Scene-VLM: Multimodal Video Scene Segmentation via Vision-Language Models

Supplementary Material

A. Additional Details

A.1. Method

A.1.1. Prompt Structure and Output Format

Fig. 8 illustrates our full prompt structure and the expected output format. The **gray** block contains the *system prompt*, the **green** block provides the *task instructions*, the **blue** block encodes the *per-shot multimodal inputs* (frames, subtitles, actor IDs) in an XML-style layout, and the **purple** block defines the *context-focus scope* (indicating which shots in the context window are also prediction targets). As depicted in the output block, the model generates `shot_id: Yes/No` decisions only for shots in the focus window. On average, a complete prompt contains 7939 words.

A.1.2. Visual Shot-ID Markers

To strengthen the correspondence between visual frames and their textual shot identifiers in the prompt, we overlay a small, high-contrast numerical tag at the top-left corner of each frame, as illustrated in Fig. 7. This visual marker serves two purposes: (1) it explicitly anchors each frame to its corresponding shot ID in the structured text input, reducing ambiguity during multimodal reasoning, and (2) it provides a consistent spatial reference that helps the model track shot boundaries across the temporal sequence. The marker is deliberately positioned to minimize occlusion of semantically relevant content while maintaining high visibility. As demonstrated in our ablation study (Sec. 4.5.1), these markers improve the final results.

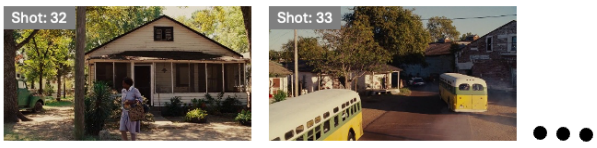


Figure 7. **Visual shot-ID markers.** A compact numerical tag is overlaid at the top-left corner of each frame, tightly coupling visual content with the corresponding textual shot IDs referenced in the prompt while preserving semantically relevant content.

A.2. Setup

A.2.1. Datasets

Explainability. To enable aligning our model to generate post-hoc rationales for its predicted scene boundaries (Sec. 4.7), we curate a small supervision set of 35 samples pairing annotated scene boundaries with human-written explanations. Each entry contains the `movie_id`, the inclusive `start_shot` and `end_shot` indices of the scene being concluded, and a free-text `rationale` describing the narrative transition. For example: *"There is a clear scene transition: the narrative shifts from an interview between two women about one woman's past to a sequence showing her working as a maid and caring for a child, changing place, time, and situation."*

A.2.2. Metrics

Scene segmentation. Following prior work [17, 27], we report the following standard detection metrics on the MovieNet [16] and BBC [6] datasets:

- **Average Precision (AP)**: area under the precision-recall curve.
- **F1**: harmonic mean of precision and recall at an optimized operating point.

Video Chaptering. Following [30, 34], we evaluate chapter segmentation and titling using:

- **Chapter F1**: boundary-detection F1 computed against creator-provided chapter endpoints. Since endpoints are continuous timestamps, matching is based on temporal overlap and averaged over multiple overlap lengths; see [30] for details.
- **tIoU**: temporal Intersection-over-Union between predicted and ground-truth chapters, measuring temporal alignment; the exact protocol follows [30].
- **SODA and CIDEr**: metrics for evaluating semantic alignment of generated titles to ground-truth titles (see [34] for details).

A.2.3. Additional Implementation Details

Frame Processing. Our vision-language model [23] can accept a user defined image size. All frames across datasets are resized to 147×63 pixels. This resolution was selected to balance computational requirements with the preservation of visual details required for training and inference.

Training. All experiments were conducted on a cluster of 8×A100 (40 GB) GPUs. We fine-tune Qwen2.5-VL-7B on MovieNet ($\sim$29k samples) using LoRA [15] (rank 8, α=16) for 4 epochs. With mixed precision, FlashAttention [12], and ZeRO-3 [25] sharding (data/optimizer/parameter partitioning), end-to-end fine-tuning on MovieNet-318 completes in approximately 2–4 hours, and on the chaptering task in around 1 hour, depending on I/O and kernel availability.

Inference. Evaluation on the MovieNet test split takes approximately 1–2 hours on 8×A100 (40GB) GPUs with data parallelism. Each movie is partitioned into non-overlapping

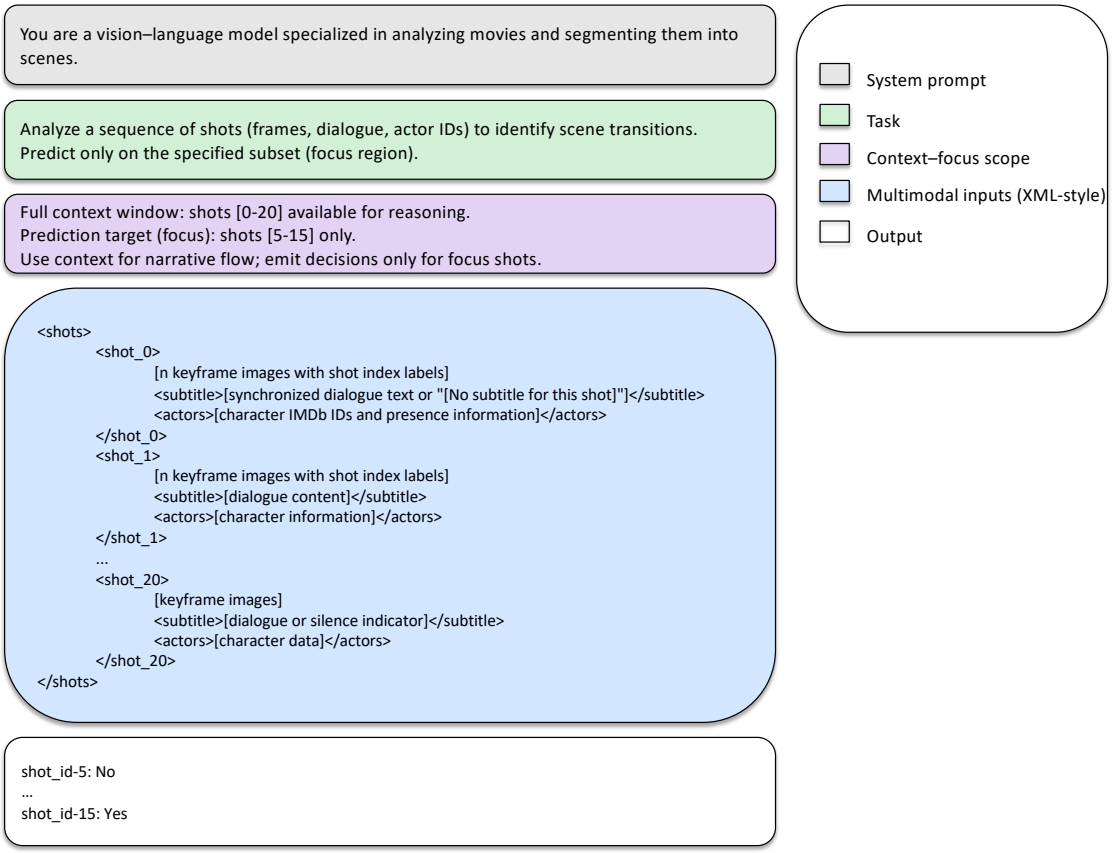


Figure 8. **Full prompt structure and output format.** Illustration of the multimodal prompt structure (instructions, visual frames, subtitles, and metadata) and the model's structured output with shot-level boundary predictions.

context windows. We run batch-wise sequential decoding per window and then aggregate the outputs to compute the final metrics. This inference procedure applies to both tasks (scene segmentation and video chaptering).

A.2.4. Chapter-LLaMA Adaptation for MovieNet

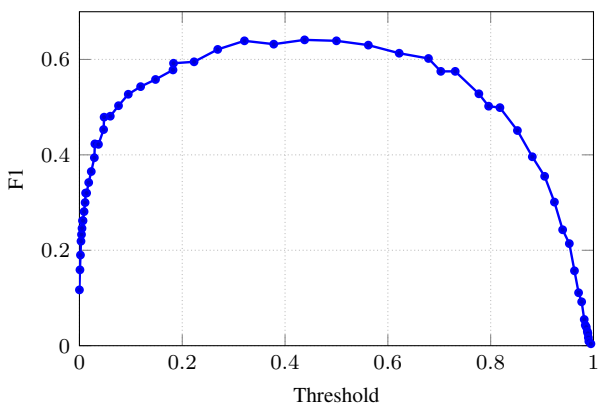
Adapting Chapter-LLaMA [30] to MovieNet is conceptually straightforward but requires addressing a modality gap. While Chapter-LLaMA relies on rich transcripts and keyframe captions, MovieNet, being cinematic, often exhibits sparser dialogue while scene changes are predominantly *visual*. Hence, directly applying Chapter-LLaMA to MovieNet under-detects boundaries. For a fair comparison, we generate a caption *per-shot* using the authors' VLM [35] to supply the missing visual semantics, and feed these captions to Chapter-LLaMA in the *exact* input format specified in [30]. This preserves their pipeline while aligning the visual signal with MovieNet's shot structure.

B. Additional Experiments

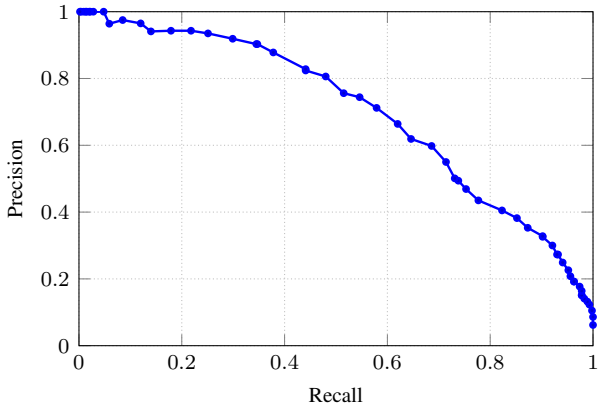
B.1. Alternative Prediction Schemes

Confidence scores are essential for scene segmentation, enabling flexible precision–recall trade-offs across different operating points. However, extracting reliable confidence from VLM outputs is non-trivial, as VLMs produce structured textual responses where confidence must be inferred from token-level probabilities rather than dedicated classification heads. In this section, we compare our proposed approach against two alternative prediction schemes, which differ in output format, confidence estimation capability, accuracy, and computational cost.

Comprehensive scheme. Our proposed scheme (see Sec. 3.2) generates structured outputs in the format `shot_id:<id>: Yes/No` for each shot in the focus window, providing *explicit* predictions for all target shots. Confidence is then computed from the normalized token logits as described in Eq. (3). As shown in Tab. 10, this



(a) **F1 versus decision threshold.** Peak $\mathrm{F1} = 0.641$ at threshold ≈ 0.438. Broad plateau $\approx$ 0.62–0.64 indicates stable operating range.



(b) **Precision–Recall curve.** Showing a balanced knee. E.g., at $t \approx 0.321$: $P \approx 0.60$, $R \approx 0.69$; at $t \approx 0.50$: $P \approx 0.71$, $R \approx 0.58$.

Figure 9. **F1 and Precision–Recall curves across decision thresholds.** Both curves demonstrate strong operating flexibility for task-dependent tuning.

scheme achieves strong performance (F1: 62.1, AP: 66.8), but requires an average inference time of 87.2 seconds per movie.

Concise scheme. While our Comprehensive scheme is accurate, its inference latency may be prohibitive for latency-sensitive applications. To address this, we explore a more efficient output format where the model emits only `Yes` tokens for detected boundaries, omitting explicit `No` predictions for non-boundary shots. However, a subtle question then arises: can we extract confidence scores from this format by probing the `Yes` token logits? Unfortunately, this approach is fundamentally flawed due to the model's output structure. When the model predicts a shot ID, it has already committed to marking that shot as a boundary, which means that the subsequent `Yes` token is obligatory rather than a genuine binary choice. Consequently, $p(\texttt{Yes} \mid \text{shot\_id\_predicted}) \approx 1$ by design, making the confidence score uninformative. This contrasts sharply with the Comprehensive scheme, where the `Yes`/`No` choice represents a genuine binary decision with meaningful probability mass on both outcomes.

Given this limitation, we evaluate the Concise scheme **without confidence extraction**, using the model's outputs directly: a shot is classified as a boundary if and only if the model predicts `Yes` for it. Despite lacking precision–recall flexibility, this variant achieves competitive performance (F1: 53.4) with recent methods such as TranS4mer (48.4) and MEGA (55.3), while delivering dramatic speedup over the Comprehensive scheme (10.5s per movie, approximately 8.3$\times$ faster). This makes it an attractive option when inference speed is critical and precision–recall control is not required.

Concise scheme with repeated sampling. Given the inability to extract meaningful confidence using the Concise scheme, we investigate whether repeated sampling can provide reliable confidence estimates. Specifically, we perform m=5 independent inference runs, draw temperatures uniformly from the interval $[0.5, 1.0]$ for each run, and compute confidence per shot as the proportion of `Yes` outcomes. Unfortunately, as shown in Tab. 10, this approach fails on both fronts: it is less accurate than Concise without confidence (F1: 52.6 vs. 53.4) and even slower than the Comprehensive scheme (105.2s vs. 87.2s), making it strictly worse than both alternatives.

To summarize, the **Comprehensive scheme** is recommended when accuracy and precision–recall control are paramount, while the **Concise scheme (without confidence)** may be a good alternative when inference speed is the priority and precision–recall control is not strictly required.

Table 10. **Comparison of prediction schemes for scene segmentation.** The Comprehensive scheme achieves the best accuracy and features a confidence prediction capability, while the Concise scheme (without confidence) offers a compelling speed-accuracy trade-off for latency-critical applications.

Prediction Scheme	F1 ↑	AP ↑	Avg. time / movie (s) ↓
Concise (without confidence)	53.4	-	**10.5**
Concise (repeated sampling)	52.6	34.7	105.2
Comprehensive	**62.1**	**66.8**	<u>87.2</u>

B.2. Model F1 and Precision–Recall Analysis

To assess our method's sensitivity to threshold changes, we plot F1 versus decision threshold (Fig. 9a) alongside the corresponding precision–recall curve (Fig. 9b). As depicted, the F1 curve rises sharply from near-zero thresholds and reaches a broad plateau, peaking at $\mathrm{F1} = 0.641$ around threshold 0.438. This plateau ($\approx$0.62–0.64 F1 across thresholds $\sim$0.27–0.56) demonstrates stable performance with minimal sen-

sitivity to threshold variations. The PR curve exhibits the expected trade-off between precision and recall. For *balanced* operation, a threshold near 0.321 yields $P \approx 0.60$ and $R \approx 0.69$ ($F1 = 0.639$). For *precision-oriented* applications, a threshold near 0.50 yields $P \approx 0.71$ and $R \approx 0.58$ ($F1 = 0.639$). Conversely, recall-oriented scenarios can use lower thresholds (e.g., 0.223–0.269) to push recall above 0.70 with modest precision. To summarize, these curves demonstrate that flexible task-dependent tuning is possible while maintaining robust performance across the F1 plateau.

B.3. Zero-Shot VLM Baselines

To assess the necessity of fine-tuning, we evaluate zero-shot performance on MovieNet using two VLMs: Qwen2.5-VL-7B [5] (our base model without fine-tuning) and Claude 4.5 Sonnet [4] (a strong closed-source VLM). Due to Claude API constraints, we use 1 frame per shot for all methods in this comparison. Since zero-shot models do not reliably follow the required output format, we use LLM-based parsing to extract their predictions and treat unparsed shots as negatives. As shown in Tab. 11, zero-shot methods significantly underperform our fine-tuned model. Qwen2.5-VL-7B achieves only 11.1 F1 with a 7.9% parse error rate, while Claude 4.5 Sonnet reaches 37.6 F1 with near-zero parse errors (0.03%). In contrast, Scene-VLM achieves 61.8 F1 with zero parse errors using simple rule-based parsing. Moreover, zero-shot methods cannot provide meaningful confidence scores for their predictions: for closed-source models (e.g., Claude) logits are inaccessible; for Qwen, LLM-based parsing makes confidence extraction infeasible while rule-based parsing fails 71% of the time (not shown). These results confirm that fine-tuning is essential for reliable scene segmentation.

Table 11. **Zero-shot VLM baselines on MovieNet.** Fine-tuning is essential: zero-shot methods significantly underperform and cannot provide confidence scores. All methods use 1 frame per shot.

Method	**Fine-Tuned**	**Parse**	**Parse Err.** ↓	**F1** ↑	**AP** ↑
Qwen2.5-VL-7B [5]	×	LLM	7.9%	11.1	–
Claude 4.5 Sonnet [4]	×	LLM	0.03%	37.6	–
Scene-VLM (7B)-1F (ours)	✓	Rule	**0%**	**61.8**	**65.3**

B.4. Computational Analysis

In this section, we present a thorough analysis of our models' computational requirements. We compare four variants, varying the number of frames per shot (1F or 3F) and model size (3B or 7B). For all models, we use a batch size of 1 to enable a fair comparison of memory and runtime. We do not apply model/system optimizations such as weight/activation quantization (e.g., 4-/8-bit) or inference engines with KV-cache optimizations (e.g., `vLLM`); these could further reduce both latency and memory in future work.

Table 12. **Computational analysis.** Peak memory, latency, and accuracy at 10 samples. "F" denotes frames per shot. Mean and standard deviation values are computed over five runs.

Method	Params	Memory (GB) ↓	10-sample latency (s) ↓	F1 ↑	AP ↑
TranS4mer-3F [17]	37M	**1**	$\mathbf{0.24 \pm 0.0}$	48.4	60.8
Scene-VLM-1F	3B	7	1.15 ± 0.07	55.7	58.2
Scene-VLM-3F	3B	9	1.35 ± 0.08	59.6	62.8
Scene-VLM-1F	7B	16	1.84 ± 0.15	61.8	65.3
Scene-VLM-3F	7B	18	2.34 ± 0.14	**62.1**	**66.8**

We report peak memory, latency, and accuracy at *10 samples* (i.e., 10 binary boundary decisions) in Tab. 12, using the same configuration as in the main experiments: a context window of 20 shots and a focus window of 10 shots. We repeat the evaluation five times, reporting the mean and standard deviation for wall-clock latency, and the maximum over runs for peak memory. As depicted, reducing frames per shot from 3F to 1F in the 7B model lowers latency from 2.34 s to 1.84 s (≈21%) and peak memory from 18 GB to 16 GB, with only a minor accuracy drop (F1/AP: 62.1/66.8 → 61.8/65.3); a similar trend holds for the 3B model. The latency reduction stems from the fact that frames dominate the token count of the input, so removing two of three frames per shot shortens the length of the multimodal input sequence and reduces computation. Meanwhile, the small accuracy drop aligns with our results from Sec. 4.5.3, which shows that performance degrades modestly when reducing the number of frames per shot. Intuitively, since shots are segments which typically contain no major visual changes, a single representative frame often preserves most scene-transition-relevant information.

We also compare against TranS4mer [17] (MEGA [27] has not released source code), using the same 3 frames per shot configuration for a fair comparison. As shown in Tab. 12, TranS4mer is faster and more memory-efficient (37M parameters, 1 GB memory, 0.24 s latency), but Scene-VLM offers significantly higher accuracy (+13.7 F1) along with explainability capabilities that are unavailable in encoder-based methods. Notably, our smaller 3B variant narrows this efficiency gap while still outperforming TranS4mer by +11.2 F1. Finally, we note again that no inference optimizations were applied to our models; such techniques could further reduce our latency and memory.

B.5. Explainability for Scene Segmentation (Cont.)

We present additional qualitative examples of model-generated rationales for scene-boundary decisions in Fig. 10 and Fig. 11. These examples span both abrupt visual transitions (e.g., title cards) and subtler, socially driven changes (e.g., shifts in location, time, or conversational structure).

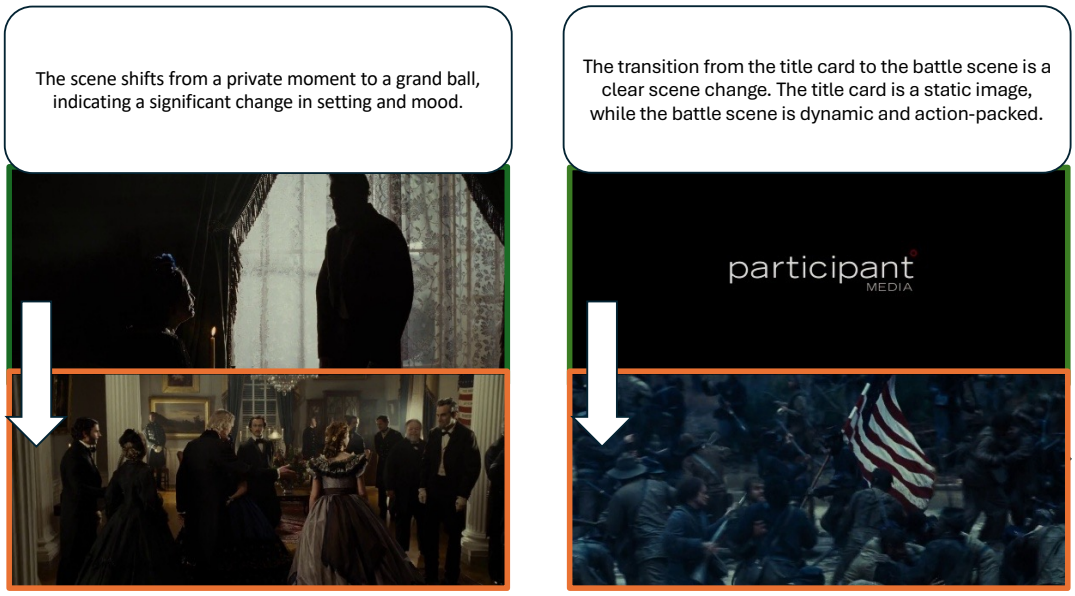


Figure 10. **Example 1.** *Left:* boundary due to a location change. *Right:* transition from a title card to the opening shot.

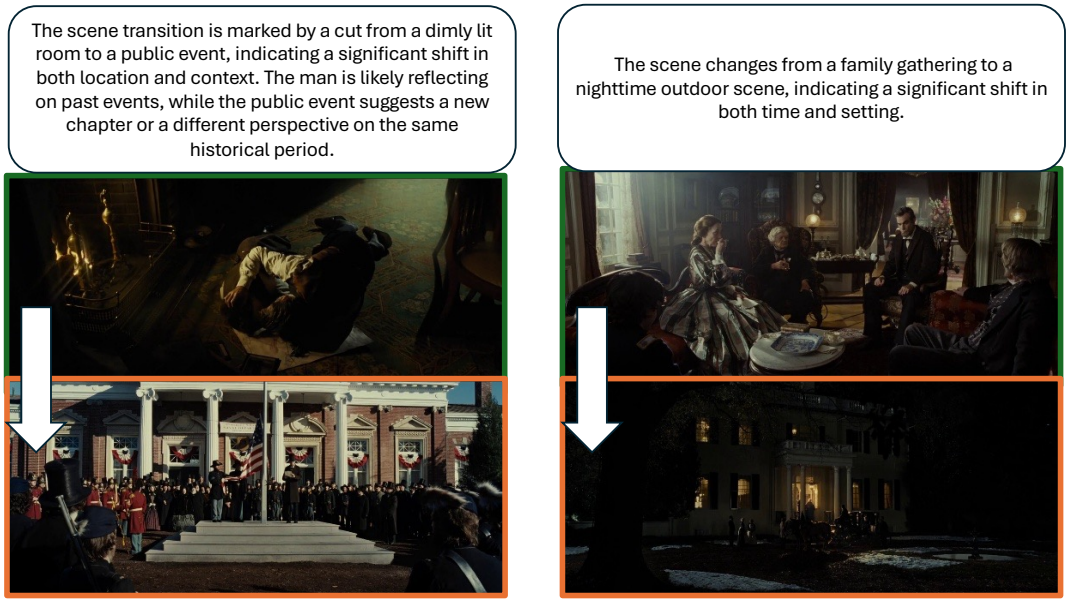


Figure 11. **Example 2.** *Left and right panels:* boundary justified by a joint change in *time* and *place*; the model references visual cues (lighting, background) and/or dialogue context to support the decision.