times w$ quadrants, and apply a Gaussian-weighted checkerboard kernel K with $+1$ on the top-left and bottom-right quadrants and -1 on the others:

$$n_c = \sum_{i,j} K_{ij}\, S_{(c+i-w),(c+j-w)}.$$

We robustly normalize the sequence $\{n_c\}$ via the same MAD-based z-score and keep local maxima separated by at

least $\ell_{\min}$ and above a quantile threshold. The corresponding times form $\mathcal{T}_{\text{SSM}}$. We set $\ell_{\min} = \max\{l, 4\}$, where l is the video length.

KTS-style segmentation. We also run a KTS-like greedy procedure [28]. Let E be the matrix whose rows are $\hat{e}_q$. We form the Gram matrix $G = EE^\top$ and its integral image to enable $O(1)$ evaluation of segment similarity. For a segment $[a, b]$, we define the cost

$$C(a,b) = -\frac{1}{(b-a+1)^2} \sum_{p=a}^{b} \sum_{q=a}^{b} G_{pq}.$$

Starting from the full interval $[0, Q-1]$, we recursively search for a split index k that maximizes the gain

$$\text{gain}(a,b,k) = C(a,b) - \big(C(a,k) + C(k+1,b)\big) - \lambda_{\text{KTS}},$$

with a data-dependent penalty λ_{KTS} (set to 1.4 in our experiment, under all benchmarks). If the best gain is positive and both resulting segments respect $\ell_{\min}$, we split and recurse. The resulting cut times form $\mathcal{T}_{\text{KTS}}$.

Merging, NMS, and truncation. We merge all candidate boundaries

$$\mathcal{T}_{\text{all}} = \mathcal{T}_{\text{PELT}} \cup \mathcal{T}_{\text{SSM}} \cup \mathcal{T}_{\text{KTS}},$$

sort them increasingly, and apply a one-dimensional non-maximum suppression with minimum separation $\ell_{\min}$. Concretely, we scan $\mathcal{T}_{\text{all}}$ in order and keep a candidate τ if $\tau - \tau_{\text{last}} \geq \ell_{\min}$, where τ_{last} is the time of the last kept boundary; otherwise we discard it. This yields a filtered set $\tilde{\mathcal{T}}$ and ensures that every resulting segment has duration at least $\ell_{\min}$.

If $|\tilde{\mathcal{T}}| > B-1$, we compute a boundary strength measure (e.g., $\gamma_k^{(\Delta)}$ on a fixed grid) for each $\tau \in \tilde{\mathcal{T}}$, rank boundaries by strength, and keep only the top $B-1$. Adding the implicit boundaries at 0 and L, we obtain

$$0 = \tau_0 < \tau_1 < \cdots < \tau_{b-1} < \tau_b = L,$$

which define the final blocks $B_k = V[\tau_{k-1}, \tau_k)$ used by A4VL.

The overall algorithm is shown in Algorithm 2.

E. Related Works

Video understanding. After the development of image QA, researchers began to explore video QA as a more complex multimodal understanding task. Pioneering works [11, 14, 46, 56, 57, 59] mainly adopt simple CNN–RNN style pipelines that encode pre-extracted frame features with LSTMs/GRUs over short clips, offering only limited modeling of long-range temporal structure and multimodal context. Modern methods begin to use LLM backbones for

Algorithm 2: Event-based video partitioning (EVENTPARTITION).

Input: Video V, max blocks B, max feature frames F, min segment length $\ell_{\min}$.
Output: Blocks $\{B_1, \ldots, B_b\}$ with $b \leq B$.

`// Step 1: sampled decode & feature extraction`

Uniformly sample at most F frames from V and record $\{I_r, t_r\}_{r=1}^{N}$;
for $r = 1$ **to** N **do**
  Compute $c_r, G_r, s_r, m_r, e_r, \hat{e}_r$ as in Appendix. D;

`// Step 2: per-step cues and fused novelty`

for $r = 2$ **to** N **do**
  Compute color novelty c_r^{nov}, $u_r^{\text{shp}} = |s_r - s_{r-1}|$, $u_r^{\text{mot}} = |m_r - m_{r-1}|$, and u_r^{emb} from embeddings (previous & EMA);
  Set midpoint $\tau_r = (t_{r-1} + t_r)/2$;
Apply MAD-based robust z-scoring and short moving average to obtain $z_r^{\text{col}}, z_r^{\text{mot}}, z_r^{\text{shp}}, z_r^{\text{emb}}$;
Compute fusion weights w_c and fused novelty $s(\tau_r) = \sum_c w_c z_r^{(c)}$, then smooth in time;

`// Step 3: PELT heads on pooled grids`

$\mathcal{T}_{\text{PELT}} \leftarrow \emptyset$;
foreach $\Delta \in \{0.25, 0.5, 1.0\}$ **do**
  Pool $s(\tau_r)$ into bins of width Δ (max pooling) to get $s_j^{(\Delta)}$;
  Run PELT on $\{s_j^{(\Delta)}\}_j$ with segment length $\geq \ell_{\min}$ and keep strong boundaries (via local strength $\gamma_k^{(\Delta)}$);
  Add resulting times to $\mathcal{T}_{\text{PELT}}$;

`// Step 4: embedding-based SSM and KTS heads`

Resample $\{e_r\}$ to a ~ 1 Hz grid $\{\tilde{e}_q, \tilde{t}_q\}$;
Compute SSM-based novelty on the Gram matrix of $\{\tilde{e}_q\}$ and keep strong peaks (gap $\geq \ell_{\min}$) to form $\mathcal{T}_{\text{SSM}}$;
Run KTS-style greedy segmentation on $\{\tilde{e}_q\}$ with penalty λ_{KTS} to obtain $\mathcal{T}_{\text{KTS}}$;

`// Step 5: merge, NMS, and truncation`

$\mathcal{T}_{\text{all}} \leftarrow \mathcal{T}_{\text{PELT}} \cup \mathcal{T}_{\text{SSM}} \cup \mathcal{T}_{\text{KTS}}$;
Sort $\mathcal{T}_{\text{all}}$ and apply 1D NMS with minimum gap $\ell_{\min}$ to obtain $\tilde{\mathcal{T}}$;
if $|\tilde{\mathcal{T}}| > B - 1$ **then**
  Rank boundaries by strength and keep the top $B-1$;
Let $\tau_0 = 0$, $\tau_b = L$, and $\{\tau_1, \ldots, \tau_{b-1}\} = \tilde{\mathcal{T}}$ in order;
return $B_k = V[\tau_{k-1}, \tau_k)$, $k = 1, \ldots, b$;

more powerful multimodal understanding. Representative multimodal LLMs such as GPT-4o [25], Gemini [4, 37], LLaVA-OneVision / LLaVA-NeXT-Video [15, 53], InternVL3.5 [41], Qwen2.5-VL / Qwen3-Max [38, 39], VideoLLaMA3 [50], and ARIA [16] couple strong language backbones with unified visual encoders, long-context modeling, and instruction tuning, substantially advancing open-ended image and long-form video understanding. Also, agent-based methods like VideoAgent [5], TraveLER [32] or MoReVQA [24] use multi-step reasoning for tasks with strong temporal or causal relationship.

Long video understanding. Recent benchmarks such as EgoSchema [23], LongVideoBench [42], MLVU [58], and Video-MME [7] push models to reason over minutes- to hour-long videos with complex queries and interleaved modalities, exposing the limitations of naive uniform sampling and single-pass decoding [7, 17, 23, 58]. To improve scalability, a line of work compresses or sparsifies the visual stream via token merging, adaptive compression, and content-aware resampling [1, 9, 13, 16, 18, 18, 51, 55], while others design specialized architectures or training objectives, e.g., RNN-based MLLMs and self-reward–aligned long-video learners [33, 35, 44]. Retrieval- and memory-based methods instead treat long videos as external knowledge to be indexed, using document retrieval, video-RAG, or sparse memory to focus reasoning on a small subset of salient clips [21, 22, 34, 42, 52]. Closest to ours are agentic or search-style systems that iteratively query long videos, such as VCA, MovieChat, Chapter-LLaMA, ∞-Video, and related multi-agent frameworks [5, 24, 31, 32, 34, 40, 47]. In contrast, A4VL is a training-free multi-agent alliance that combines event-based partitioning with clue-guided block selection and multi-round agent collaboration, and can be plugged on top of arbitrary MLLM backbones to improve long-video reasoning under strict frame budgets.

Agents in video understanding. Agent-based frameworks have recently been introduced for video understanding, where multiple specialized modules ("agents") collaborate via planning, tool use, and iterative querying. For short videos, systems such as ViperGPT [36], MoREVQA [24], and Traveler [32] decompose questions into sub-tasks and route them to dedicated reasoning or perception components, often improving compositionality and interpretability over clip-level inputs. For long video understanding, BOLT [20], VCA [47], VideoAgent [5], Deep Video Discovery [52], MovieChat [34], and related agentic pipelines employ global controllers, external memory, and iterative search over the temporal axis to locate relevant segments under strict token budgets. Our A4VL follows this agentic perspective but remains training-free, combining event-based partitioning, clue-guided block selection, and multi-agent consensus to robustly discover key evidence in very long videos.