BoxTuning: Object-Aware Visual Prompting for Multimodal Model Fine-Tuning

Zekun Qian[1], Wei Feng[1], Ruize Han[2]*
[1]School of Computer Science and Technology, Tianjin University
[2]Faculty of Computer Science and Artificial Intelligence, Shenzhen University of Advanced Technology
{clarkqian, wfeng}@tju.edu.cn, hanruize@suat-sz.edu.cn

Abstract

Object-level spatial-temporal understanding is essential for video question answering, yet existing multimodal large language models (MLLMs) encode frames holistically and lack explicit mechanisms for fine-grained object grounding. Recent work addresses this by serializing bounding box coordinates as text tokens, but this text-coordinate paradigm suffers from a fundamental *modality mismatch*: object information is inherently visual, yet encoding it as text incurs a high token cost that forces aggressive temporal downsampling. We propose *BoxTuning*, an object-aware visual prompting framework for multimodal model fine-tuning that moves spatial-temporal object states from text-coordinate serialization into the visual stream while retaining only object identity in a compact text legend. Colored bounding boxes and trajectory trails encode geometry and motion on video frames, while the color-to-object legend provides the minimal textual identity bridge. This reduces the token cost significantly, achieving 87–93% text token reduction in practice. It also preserves full temporal resolution, where the trajectory trails further encode inter-frame motion direction and speed within each keyframe, recovering fine-grained dynamics that text-coordinate methods are forced to discard. Experimental results on five video QA benchmarks (CLEVRER, Perception Test, STAR, NExT-QA, IntentQA) show that BoxTuning surpasses text-coordinate baselines on spatially oriented tasks and avoids their degradation on reasoning-centric tasks, establishing object-aware visual prompting as a natural and efficient way to convey object information to MLLMs. The code is available at https://github.com/zekunqian/BoxTuning.

1 Introduction

Video understanding requires models to jointly reason over appearance, motion, and interaction across time, a challenge that becomes especially acute when the scene contains multiple interacting agents and objects [1–4]. A broad range of video question answering tasks demand *object-level* grounding: identifying *what* objects are present, *where* they are located, and *how* they move and interact over time. Such understanding goes beyond simple video recognition (classification). For example, answering "*Will the ball hit the red cylinder?*" requires tracking spatial trajectories across frames [1]; answering "*Why did the man pick up the bag?*" requires grounding the agent and object through a causal chain [4]; answering "*What does the person do after sitting down?*" requires localizing the actor and linking actions to objects in sequence [2].

For visual understanding, multimodal large language models (MLLMs) [5–8] have achieved remarkable progress by bridging vision encoders with large language models. Recent efforts have extended these architectures to video, producing a family of video MLLMs [9–13] that process temporal

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



Figure 1: Comparison of object-centric representation strategies for video MLLMs. *Left*: ObjectMLLM [15] serializes per-frame bounding box coordinates as text tokens (*e.g.*, "`(Object 0) adult: frame 0 [73 30 92 100] ...`"), producing 1,218 tokens for a single video. The original keyframes lack intermediate dynamics, leading to action misinterpretation. *Right*: BoxTuning renders colored bounding boxes and trajectory trails directly on the video frames, with only a concise color-mapped text legend (*e.g.*, "`(1) Red box: Adult; (2) Blue box: Elephant; ...`", 182 tokens). The trajectory-augmented keyframes capture continuous dynamics, enabling accurate temporal reasoning. Overall, BoxTuning achieves 85% token savings (in this case) while bridging temporal gaps that text-coordinate methods miss.

video sequences. Despite the progress, current MLLMs struggle to handle the above object-level understanding tasks. This is because these models encode each frame *holistically* through a vision encoder (*e.g.*, CLIP ViT [14]), producing frame-level feature representations that lack explicit information about individual objects. As a result, the language model must implicitly infer object identities, locations, and movements from distributed visual features, a task that becomes increasingly difficult as the number of objects and the complexity of their interactions grow. In general, a common bottleneck of MLLMs is the lack of an explicit, efficient mechanism to communicate the object-level information to the language decoder.

To address this gap, Tang *et al.* [15] propose ObjectMLLM, which augments the MLLM input with structured bounding box information from an off-the-shelf detector and tracker. Each object's bounding box coordinates across frames are serialized as text tokens and appended to the prompt (*e.g.*, "`Boy: frame 1 [60 58 70 98] frame 2 [57 48 70 95] ...`"). Such a text-coordinate representation proves highly effective on benchmarks that emphasize spatial-temporal object understanding: ObjectMLLM significantly improves accuracy over the baseline VideoLLaMA2 [9] on synthetic collision reasoning (CLEVRER [1]) and real-world perceptual understanding (Perception Test [16], STAR [2]). The success of ObjectMLLM establishes a clear message: *explicit object-level spatial-temporal information is valuable for video understanding*.

However, the pure text-coordinate representation introduces a fundamental efficiency bottleneck. As shown in the left panel of Figure 1, the token cost scales as $\mathcal{O}(N \times T \times C)$, where N is the number of tracked objects, T is the number of sampled frames, and $C \approx 16$–22 is the token cost per coordinate entry (*e.g.*, "`frame 1 [60 58 70 98]`"). To keep the total token count manageable, ObjectMLLM must temporally downsample the bounding box annotations at rates far below the original video frame rate. For instance, on STAR [2] with dense interacting objects, the bounding boxes are sampled at only one frame per 120 original frames (1/120); on NExT-QA [3] the rate is approximately 1/60, and even on the synthetic CLEVRER [1] with relatively few objects the rate reaches 1/25. Such coarse temporal sampling inevitably discards fine-grained motion information that is precisely what the bounding box coordinates are meant to capture. Even after this aggressive downsampling, the text overhead remains prohibitively large. The mean box token count still reaches 1,184 per sample on NExT-QA, roughly 27$\times$ larger than the question tokens (mean 43 tokens). Limited by the computing memory budget, the overlong box representation tokens may overflow the context window, and even adversely affect the visual tokens and other text tokens from the question and answer.

We identify the root cause of these problems as a *modality mismatch*: object spatial-temporal information, including locations, sizes, and motion trajectories, is inherently visual, yet the text-coordinate paradigm forces it through the language modality. This creates a direct budget conflict: comprehensive object tracks require many coordinate tokens, while reducing the token cost requires aggressive temporal downsampling and discards the fine-grained motion cues needed for video QA. Meanwhile, the vision encoder already processes every sampled frame, but the explicit object-track information is still routed almost entirely through the language stream. Such being the case, *why not directly encode object geometry and motion into the video (visual modality)?*

Based on this insight, we propose *BoxTuning*, an object-aware visual prompting framework for video MLLMs built around a modality-aligned object-track representation. Rather than encoding all object states as text-based symbolic sequences as in ObjectMLLM [15], BoxTuning assigns each type of object information to its natural modality: geometry and motion are rendered visually, while category identity remains textual through a concise legend. This representation-level change directly targets both problems above, *i.e.*, excessive text token consumption and the loss of temporal motion information due to downsampling. As illustrated in the right panel of Figure 1, BoxTuning introduces two complementary visual prompts. First, we render **colored bounding boxes** directly onto each video frame, where each tracked object is assigned a distinct color from a predefined palette. The color serves as the bridge between the visual and textual modalities: the only text required is a concise legend mapping colors to object classes (*e.g.*, "`(1) Red box: Adult; (2) Blue box: Elephant; ...`"), reducing the text cost from $\mathcal{O}(N \times T \times C)$ to $\mathcal{O}(N \times C')$, where $C' \approx 9$ tokens per object is far smaller than the per-frame coordinate cost $C \approx 18$, and more importantly the temporal factor T is entirely eliminated. Second, to compensate for the temporal information lost between sampled frames, we draw **temporal trajectory trails** that connect the bounding box centers of each object over recent frames. This is inspired by Johansson's classical biological motion experiments [17], which demonstrated that humans can recognize complex actions from point-light trajectory displays alone, suggesting that trajectory patterns are a powerful and natural encoding of motion. Our trails explicitly encode motion direction and speed as visual patterns within a single frame, recovering the fine-grained motion cues that the text-coordinate approach is forced to discard through temporal downsampling.

This encoding approach yields three concrete advantages over text-coordinate representations: 1) **Dramatic token reduction.** The text cost drops from $\mathcal{O}(N \times T \times C)$ to $\mathcal{O}(N \times C')$. In practice, this amounts to an 87%-93% reduction, freeing context window capacity for the visual and question/answer text tokens. 2) **Lossless temporal resolution.** We can render each object's motion trajectory across all intermediate frames directly onto the sampled keyframes. This preserves the fine-grained motion cues that text-coordinate approaches are forced to discard through aggressive temporal downsampling. 3) **Perceptually aligned encoding.** Colored bounding boxes and trajectory trails express object location and motion as visual patterns directly on the frames, matching the modality in which these signals are observed. The compact legend only names the colored objects, making the prompts easy to inspect while avoiding long symbolic coordinate strings.

We apply BoxTuning to VideoLLaMA2-7B [9] with LoRA [18], using the same object detection and tracking pipeline as ObjectMLLM [15], ensuring that any performance difference stems solely from the representation rather than the object perception quality. On five benchmarks, CLEVRER [1], Perception Test [16], STAR [2], NExT-QA [3], and IntentQA [4], BoxTuning exceeds ObjectMLLM's accuracy while consuming only about 10% of the text tokens required by the text-coordinate representation. Our main contributions are:

- We introduce BoxTuning, a unified object-aware visual prompting framework for video MLLM fine-tuning that integrates colored bounding boxes, trajectory trails, and a compact object legend into a modality-aligned representation.
- In BoxTuning, we encode spatial object bounding boxes and temporal trajectory trails as visual overlays without additional text-token overhead, and use only a small number of text legend entries for object identity. Through systematic analysis of token efficiency, compared to ObjectMLLM, BoxTuning achieves 87%–93% text token reduction.
- Experimental results on five video QA benchmarks show consistent improvements over ObjectMLLM, with gains on object-sensitive tasks and reduced degradation on reasoning-centric tasks. Comprehensive ablation studies validate the contribution of each component.

2 Related Work

2.1 Video Understanding with MLLMs

Large language models have inspired extensive work on video MLLMs [19, 11, 10, 9, 13, 12], where a central challenge is to represent rich spatiotemporal information efficiently. Most methods extend image MLLMs [5–8] by applying pretrained image encoders [14] to sampled frames, then pooling, concatenating, or projecting frame features into the language space. Representative systems improve this pipeline with temporal pooling [11], Video Q-Former alignment [10], spatial-temporal convolution connectors [9], unified image-video alignment [20], progressive multi-modal training and benchmarks [12], dynamic visual tokens [13], frame compression for long videos [21], and cross-scenario image-to-video transfer [22]. Another line represents videos as captions or structured text for LLM consumption [23, 24], which improves interpretability but often loses fine-grained spatiotemporal cues such as object trajectories [15]. Recent studies further reveal systematic visual shortcomings of current MLLMs [25, 26]. Overall, most video MLLMs still encode frames holistically and lack explicit mechanisms for object-level spatial reasoning, limiting their ability to model individual object interactions and movements.

2.2 Object-Level Spatial Representations in MLLMs

Explicit object information has long been used in vision-language models. Oscar [27] uses detected object tags as cross-modal anchors, VinVL [28] shows the benefit of stronger region features, and CoVLM [29] composes visual entities and relationships through communication tokens. Recent MLLMs encode boxes as text for referential dialogue [30, 31], inject region-of-interest features [32], align masks with text [33, 34], build spatial QA data [35], and study object-centric inductive biases for efficient foundation-model representations [36]. In video, ObjectMLLM [15] serializes per-frame bounding box coordinates as text tokens and improves motion-related video QA, while Vamos [24] incorporates object state descriptions. However, text-coordinate token cost scales with both object count and frame count, creating the efficiency bottleneck discussed in Section 1.

Instead of encoding spatial information as text, another strategy modifies the visual input directly. Prior work shows that red circles can guide VLM attention without training [37], numbered markers can support spatial referencing [38], and visual prompts such as arrows, boxes, and circles can be learned by MLLMs [39]. Related methods also use internal representations for visual cropping [40], analyze how MLLMs process small details [41], or guide visual search in high-resolution images [42]. These methods show that visual modifications can convey spatial information, but they mainly target static images, single-region highlighting, or attention guidance. BoxTuning instead treats object tracks in video QA as a modality-aligned visual/text representation, directly addressing the token growth and temporal downsampling bottlenecks of ObjectMLLM-style text-coordinate prompting.

3 Method

We aim to enhance video MLLMs with explicit object-level spatial-temporal information while avoiding the modality mismatch and token overhead of text-coordinate object-track representations. To this end, we propose BoxTuning, an object-aware and modality-aligned visual prompting framework that renders object geometry and motion directly on video frames while preserving object identity in a compact text legend. Figure 2 illustrates the overall pipeline. We first formulate the problem and contrast our approach with the text-coordinate paradigm (Section 3.1), then describe the object detection and tracking pipeline (Section 3.2), the visual prompt construction (Section 3.3), the concise text legend (Section 3.4), and the fine-tuning procedure (Section 3.5). Implementation details are provided in Section A.

3.1 Problem Formulation

Consider a video QA task where a multimodal model must produce an answer A given an input video $V = \{v_1, v_2, \ldots, v_T\}$ of T *uniformly sampled frames* and a text question Q. We enhance the model with explicit object-level spatial information: an off-the-shelf detector and tracker (Section 3.2) provide N *tracked objects*, each with a class label l_i and per-frame bounding boxes. The key design

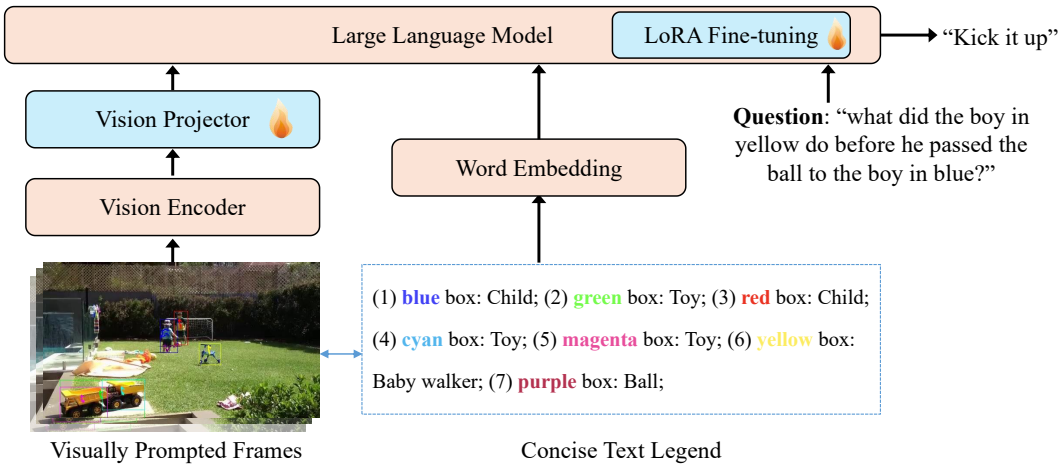


Figure 2: Overview of the BoxTuning framework. Visually prompted frames, produced by rendering colored bounding boxes and trajectory trails from an off-the-shelf detector (YOLO-World [43]) and tracker (SAM 2 [44]), are fed into the vision encoder together with a concise text legend mapping colors to object classes and the question. The vision encoder is kept frozen, while the vision projector (STC connector) is fully fine-tuned and the LLM is adapted via LoRA [18].

choice is the *representation manner* through which the spatial-temporal information produced by the detector and tracker reaches the multimodal model.

ObjectMLLM [15] conveys all spatial-temporal information through text. Each object's coordinates across frames are serialized into a token sequence:

$$l_i\colon\ \texttt{frame 1}\ [x_1^1\ y_1^1\ x_2^1\ y_2^1],\ \texttt{frame 2}\ [x_1^2\ y_1^2\ x_2^2\ y_2^2]\ \ldots\texttt{frame T}\ [x_1^T\ y_1^T\ x_2^T\ y_2^T],$$

where coordinates are quantized to $[0, 100]$. The concatenation of all such strings is appended to the prompt alongside Q, incurring a text cost of $\mathcal{O}(N \times T \times C)$ tokens (where C denotes the token requirement for the object's coordinates, $C{\approx}16$-22 per entry), while the video frames remain unmodified.

The proposed BoxTuning instead conveys spatial-temporal information directly through the visual modality. It renders colored bounding boxes and trajectory trails directly onto the video frames, producing augmented frames $\hat{V} = \{\hat{v}_1, \ldots, \hat{v}_T\}$. Compared with ObjectMLLM [15] using V, the computational load of visual embedding does not increase.

The only text required is a concise legend D that maps each color to its object class. The model input thus becomes:

$$\text{Input} = \big[\hat{V};\ D;\ Q\big], \tag{1}$$

reducing the text cost from $\mathcal{O}(N \times T \times C)$ to $\mathcal{O}(N \times C')$, where C' denotes the token requirement of D for each object. The following subsections describe the construction of $\hat{V}$ and D.

3.2 Object Information Acquisition

To obtain object bounding boxes across video frames, we employ a detect-then-track strategy using two complementary off-the-shelf models, following the same detection and tracking pipeline as ObjectMLLM [15] to ensure a fair comparison.

Open-vocabulary detection. We use YOLO-World [43], an open-vocabulary object detector that accepts arbitrary category names as text prompts. For each benchmark, we define a category vocabulary covering the object types relevant to that dataset. Detection is performed on uniformly sampled keyframes, followed by non-maximum suppression to remove duplicate detections.

Mask-based tracking. Initial detections are propagated across all frames using SAM 2 [44] (Segment Anything Model 2), which provides robust mask-based video object segmentation. Given the bounding box prompts from the initial detections, SAM 2 generates per-frame segmentation masks

that are robust to occlusions, scale changes, and appearance variations. The per-frame bounding box $b_i^t=(x_1^t, y_1^t, x_2^t, y_2^t)$ for object o_i in frame v_t is derived as the tightest axis-aligned rectangle enclosing the predicted mask.

Iterative detection for new objects. Objects may enter the scene after the first frame. To handle this, we run YOLO-World on multiple uniformly spaced keyframes throughout the video. For each subsequent keyframe, we compute the IoU between new detections and existing tracks; detections with high IoU against any existing track are discarded as duplicates, and the remaining ones initialize new tracks via SAM 2. This iterative strategy ensures comprehensive object coverage across the entire video.

3.3 Visual Prompt Construction

The core of BoxTuning is the construction of visual prompts that encode object spatial-temporal information directly on the video frames. We compose two complementary elements: colored bounding boxes for spatial localization and trajectory trails for motion encoding, rendered with a consistent color scheme across all frames.

Color assignment. Each tracked object o_i is assigned a unique color c_i from a predefined palette of K maximally distinguishable colors (*e.g.*, red, blue, green, orange, purple, cyan, ...), as illustrated in Figure 3(a). The same color is used consistently for a given object across all frames, enabling the model to associate the same object across time through color consistency.

Bounding boxes for spatial localization. For each object o_i visible in frame v_t, we draw a colored rectangle with color c_i around its bounding box b_i^t. The box provides precise per-frame spatial localization, allowing the model to directly 'see' where each object is, without parsing coordinate numbers from text.

Trajectory trails for motion encoding. For the video understanding task, temporal information is critical. Therefore, besides the object box for each frame, next we consider encoding the temporal information of the objects. For video object perception, the motion trajectory is a commonly used representation of temporal information [45]. In classical MLLM-based video understanding methods, only the sampled T frames are used for visual embedding. This way, to encode temporal motion trajectory, we draw a polyline connecting the bounding box centers of object o_i over the most recent L consecutive original frames

$$\text{trail}_i^t = \big\{(\bar{x}_i^{t-L}, \bar{y}_i^{t-L}),\ \ldots,\ (\bar{x}_i^{t-1}, \bar{y}_i^{t-1})\big\}, \tag{2}$$

where $(\bar{x}_i^t, \bar{y}_i^t)$ denotes the center of b_i^t, as in Figure 3(b). The trail length L is set equal to the sampling interval between two consecutive model input frames, *i.e.*, $L = \lfloor T_{\text{video}}/T \rfloor$, where T_{video} is the total number of original video frames and T is the number of uniformly sampled frames. This design ensures that the trail on each sampled frame covers exactly the temporal gap from the previous sampled frame to the current sampled frame, so consecutive frames have adjacent trails that together reconstruct the complete motion trajectory without gaps.

Trajectory trails serve a purpose analogous to motion vectors in video codecs: they encode the direction and speed of object movement as a visual pattern rather than numerical values. This is particularly valuable for causal and predictive reasoning, *e.g.*, 'Will the ball hit the cylinder?', where understanding motion direction is essential.

The augmented frame is obtained by compositing all overlays onto the original frame:

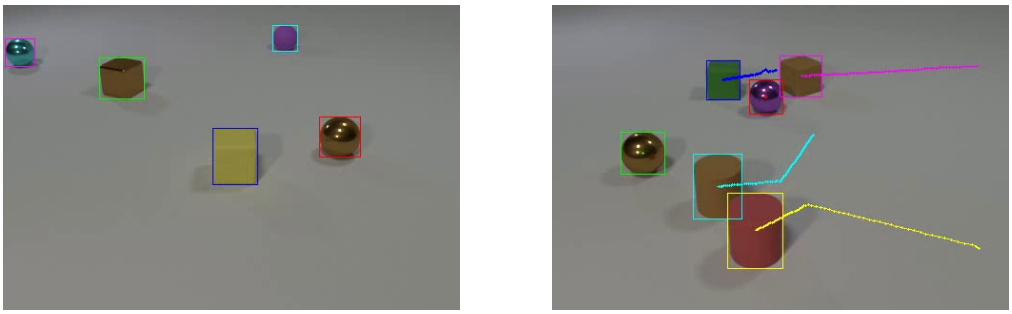
$$\hat{v}_t = \text{Render}\big(v_t,\ \{(b_i^t, c_i, \text{trail}_i^t)\}_{i=1}^{N_t}\big), \tag{3}$$

where N_t is the number of visible objects in frame v_t. The trail is drawn in color c_i from the box b_i^t, making the motion history of each object explicitly visible within a single frame. The rendering is performed to preserve the underlying visual content.

3.4 Concise Text Legend

While the visual prompts encode *where* objects are and *how* they move, the model still needs to know *what* each colored object is. We provide this information through a concise text legend:

```
(1) blue box:  Child; (2) green box:  Toy; (3) red box:  Child; ...
```



(a) Box only (b) Box + trajectory trail

Figure 3: Visual prompt examples on CLEVRER video frames. (a) Bounding boxes alone provide spatial localization of each object. (b) Adding trajectory trails encodes motion direction and speed across intermediate frames, aiding temporal reasoning. The text legend (not shown) maps each color to its object class.

where different colors can indicate the same category. This legend requires only $\sim$9 tokens per object (index + color name + class label), compared to the $\sim$18 tokens *per object per frame* required by the text-coordinate format. Table 1 provides a concrete breakdown using NExT-QA as an example, with $N{\approx}8$ tracked objects. ObjectMLLM requires $\sim$1,300 text tokens for object spatial-temporal information, while BoxTuning's color legend requires only $\sim$72 tokens, a reduction of over 94%. The measured averages across all benchmarks are reported in Table 2 in the experiment section.

Table 1: Token cost breakdown for a representative NExT-QA video ($N{\approx}8$ objects, $T{\approx}9$ temporally sampled frames per object).

Component	ObjectMLLM	BoxTuning (Ours)
Per-entry cost	$\sim$18 tok (frame + coords)	$\sim$9 tok (index + label + color)
Entries per video	$N \times T \approx 8 \times 9 = 72$	$N \approx 8$
Total object text tok	$\sim 18 \times 72 =$ **1,296**	$\sim 9 \times 8 =$ **72**

This token reduction has two practical benefits. 1) *Embedding space saving*: the token overhead is only marginally larger than the base MLLM without object input, leaving more capacity for video and question tokens. 2) *Temporal downsampling needless*: since the text tokens in our prompting manner are not related to the number of frames, we can annotate every sampled frame at full temporal resolution, preserving fine-grained motion cues that text-coordinate methods [15] are forced to discard.

3.5 Fine-Tuning Strategy

We adopt VideoLLaMA2-7B [9] as the base model, which consists of a CLIP ViT-L/14 vision encoder, a spatial-temporal convolution (STC) connector (vision projector) that captures cross-frame dependencies, and a Mistral-7B [46] language model. As shown in Figure 2, we keep the vision encoder frozen and fine-tune the STC connector with full-parameter updates together with the language model via LoRA [18], using the same learning rate for both. This design keeps the visual features intact while allowing the connector and language model to jointly learn to interpret the colored overlays and trajectory trails.

The training objective is the standard autoregressive next-token prediction loss on the answer tokens:

$$\mathcal{L} = -\sum_{j=1}^{|A|} \log P_\theta\big(a_j \mid \hat{V}, D, Q, a_{<j}\big), \tag{4}$$

where $A = (a_1, \ldots, a_{|A|})$ is the ground-truth answer, θ includes the STC connector and LoRA adapter parameters. For fair comparison, we use the same training data splits and evaluation protocols as ObjectMLLM [15] on all benchmarks. More implementation details can be found in Section A.

4 Experiments

4.1 Evaluation Benchmarks

We include five video QA benchmarks spanning synthetic and real-world data:

CLEVRER [1] contains synthetic videos of moving objects with questions about existence, motion direction, counting, attributes, and counterfactual inference. Following ObjectMLLM [15], we use the multi-choice subset (CLEVRER-MC) converted by MVBench [12].

Perception Test [16] is a diagnostic benchmark evaluating memory, abstraction, physics, and semantics across video and audio.

STAR [2] tests situated reasoning about interactions, sequences, predictions, and feasibility in real-world activity videos.

NExT-QA [3] targets causal (why/how), temporal (when/before/after), and descriptive (what) reasoning in videos. The emphasis on causal and temporal inference means that high-level action understanding often matters more than precise object localization.

IntentQA [4] evaluates understanding of human intent behind actions in videos. Like NExT-QA, it is reasoning-centric, requiring the model to infer *why* and *how* people act rather than simply locating objects.

4.2 Token and Inference Efficiency Analysis

A central claim of BoxTuning is the reduction in text token cost and the resulting improvement in computational efficiency. Table 2 quantifies the text token reduction across all five benchmarks. ObjectMLLM [15] requires 611–1,241 text tokens per sample for bounding box descriptions even after temporal downsampling, consuming the majority of the available text budget. BoxTuning requires only 61–120 tokens regardless of the number of frames, achieving an 87–93% reduction. This reduction is especially important for reasoning-centric benchmarks, where long coordinate strings can compete with question-answer context instead of providing useful object evidence.

Table 2: Average text tokens for object information per sample. ObjectMLLM [15] serializes bounding box coordinates as text. BoxTuning uses only a concise color legend.

Benchmark	ObjectMLLM	BoxTuning	Reduction
CLEVRER	611	61	90.1%
Perception Test	1,241	120	90.4%
STAR	761	96	87.4%
NExT-QA	1,184	77	93.5%
IntentQA	1,204	85	93.0%
Average	1,000	88	90.8%

To verify that this token reduction translates into practical computational savings, we further evaluate end-to-end runtime efficiency, model FLOPs, and GPU memory consumption in Table 3. We report measurements averaged across all five datasets on a single NVIDIA A100 80 GB GPU with a batch size of 1 under identical settings, processing each video-question sample independently without cross-question caching. Because ObjectMLLM and BoxTuning share the exact same upstream detector and tracker, BoxTuning introduces no additional perception overhead; we therefore compare the method-dependent stages: representation construction (Part I) and MLLM inference (Part II).

Constructing ObjectMLLM's coordinate text takes 3.06×10^{-5} s per sample, whereas rendering BoxTuning's bounding boxes and trajectory trails onto frames takes 0.0405 s (Part I). Despite this modest rendering overhead, the substantial reduction in text tokens greatly accelerates subsequent LLM decoding, cutting MLLM inference time from 0.5250 s to 0.3060 s (a 1.72× speedup in Part II). Consequently, BoxTuning reduces the combined representation and inference latency by 34.0%, achieving an overall 1.52× end-to-end speedup. Furthermore, BoxTuning reduces model FLOPs by 30.2% and lowers total and dynamic peak GPU memory by 9.8% and 37.8%, respectively. The 37.8% reduction in input-dependent dynamic memory directly supports larger batch sizes or longer video contexts during deployment.

Table 3: Inference latency, compute, and memory comparison averaged across all five benchmarks per video-question sample on a single NVIDIA A100 (80 GB) GPU.

Metric	ObjectMLLM	BoxTuning	Change	Rel. Change
Part I: Rep. construction latency (s)	0.00003	0.0405	+0.0405	–
Part II: MLLM inference latency (s)	0.5250	**0.3060**	−0.2190	−41.7% (1.72×)
Overall latency (Part I + II, s)	0.5250	**0.3465**	−0.1785	−34.0% (1.52×)
Model-inference FLOPs (TFLOPs)	60.3	**42.1**	−18.2	−30.2%
Peak GPU memory (GB)†	21.4	**19.3**	−2.1	−9.8%
Dynamic peak GPU memory (GB)‡	5.6	**3.5**	−2.1	−37.8%

†Includes 15.84 GB of fixed model weights. ‡Excludes fixed model weights.

4.3 Main Results: BoxTuning vs. ObjectMLLM

Table 4 presents the controlled comparison to the base model without object information and ObjectMLLM with text-coordinate bounding box encoding. We compare BoxTuning with ObjectMLLM under the same VideoLLaMA2-7B backbone and benchmark evaluation protocols, highlighting the effectiveness of visual object-track representations. The baseline trend highlights ***where object tracks help and where long coordinate prompts hurt***: ObjectMLLM improves most on CLEVRER (+9.7%), whose questions directly depend on object attributes, positions, motion, and collisions; the gains are smaller on Perception Test (+0.6%) and STAR (+0.7%), and turn negative on the reasoning-centric NExT-QA (−1.3%) and IntentQA (−1.2%), where long text-coordinate prompts can crowd out question-answer context.

Table 4: Accuracy (%) on five video QA benchmarks. All methods use VideoLLaMA2-7B with LoRA fine-tuning. Baseline numbers are from [15]. Best in **bold**. Green/red indicate change relative to the base model.

Method	CLEVRER	Perc. Test	STAR	NExT-QA	IntentQA
VideoLLaMA2 (base)	67.9	66.0	66.5	**79.8**	**76.7**
+ ObjectMLLM (text coord.)	77.6 +9.7	66.6 +0.6	67.2 +0.7	78.5 -1.3	75.5 -1.2
+ BoxTuning (ours)	**78.5** +10.6	**66.9** +0.9	**67.7** +1.2	79.7 -0.1	76.5 -0.2

BoxTuning keeps the object-level benefit while removing most of the text-coordinate burden: it outperforms ObjectMLLM on all five benchmarks while using only ∼10% of the object text tokens. On object-sensitive benchmarks, it improves over ObjectMLLM by +0.9% on CLEVRER, +0.3% on Perception Test, and +0.5% on STAR, showing that visual prompts convey spatial-temporal object cues effectively with far fewer text tokens. On reasoning-centric benchmarks, the key effect is ***avoiding context-budget damage***: BoxTuning nearly matches the base model on NExT-QA (79.7% vs. 79.8%) and IntentQA (76.5% vs. 76.7%), whereas ObjectMLLM drops by 1.3% and 1.2%.

4.4 Comparison with Existing Video MLLMs

Table 5 places BoxTuning in the broader video MLLM landscape, complementing the controlled ObjectMLLM comparison in Table 4. Within this broader comparison, BoxTuning achieves the strongest reported results across all five benchmarks.

Table 5: Comparison with existing video MLLMs. Accuracy (%) is reported.

Method	Size	CLEVR.	Perc.T.	STAR	NExT	Intent
LLaVA-NeXT-Vid. [47]	7B	38.4*†	49.3*	–	–	–
VideoLLaMA2 [9]	7B	45.6*†	51.4*	57.1*†	74.1*†	73.8*†
SeViLA [48]	3B	–	62.0	64.9	73.8	–
ViLA [49]	3B	–	–	67.1	75.6	–
ObjectMLLM [15]	7B	77.6	66.6	67.2	78.5	75.5
BoxTuning (ours)	7B	**78.5**	**66.9**	**67.7**	**79.7**	**76.5**

*Zero-shot results. †Reproduced by [15].

4.5 Ablation Studies

We ablate the text legend, trajectory trails, trail length, and rendering line width. Table 6 shows three clear patterns: *boxes alone are already strong*, *the text legend adds only a small but consistent gain*, and *trajectory trails provide the main boost*, especially on motion-heavy benchmarks. Bounding boxes alone already match ObjectMLLM closely (*e.g.*, 77.0% *vs.* 77.6% on CLEVRER), while trajectory trails bring the largest improvements (+1.2% on CLEVRER, +0.4% on Perception Test, +0.5% on STAR). On the reasoning-centric NExT-QA and IntentQA, the trail gain remains positive (+0.3%), indicating that the added motion context does not hurt reasoning. Additional analysis on trail length and rendering line width is provided in Section B.

Table 6: Component-wise ablation. Accuracy (%) is reported.

Variant	CLEVRER	Perc. Test	STAR	NExT-QA	IntentQA
Box only	77.0	66.3	66.8	79.3	76.1
Box + Legend	77.3	66.5	67.2	79.4	76.2
Box + Legend + Trail	78.5	66.9	67.7	79.7	76.5

5 Conclusion

We have designed BoxTuning, an object-aware visual prompting approach for multimodal model fine-tuning that encodes object bounding boxes and trajectory trails directly on video frames, replacing the text-coordinate paradigm with a visually grounded alternative. By shifting object information from the text modality to the visual modality, this object-aware visual prompting representation reduces text token cost by 87%-93% and eliminates the need for temporal downsampling, preserving fine-grained motion cues that text-coordinate methods are forced to discard. Experiments on five video QA benchmarks show that BoxTuning improves multimodal model fine-tuning over text-coordinate baselines on spatially oriented tasks while nearly eliminating the accuracy degradation on reasoning-centric benchmarks caused by text token overflow. Ablation studies confirm the contribution of trajectory trails for motion-sensitive tasks, validate an adaptive trail length strategy that encodes complete inter-frame trajectories, and show that the rendering hyperparameters are robust within a reasonable range. We hope these findings encourage further exploration of modality-aligned object-level visual prompts as a practical direction for multimodal model fine-tuning.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62402490 and 62572349, and the Shenzhen Basic Research Foundation under Grant QNXMC20250701101037019. The authors declare no competing interests.

References

[1] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. CLEVRER: Collision events for video representation and reasoning. In *ICLR*, 2020.

[2] Bo Wu, Shoubin Yu, Zhenfang Chen, Joshua B Tenenbaum, and Chuang Gan. STAR: A benchmark for situated reasoning in real-world videos. In *NeurIPS*, 2021.

[3] Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. NExT-QA: Next phase of question-answering to explaining temporal actions. In *CVPR*, 2021.

[4] Jiapeng Li, Ping Wei, Wenjuan Han, and Lifeng Fan. IntentQA: Context-aware video intent reasoning. In *ICCV*, 2023.

[5] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[6] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023.

[7] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. In *NeurIPS*, 2023.

[8] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

[9] Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, and Lidong Bing. VideoLLaMA 2: Advancing spatial-temporal modeling and audio understanding in video-LLMs. *arXiv preprint arXiv:2406.07476*, 2024.

[10] Hang Zhang, Xin Li, and Lidong Bing. Video-LLaMA: An instruction-tuned audio-visual language model for video understanding. In *EMNLP*, 2023.

[11] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-ChatGPT: Towards detailed video understanding via large vision and language models. In *ACL*, 2024.

[12] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. MVBench: A comprehensive multi-modal video understanding benchmark. In *CVPR*, 2024.

[13] Peng Jin, Ryuichi Takanobu, Caiwan Zhang, Xiaochun Cao, and Li Yuan. Chat-UniVi: Unified visual representation empowers large language models with image and video understanding. In *CVPR*, 2024.

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[15] Zitian Tang, Shijie Wang, Junho Cho, Jaewook Yoo, and Chen Sun. How can objects help video-language understanding? In *ICCV*, 2025.

[16] Viorica Pătrăucean, Lucas Smaira, Ankush Gupta, Adrià Recasens, Larisa Markeeva, Dylan Banarse, Skanda Koppula, Joseph Heyward, Mateusz Malinowski, Yi Yang, Carl Doersch, Tatiana Matejovicova, Yury Sulsky, Antoine Miech, Alex Frechette, Hanna Klimczak, Raphael Koster, Junlin Zhang, Stephanie Winkler, Yusuf Aytar, Simon Osindero, Dima Damen, Andrew Zisserman, and João Carreira. Perception test: A diagnostic benchmark for multimodal video models. In *NeurIPS*, 2023.

[17] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & Psychophysics*, 14(2):201–211, 1973.

[18] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[19] KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. VideoChat: Chat-centric video understanding. *Science China Information Sciences*, 68(10):200102, 2025.

[20] 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 *EMNLP*, 2024.

[21] Yanwei Li, Chengyao Wang, and Jiaya Jia. LLaMA-VID: An image is worth 2 tokens in large language models. In *ECCV*, 2024.

[22] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy visual task transfer. *TMLR*, 2025.

[23] Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Marcin Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael S Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic models: Composing zero-shot multimodal reasoning with language. In *ICLR*, 2023.

[24] Shijie Wang, Qi Zhao, Minh Quan Do, Nakul Agarwal, Kwonjoon Lee, and Chen Sun. Vamos: Versatile action models for video understanding. In *ECCV*, 2024.

[25] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal LLMs. In *CVPR*, 2024.

[26] Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Austin Wang, Rob Fergus, Yann LeCun, and Saining Xie. Cambrian-1: A fully open, vision-centric exploration of multimodal LLMs. In *NeurIPS*, 2024.

[27] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *ECCV*, 2020.

[28] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. VinVL: Revisiting visual representations in vision-language models. In *CVPR*, 2021.

[29] Junyan Li, Delin Chen, Yining Hong, Zhenfang Chen, Peihao Chen, Yikang Shen, and Chuang Gan. CoVLM: Composing visual entities and relationships in large language models via communicative decoding. In *ICLR*, 2024.

[30] Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. Shikra: Unleashing multimodal LLM's referential dialogue magic. *arXiv preprint arXiv:2306.15195*, 2023.

[31] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. In *ICLR*, 2024.

[32] Shilong Zhang, Peize Sun, Shoufa Chen, Min Xiao, Wenqi Shao, Wenwei Zhang, Yu Liu, Kai Chen, and Ping Luo. GPT4RoI: Instruction tuning large language model on region-of-interest. In *ECCV Workshops*, 2024.

[33] Yuqian Yuan, Wentong Li, Jian Liu, Dongqi Tang, Xinjie Luo, Chi Qin, Lei Zhang, and Jianke Zhu. Osprey: Pixel understanding with visual instruction tuning. In *CVPR*, 2024.

[34] Hanoona Rasheed, Muhammad Maaz, Sahal Shaji Mullappilly, Abdelrahman Shaker, Salman Khan, Hisham Cholakkal, Rao M Anwer, Eric Xing, Ming-Hsuan Yang, and Fahad S Khan. GLaMM: Pixel grounding large multimodal model. In *CVPR*, 2024.

[35] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Danny Driess, Pete Florence, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. SpatialVLM: Endowing vision-language models with spatial reasoning capabilities. In *CVPR*, 2024.

[36] Amir Mohammad Karimi Mamaghan, Samuele Papa, Karl Henrik Johansson, Stefan Bauer, and Andrea Dittadi. Exploring the effectiveness of object-centric representations in visual question answering: Comparative insights with foundation models. In *ICLR*, 2025.

[37] Aleksandar Shtedritski, Christian Rupprecht, and Andrea Vedaldi. What does CLIP know about a red circle? visual prompt engineering for VLMs. In *ICCV*, 2023.

[38] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in GPT-4V. *arXiv preprint arXiv:2310.11441*, 2023.

[39] Mu Cai, Haotian Liu, Siva Karthik Mustikovela, Gregory P Meyer, Yuning Chai, Dennis Park, and Yong Jae Lee. ViP-LLaVA: Making large multimodal models understand arbitrary visual prompts. In *CVPR*, 2024.

[40] Liangyu Zhong, Fabio Rosenthal, Joachim Sicking, Fabian Hüger, Thorsten Bagdonat, Hanno Gottschalk, and Leo Schwinn. FOCUS: Internal MLLM representations for efficient fine-grained visual question answering. In *NeurIPS*, 2025.

[41] Jiarui Zhang, Mahyar Khayatkhoei, Prateek Chhikara, and Filip Ilievski. MLLMs know where to look: Training-free perception of small visual details with multimodal LLMs. In *ICLR*, 2025.

[42] Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal LLMs. In *CVPR*, 2024.

[43] Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. YOLO-World: Real-time open-vocabulary object detection. In *CVPR*, 2024.

[44] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *ICLR*, 2025.

[45] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, 2021.

[46] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

[47] Yuanhan Zhang, Bo Li, Haotian Liu, Yong Jae Lee, Liangke Gui, Di Fu, Jiashi Feng, Ziwei Liu, and Chunyuan Li. LLaVA-NeXT: A strong zero-shot video understanding model. https://llava-vl.github.io/blog/2024-04-30-llava-next-video/, 2024.

[48] Shoubin Yu, Jaemin Cho, Prateek Yadav, and Mohit Bansal. Self-chained image-language model for video localization and question answering. In *NeurIPS*, 2023.

[49] Xijun Wang, Junbang Liang, Chun-Kai Wang, Kenan Deng, Yu Lou, Ming Lin, and Shan Yang. ViLA: Efficient video-language alignment for video question answering. In *ECCV*, 2024.

A Implementation Details

Detection and tracking. We use YOLO-World-L [43] for open-vocabulary detection with benchmark-specific category vocabularies (*e.g.*, geometric shapes for CLEVRER; people, vehicles, and everyday objects for NExT-QA). The confidence threshold is set to 0.3, and non-maximum suppression is applied at IoU 0.5. Tracking uses SAM 2 [44] with default hyperparameters. For iterative detection, new detections with IoU $>$ 0.5 against existing tracks are discarded as duplicates.

Visual prompt settings. The color palette contains K=32 maximally distinguishable colors. Bounding boxes are drawn with a line width of 3 pixels, and trajectory trails with 2 pixels. Since we uniformly sample T=16 frames per video clip, the trail length $L = \lfloor T_{\text{video}}/T \rfloor$ varies across datasets depending on the original video clip length.

Training. The STC connector is fully fine-tuned, and LoRA is applied to the query and value projection matrices of the language model with rank r=128, scaling factor α=256, and dropout 0.05. Both share the same learning rate. We train with AdamW (learning rate 2×10^{-4}, weight decay 0.01, cosine schedule with 100 warmup steps) and batch size 16 on 4×A100 80 GB GPUs.

B Additional Ablation Studies

B.1 Effect of Trajectory Length.

We vary the trail length using fixed $L \in \{10, 20, 30, 40, 60\}$ and compare against our adaptive length ($L = \lfloor T_{\text{video}}/T \rfloor$) on the motion-sensitive benchmark CLEVRER. As shown in Table 7, short trails (L=10) lead to a notable accuracy drop as they fail to capture sufficient motion context, while longer trails ($L\geq$30) yield similar performance regardless of the exact value. The adaptive strategy ($L = \lfloor T_{\text{video}}/T \rfloor$) outperforms all fixed lengths, as it aligns the trail length with the actual sampling interval, thereby encoding the complete trajectory between adjacent sampled frames without redundancy or truncation. This demonstrates that the adaptive length obtains the optimal trade-off between motion context gain and visual clutter distraction.

Table 7: Ablation study on trajectory trail length L. Accuracy (%) is reported.

Trail length L	10	20	30	40	60	Adaptive
CLEVRER	77.6	78.0	78.2	78.1	78.0	**78.5**

B.2 Effect of Line Width.

We study the sensitivity to the rendering line width for bounding boxes and trajectory trails, separately, on CLEVRER. As shown in Table 8, a box width of 3 and a trail width of 2 achieve the best accuracy. Thinner lines (w=1) are less visible at the model's input resolution, weakening the spatial signal, while wider lines ($w\geq$5) occlude the underlying frame content and degrade performance noticeably.

Table 8: Ablation study on rendering line width on CLEVRER using accuracy (%).

Width	1	2	3	5	8
Box	77.8	78.2	**78.5**	78.0	77.1
Trail	78.0	**78.5**	78.3	77.4	76.8

C Limitations and Future Directions

BoxTuning is a representation method rather than an object perception method, so it inherits the quality of the upstream detector and tracker, including missed objects, category ambiguity, and track drift. This shared dependency does not affect our controlled comparison with ObjectMLLM, which uses the same detection and tracking pipeline. In dense scenes, many boxes and trails may introduce visual clutter, motivating question-aware object selection, confidence-aware track filtering, and adaptive overlay sparsification. For longer videos, BoxTuning can further benefit from adaptive keyframe selection and compact motion summarization.

NeurIPS Paper Checklist

1. **Claims**
   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?
   Answer: [Yes]
   Justification: The abstract and introduction state the method, token-efficiency claim, evaluated benchmarks, and empirical scope. These claims are supported by the method description, token analysis, main results, and ablation studies.
   Guidelines:
   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.
2. **Limitations**
   Question: Does the paper discuss the limitations of the work performed by the authors?
   Answer: [Yes]
   Justification: Appendix C discusses the main practical boundaries of BoxTuning, including its dependence on upstream detector/tracker quality, possible visual clutter in dense scenes, and natural future directions such as question-aware object selection and adaptive overlay sparsification.
   Guidelines:
   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.
3. **Theory assumptions and proofs**
   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?
   Answer: [N/A]
   Justification: The paper does not present theoretical results, formal theorems, or proofs.
   Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sections 3, A, and 4 describe the object acquisition pipeline, visual prompt construction, model adaptation, training setup, datasets, and evaluation comparisons needed to reproduce the reported results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The training/evaluation code, rendering scripts, and reproduction instructions are publicly available at `https://github.com/zekunqian/BoxTuning`. The method relies on public benchmarks and cited open-source model components; the implementation details are described in Appendix A.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section A specifies detection, tracking, prompt rendering, LoRA settings, optimizer, learning rate, batch size, and compute hardware. Section 4.1 lists the evaluated datasets and their roles.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The experiments report point estimates for accuracy, token counts, inference latency, FLOPs, and GPU memory, but do not include error bars, confidence intervals, or repeated-run statistical tests.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).

- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**
   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?
   Answer: [No]
   Justification: Section A reports the GPU type and count used for training, and Section 4.2 reports the hardware, per-sample inference latency, FLOPs, and peak GPU memory for the efficiency comparison. However, the manuscript does not report training wall-clock time, compute costs for each training run, or the total compute used for the research project.
   Guidelines:
   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).
9. **Code of ethics**
   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?
   Answer: [Yes]
   Justification: The work studies model fine-tuning and evaluation on existing video QA benchmarks and does not introduce procedures that appear to violate the NeurIPS Code of Ethics.
   Guidelines:
   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).
10. **Broader impacts**
    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?
    Answer: [No]
    Justification: The current manuscript does not include a broader impact section. The method may improve video understanding systems, but also shares general risks associated with stronger video analysis models.
    Guidelines:
    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the

technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not introduce a new high-risk pretrained model, scraped dataset, or generative model release requiring dedicated safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The manuscript cites the original datasets, model components, and baselines used in the experiments, and we use them under their original public terms. We plan to include attribution and license notices in the released project materials, with our own materials distributed under a CC BY-NC-SA 4.0 license.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The released implementation is accompanied by documentation and reproduction instructions at `https://github.com/zekunqian/BoxTuning`. The paper does not introduce a new dataset; the method and implementation details are described in Section 3 and Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The method fine-tunes VideoLLaMA2-7B with a Mistral-7B language model component, as described in Sections 3.5 and A.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.