Appendix

Table of Contents

A. Ablation Study

This section investigates the sensitivity of different parameters in our proposed *T** framework.

A.1. Ablation on Question Grounding

Question Grounding transforms the original question into spatially queryable targets, as detailed in Section 4. In this study, we examine how increasing computational resources for Question grounding affects the efficiency and quality of the search process. Specifically, we analyze the impact of scaling Vision-Language Models (VLMs) from 7B (LLaVA) to 72B (LLaVA) parameters, as well as varying the number of initial frames, from 8 to 32.

The results, summarized in Table 7, indicate that increasing the VLM size and the number of initial frames marginally enhances both the effectiveness of the search process and downstream task performance. These results suggest that Question Grounding can be effectively achieved with modest resource allocations, offering a favorable balance between performance and resource usage.

Grounding VLM	Frames	TFLOPs	Visual F1	QA Acc
LLaVA-OneVision-7B	8	26.9	59.9	59.8
LLaVA-OneVision-72B	8	148.5	60.6	59.9
LLaVA-OneVision-7B	32	108.2	60.7	60.3

Table 7. Impact of VLM size and initial frame count on question grounding and search effectiveness on LV-HAYSTACK. Experimental settings are aligned with the baseline setup reported in Section 4 (main paper). Our results indicate that increasing the resources for question grounding results in marginal improvements in search effectiveness ($< 1\%$ gain in QA Acc.).

A.2. Ablation on Searching Algorithm

*T** aims to reduce computational overhead by partially representing the video as an $n \times n$ image grid. This approach leverages well-trained image models to systematically replace irrelevant grid cells until the target is found, based on a specified threshold θ.

Impact of Grid Size (n): We investigate how the configuration of the concatenated image grid affects both the search cost and the efficacy of the search process. Figure 5 displays the impact of varying grid sizes n (represented on the X-axis) on the average number of search steps and the corresponding average accuracy on LongVideoBench [72] XL subset.

Impact of Return Threshold θ: We examine how varying the return threshold θ impacts the efficiency and efficacy of the search process. As demonstrated in Figure 6, increasing the threshold tends to improve the accuracy of the search results but at the cost of increased computational effort. This trade-off is critical; thus, we have selected a default threshold of $\theta = 0.6$ for a balanced approach.

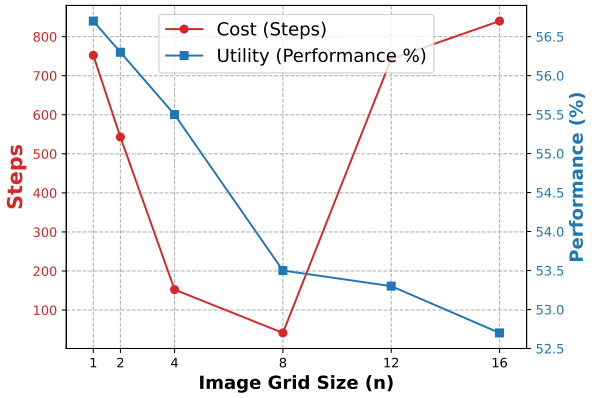

Figure 5. **Grid Size Impact on Search Performance**. The red line represents the average number of search iterations for different image grid configurations, while the blue line shows the performance on the LongVideoBench [72] XL subset using 8 frames and the LLaVA-72B as the downstream QA model.

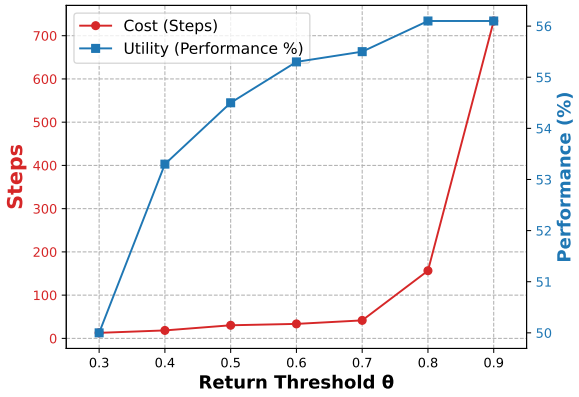

Figure 6. **Impact of Return Threshold** θ. The graph clearly illustrates the trade-off between threshold settings and search performance: lower thresholds result in quicker searches but may reduce accuracy, while higher thresholds enhance accuracy at the expense of increased search steps and computational cost.

A.3. Ablation on Searching Utility Metrics

In our primary evaluation framework, we adopt Temporal Similarity and Visual Similarity as core metrics for measuring search utility. To further investigate the robustness of our framework, we include semantic distance as an additional metric for ablation studies. Semantic distance measures the alignment of high-level features between predicted and annotated frames, as encoded in pretrained models such as `openai/clip-vit-large-patch14`.

The results on HAYSTACK-EGO4D are shown Appendix Table 8. While this metric provides insights into the semantic relevance of frames, our results reveal that its scores are closely clustered across methods, ranging from 87.9 to 89.2. Therefore, semantic distance, while informative, does not significantly discriminate between methods due to the high-level feature similarities shared across retrieved keyframes. We exclude it as an evaluation metric to maintain focus on the more distinctive temporal and visual search utility.

A.4. Correlation between Search Utility and Video Understanding

As discussed in Section 2.3, we propose multiple temporal and visual metrics to evaluate search utility. To identify the metrics most correlated with long-form video understanding, we analyzed the Pearson and Spearman correlation coefficients between utility scores and downstream task accuracy. Table 9 shows that Temporal $F1$ has the highest Pearson correlation, while Temporal Precision has the highest Spearman correlation with downstream performance, highlighting these metrics as strong predictors of effective video understanding.

B. Complexity Analysis on *T**

In this section, we analyze the time and cost complexity of the *T** search algorithm. *T** leverages adaptive temporal and spatial upsampling to efficiently collect partial information from the video and progressively determine the keyframe distribution. Based on previous observations, *T** prioritizes high-probability regions for efficient keyframe localization, similar to an A* search algorithm. By retaining only a portion of the video grid cells in each iteration (up to $1/b$ of total cells), *T** effectively performs a multi-branch search guided by a heuristic scoring function, forming a b-way search tree.

As illustrated in Figure 7, *T** is a quaternary search algorithm operating on a b-ary search tree. At each step, video frames are sampled on a grid with $b = n \times n$ cells. The top 25% of regions based on their scores are retained, and the algorithm prioritizes sampling frames around these high-scoring regions. Similar to the A* algorithm, *T** uses a heuristic scoring function to select branches, thereby shortening the search path. Ultimately, it performs a quaternary search with a heuristic function on a b-ary tree.

To simplify the discussion, assume a video of length L, containing only one frame that satisfies the target condition f_t. The grid size is $b = 2 \times 2$, and the probability that the scoring function selects the correct branch is P. The complexity analysis is conducted for the worst-case, best-case, and average-case scenarios.

Worst Case ($P \leq \frac{1}{b}$): In the worst case, when $P \leq \frac{1}{b}$, the scoring function provides no effective guidance, effectively selecting branches at random. The algorithm degrades to a linear search, sequentially checking each frame until the target frame f_t is found. The time complexity can be expressed as:

$$T_{\text{worst}} = \mathcal{O}(L), \tag{9}$$

Method	Frames↓	HAYSTACK-EGO4D								
		Temporal			Visual			Semantic		
		Precision ↑	Recall ↑	F_1 ↑	Precision ↑	Recall ↑	F_1 ↑	Precision ↑	Recall ↑	F_1 ↑
Baselines: Static Frame Sampling										
Uniform [72]	8	1.0	3.4	1.6	58.0	63.0	60.2	87.2	89.3	88.2
Uniform [72]	32	1.1	14.8	2.0	58.5	65.6	61.5	**87.3**	90.4	88.8
Baselines: Adaptive Frame Selection										
VideoAgent [68]	10.1	1.7	5.8	2.7	58.0	62.4	59.9	87.0	88.9	87.9
Retrieval-based	8	1.2	4.2	1.9	58.5	61.7	59.9	87.3	88.7	88.0
Retrieval-based	32	1.0	13.8	1.9	58.5	65.4	61.4	87.3	90.5	88.9
Ours: T^* for Zooming In Temporal Search										
Attention-based	8	2.2	7.5	3.3	58.4	62.5	60.2	87.3	89.1	88.1
Training-based	8	1.4	4.9	2.1	58.0	61.5	59.6	87.2	89.0	88.0
Detector-based	8	1.7	5.8	2.7	63.8	70.1	66.8	87.2	88.9	87.9
Detector-based	32	**1.8**	**26.3**	**3.4**	**62.9**	**76.2**	**68.9**	87.2	**91.4**	**89.2**

Table 8. Results of searching utility on LV-HAYSTACK. Best results for the 8-frame setting are underlined, and best results for the 32-frame setting are in **bold**. We include semantic metric (detailed in Appendix A.3) for ablation. Scores range closely from 87.9 to 89.2, showing limited differentiation across methods compared to temporal and visual metrics.

Metric	Pearson Correlation	Pearson p-value	Spearman Correlation	Spearman p-value
Temporal F_1	**0.901**	0.037	0.700	0.188
Temporal Precision	0.828	0.084	**0.975**	0.005
Visual F_1	0.829	0.083	0.600	0.285
Temporal Recall	0.655	0.231	0.700	0.188
Visual Recall	0.568	0.317	0.500	0.391
Visual Precision	0.327	0.591	0.100	0.873

Table 9. Pearson and Spearman correlations (with p-values) between search utility metrics and downstream task accuracy. The highest correlations are highlighted in **bold** for Temporal F_1 and Temporal Precision, suggesting they are strong predictors of effective video understanding performance.

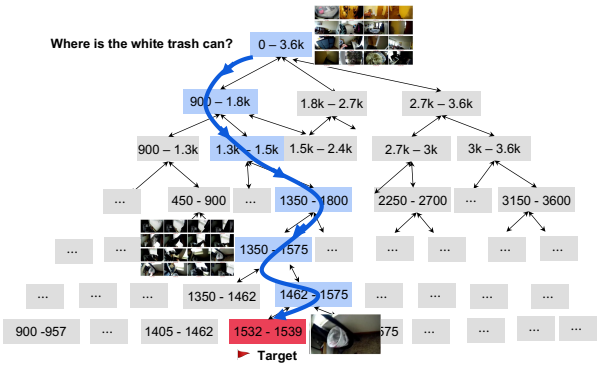

Figure 7. **Illustration of the *T** search process on a b-ary tree.** The video duration is 3.6k seconds, and the target white trash appears between 1532 and 1539 seconds. Numbers in the figure indicate the visited intervals of nodes, while the lines indicate the visited nodes and the search trajectory.

where L is the total number of video frames.

Best Case ($P = 1$): In the best case, when $P = 1$, the scoring function always selects the correct branch leading towards the target frame. The algorithm approaches the target frame directly at each step, similar to a b-ary search. The time complexity is given by:

$$T_{\text{best}} = \mathcal{O}(\log_b L), \tag{10}$$

where $b = n \times n$ is the branching factor determined by the grid size.

General Case ($\frac{1}{b} < P < 1$): In the general case, the scoring function improves branch selection accuracy based on scene correlations (e.g., a kitchen scene is more likely to contain a refrigerator than a bed). The search process can be modeled as a tree with depth:

$$m = \log_b L, \tag{11}$$

where m represents the depth of the tree. At each level, the expected number of attempts to correctly select the branch is $\frac{1}{P}$. Therefore, the total expected number of nodes visited is:

$$E[N] = m \times \frac{1}{P}, \tag{12}$$

where $E[N]$ represents the expected number of nodes visited.

The average time complexity is then given by:

$$T_{\text{avg}} = \mathcal{O}\left(\frac{\log_b L}{P}\right), \tag{13}$$

where the efficiency of the algorithm is inversely proportional to the scoring function's accuracy P. A higher P

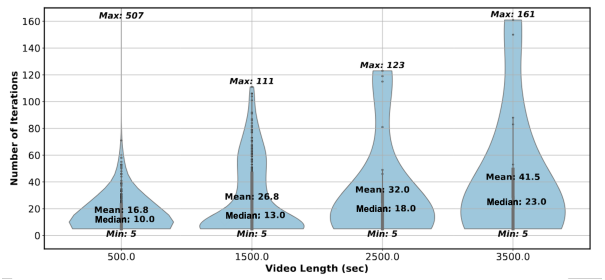

Figure 8. **Empirical results on search complexity of *T**.** We show the number of iterations of T^* on the LongVideoBench dataset, dividing videos with lengths between 0 and 4000 seconds into four groups. The figure shows the generally required intensity of iterative search as the length of videos vary.

value reduces the exploration of incorrect branches, significantly improving efficiency.

Figure 8 shows the behavior of *T** across various video lengths, presenting empirical statistics of search steps from our LV-HAYSTACK dataset. The statistical results demonstrate that for videos ranging from 100 to 3600 seconds, the average number of search steps is categorized into four equidistant groups. The average number of steps required by *T** increases gradually with video length. Notably, for videos longer than 3000 seconds, the maximum number of search steps recorded is 161, the minimum is 5 steps, and the average is 41.5 steps to complete the search. These variations are attributable to the differing intrinsic correlations within the content of each video, which informs the heuristic-based object detection process.

C. Detail Analysis on LongVideoBench

Table 10 highlights the impact of incorporating *T** as a frame selection module on QA accuracy across different video lengths in the LongVideoBench dataset.

Overall Effectiveness of *T:** Incorporating *T** consistently improves QA accuracy for both GPT4o and LLaVA-OneVision-72B across all video lengths, demonstrating the effectiveness of the keyframe selection module in enhancing video understanding. For instance, in XLong videos (15-60 minutes), GPT4o's accuracy improves from 47.1 to 51.55 $\pm$ 0.35, while LLaVA-OneVision-72B's accuracy increases from 53.7 to 55.25 $\pm$ 0.25.

Impact of Video Length: The improvements are more pronounced for longer videos (XLong and Long), where information density is higher, suggesting that *T** is particularly effective in identifying and prioritizing relevant frames in complex scenarios. For shorter videos (Medium and Short), while the improvements are relatively smaller, *T** still contributes to stabilizing performance across multiple runs.

Effect of Model Size: Larger models, such as LLaVA-OneVision-72B, benefit slightly more from *T** compared to smaller models like GPT4o, especially for longer videos. This indicates that larger models can better utilize the high-quality keyframes selected by *T**.

In conclusion, *T** consistently enhances QA accuracy across various video lengths, with greater impact on longer videos and larger models. These results demonstrate the potential of *T** in improving video-language understanding and reasoning in long-form videos.

D. Implementation Details

D.1. Implementation of Training-free *T**

Question Grounding: For Question Grounding, we primarily use the LLaVA-OneVision 7B model, applying it to 8 uniformly sampled frames. The prompt adheres to the official release guidelines, and the specific template used is listed in Table 12.

Iterative Temporal Search: The default configuration for the image grid size is $b = 8 \times 8$. We set the return threshold θ at 0.6 for object-based and training-based scoring functions as trade-off in Figure 6. For the attention-based method, we typically use the sum of the attention scores from the target object in the last layer of each frame. This approach was chosen because using smaller models or shallower layers resulted in performance below the baseline. Additionally, the process terminates after three iterations to manage the high computational costs associated with using the 72B model.

Downstream Question Answering: For downstream task evaluations, we experiment with the most prominent state-of-the-art (SOTA) models, both open- and closed-source, namely GPT4o and LLaVA-OneVision 72B. For GPT4o, we use the official API. For LLaVA, we employ the official code. The prompt template for this testing is listed in Table 13.

D.2. Implementation of Trainable T^*

In our framework, both object-based and attention-based *T** methods score each cell within the image grid and guide zooming based on straightforward rules. The training-based *T** approach, however, renders this iterative search process learnable.

To learn the search policy, we employ a reinforcement learning approach. Its action space, reward function, and loss function are described as follows:

Action Space To implement trainable scoring, we replace YOLO's detection header with a single-layer Multilayer Perceptron (MLP). This MLP maps high-level detection features into a score, indicating the likelihood that a specific area within a frame contains the visual context necessary to answer the question. For an image grid with $b = n \times n$ cells, each cell is assigned a predicted score, represented as

Model and Size	#Frame	LongVideoBench			
		Video Length			
		XLong 15-60min	Long 2-10min	Medium 15-60s	Short 8-15s
GPT4o	8	47.1	49.4	67.3	69.7
GPT4o + *T**	8	**51.6 ± 1.4**	**51.7 ± 1.7**	**72.9 ± 1.2**	**70.2 ± 0.2**
LLaVA-OneVision-72B	8	53.7	57.4	74.1	73.0
LLaVA-OneVision-72B + *T**	8	**55.3 ± 1.3**	**63.5 ± 1.2**	**76.6 ± 1.3**	**73.7 ± 0.2**
GPT4o	32	50.5	57.3	73.5	71.4
GPT4o + *T**	32	**53.3 ± 1.2**	**59.2 ± 1.2**	**74.3 ± 0.0**	**71.4 ± 0.0**
LLaVA-OneVision-72B	32	56.5	61.6	77.4	74.3
LLaVA-OneVision-72B + *T**	32	**62.6 ± 1.2**	**63.9 ± 1.2**	**79.3 ± 0.0**	**74.6 ± 0.0**

Table 10. Detailed downstream task evaluation results for T^* as an additional frame selection module for VLMs on LongVideoBench. The metric is QA accuracy (%). We run *T** two times and report the average accuracy and standard deviation (±).

$\mathcal{C} \in \mathbb{R}^{n \times n}$, which serves as the action space:

$$\mathcal{C}, \mathcal{B} \leftarrow \text{ScoreFunction}(G, \mathcal{T}) = \text{MLP}(\text{YOLO}(G, \mathcal{T})). \tag{14}$$

Reward Function To evaluate the quality of selected frames, we define a reward function based on their effectiveness in answering the question. Using the predicted scores $\mathcal{C} \in \mathbb{R}^{n \times n}$, we select K frames and pass them to a VLM for question answering. The reward is calculated as the difference in accuracy between selected frames and a uniform baseline:

$$\text{reward} = \text{VLM}(K_{\text{selected}}, Q) - \text{VLM}(K_{\text{uniform}}, Q), \tag{15}$$

where $\text{VLM}(K_{\text{uniform}}, Q)$ represents the baseline accuracy using uniform sampled frames, and $\text{VLM}(K_{\text{selected}}, Q)$ represents the accuracy using frames sampled based on the predicted relevance scores $\mathcal{C}$.

Loss Function To optimize the trainable scoring mechanism, we employ a reinforcement learning-inspired approach using Monte Carlo estimation. We sample K frames M times based on the predicted scores $\mathcal{C}$. These sampled frames are passed into a Visual Language Model (e.g., `llava-OneVision-7B`) to answer the question. The average accuracy across these attempts is used as the reward signal. The loss function for training is defined as:

$$\text{loss} = \sum_{i=1}^{M} \left(\text{reward}_i \times \text{CrossEntropy}(\mathcal{C}, \mathcal{C}_{\text{i}})\right), \tag{16}$$

where $\mathcal{C}_{\text{i}}$ represents binary labels for the i-th Monte Carlo sample, with selected cells for K as 1 and others as 0, and reward_i is the reward for the i-th sample, according to Eqn. 15.

This formulation ensures that the model is reinforced to predict scores $\mathcal{C}$ aligning with the sampling labels $\mathcal{C}_{\text{i}}$ when the reward is positive. Conversely, when the reward is negative, the model adjusts its predictions to reduce the similarity between $\mathcal{C}$ and $\mathcal{C}_{\text{i}}$, penalizing incorrect sampling patterns.

Training and Inference The search policy is trained on existing short videos and tested on unseen long videos. During the inference phase, *T** uses the output of the trained YOLO model as a heuristic score. All training and inference operations are carried out on a cluster of 8*H800 Nvidia GPUs.

We observed that models trained on the NExT-QA dataset can also effectively identify better frames for long video tasks, such as those in LongVideoBench. This suggests that unifying video representation as an $n \times n$ grid—whether for short or long videos—enables a consistent approach. Furthermore, identifying better frames should be considered a foundational task, facilitating cross-dataset generalization.

D.3. Implementation of the baseline VideoAgent

VideoAgent [68], the state-of-the-art temporal search baseline, leverages LLM-based video keyframe selection to optimize VLM input. It generates captions to describe video content and incrementally aggregates relevant information for question answering. We adapt the original public code to make it runnable for long video haystack setting and benchmark. While the original VideoAgent implementation uses BLIP-Large [30] for caption generation, which is significantly larger than our YOLO-based approach [8] (110M parameters), we adapted the implementation to use CLIP-1B [51] for fair comparison. Specifically, we employed `clip-vit-large-patch14` and `blip-image-captioning-large`* for our experiments.

*Available at CLIP and BLIP

D.4. Video QA Implementation

For downstream video question answering experiments, we uniformly use LLaVA-OneVision 72B [28] as our QA model. This open-source VLM excels in processing multimodal inputs, integrating text, image, and video analysis. We selected this model for its ability to handle arbitrary video frames and its demonstrated superior performance across diverse VQA benchmarks.

D.5. Implementation of Different Search Strategies

The core of *T** leverages a well-trained open-world YOLO model for rapid object verification based on question questions. We evaluate *T**'s effectiveness through three distinct search strategies:

- **Retrieval-based Search**: Utilizes the YOLO model [8] to exhaustively scan and rank video frames based on target object detection confidence. The top 8 frames (by default) are selected as final outputs.
- **Zooming In Search**: Implements a hierarchical approach starting with an $N \times N$ image grid matrix at low fps and resolution. The search progressively refines both fps and resolution in promising segments identified through object detection and visual cues, ultimately returning 8 frames.
- **Trainable Search**: Adapts frame processing dynamically through YOLO model fine-tuning. Beginning with uniform sampling on an $N \times N$ image grid, it predicts correlation coefficients to guide subsequent grid sampling distributions. This process iterates three times by default, maintaining an 8-frame output. The model is trained on NExT-QA dataset and evaluated across multiple datasets.

E. Data Annotation Details

To curate data for our benchmark, we repurpose established long-video understanding datasets that focused on question answering. To represent different visual scenes, we cover both egocentric and allocentric views [18, 72], resulting in two diverse subsets HAYSTACK-EGO4D and HAYSTACK-LVBENCH. This approach not only allows direct comparison with past results but also saves time and resources for extensive data curation. We ask crowd-source annotators to identify keyframes and answers for HAYSTACK-EGO4D, while directly borrowing the keyframes and answers annotated from LONGVIDEOBENCH.

E.1. HAYSTACK-LVBENCH

To curate data for HAYSTACK-LVBENCH, we utilize the frame positions from LONGVIDEOBENCH [72] as ground-truth human-recommended frame indices. This decision is based on LONGVIDEOBENCH's annotation process, where annotators were required to propose questions based on given frame positions. Since LONGVIDEOBENCH only retained frame position records in the validation set, we exclusively constructed HAYSTACK-LVBENCH using data from the validation set. Furthermore, considering our focus on long-video understanding, we only included cases from the 3600-second duration group. To ensure broader applicability, we also excluded cases that referenced subtitles in their questions. As a result, we obtained a final set of 114 videos and 342 question pairs, none of our selected cases relied on text subtitles. You can use our script *'Longvideobench2LVHaystackFormat.py'* to obtain HAYSTACK-LVBENCH and check more detailed statistics.

E.2. HAYSTACK-EGO4D

To create HAYSTACK-EGO4D, we conducted data annotation on a dataset comprising 1,324 video clips, which were extracted from the original 988 videos containing a total of 15,092 questions. The video clips were pre-segmented by the Ego4D dataset to ensure that each clip contained sufficient context for answering the associated questions. This segmentation also simplified the annotation process, as shorter videos allowed annotators to better comprehend the content and efficiently identify keyframes.

The detailed instructions and interface (see Figure 9) provided to the annotators are described in the next section, *Data Annotation Interface*. Annotators were instructed to watch each video clip and answer a predefined set of questions. For every question, they were required to identify and mark several keyframes within the video that were relevant to their answers. Subsequently, they answered the questions based on these selected frames.

To assist annotators, we provided a recommended time interval to help them quickly identify relevant frames. However, we also instructed them to watch the entire video before answering the questions, as the recommended intervals identified by the Ego4D dataset may not always be accurate. Watching the full video is crucial for ensuring logical correctness in keyframe identification. For example, some questions involve events such as *"What is the second time that somebody does something?"*, requiring the annotator to identify both the first occurrence and the second occurrence to answer accurately.

In cases where a video did not provide sufficient clues to answer certain questions (potentially due to mistakes in the original dataset), annotators were instructed to respond with *"Not able to answer the question"* and provide corresponding reasons (e.g., *"The object does not occur in this video"*).

Since the annotators were not native English speakers, we utilized the `googletrans` package to translate the original questions and the interface into their native language (Chinese). Similarly, their answers were translated back into English for consistency.

To ensure the quality of the annotations, we randomly sampled 100 question-answer pairs from the annotated dataset. Only 2 obvious mistakes were identified in the

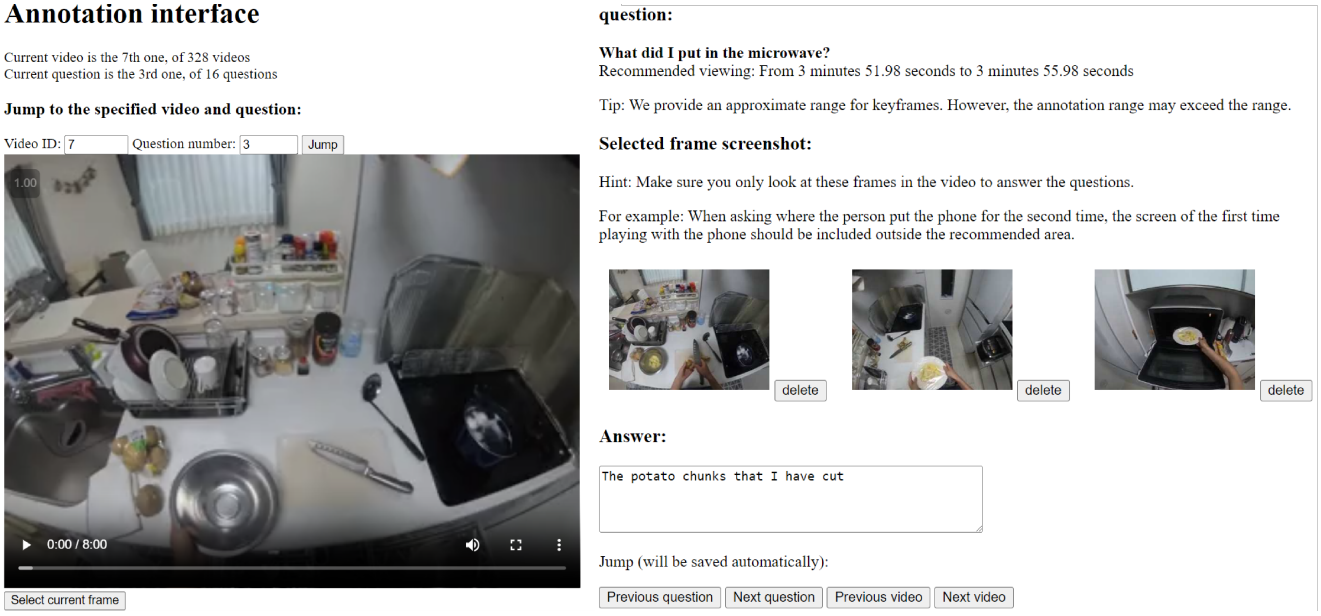

Figure 9. **Annotation Interface for Videos.** This interface allows annotators to answer questions based on video clips by keyframe annotations. Annotators can navigate to specific videos and questions using the provided controls (Video / Question ID). The current question is displayed with the recommended time range identified by the Ego4D dataset. Annotators can select key video frames and delete or modify annotations. A text box is available for entering answers based on observed video content.

Figure 10. The visualization of frame selection results demonstrates the effectiveness of our approach compared to baseline methods. Our method consistently identifies more relevant and temporally diverse keyframes, capturing important frames that directly address the question. In contrast to baseline approaches which may select redundant or less informative frames, our strategy achieves better coverage of key events while maintaining temporal coherence across the video sequence.

sample, indicating a decent overall annotation quality. Consequently, we retained this annotation set for further analysis.

F. Data Annotation Interface

Our data annotation interface facilitates annotators to provide precise answers to questions based on video clips. Key features include:

- **Video Navigation:** Annotators can jump to a specific video and question using the provided controls (e.g., Video ID and Question Number).
- **Question Display:** The current question is displayed prominently, along with the recommended time range for viewing relevant keyframes in the video.
- **Frame Selection:** Annotators can select specific video

frames for reference and delete or adjust their selection as needed to support their answer.

- **Answer Input:** A dedicated text box allows annotators to provide their responses based on the observed content in the selected video frames.
- **Navigation Controls:** Quick navigation buttons enable moving between videos or questions efficiently.

This tool ensures accurate, contextual, and streamlined annotations for video content analysis tasks.

G. Qualitative Analysis

Figure 11 compares uniform sampling with T* sampling for long-format video understanding. The task involves identifying a "metal cylinder" in an hour-long video. Uniform sampling, which selects 8 frames randomly, misses key frames containing the metal cylinder.

In contrast, T* sampling focuses on semantically relevant frames, successfully capturing those featuring the metal cylinder. This highlights T* sampling's ability to prioritize critical visual information

H. Prompt Design

In this section, we include the prompts designed for environment representation, focusing on question grounding and question answering tasks.

H.1. Prompt for Question Grounding

The following is the prompt used by our system for question grounding:

Prompt Template for Question Grounding

```
<system prompt>
Here is a video:
<image>
<image>
<image>
...
Here is a question about the video:
Question: <Question>
Options: <Options>

When answering this question about the video:
1. What key objects to locate the answer?
   - List potential key objects (short sentences,
     separated by commas).
2. What cue objects might be near the key objects
     and might appear in the scenes?
   - List potential cue objects (short sentences,
     separated by commas).

Please provide your answer in two lines, directly
     listing the key and cue objects, separated by
     commas.
```

Figure 12. The template of a question grounding prompt *T**. `<system prompt>` is the default system instruction from LLaVA and the `<image>` are PIL.Image objects for each frame and other text elements are strings.

This prompt is designed to generate a representation of the environment that facilitates the grounding of queries in a structured, object-centric manner.

H.2. Prompt for Question Answering

The following prompt is used to answer questions based on the embodied environment representation. This design is adapted from LLaVA:

Prompt Template for Question Answering

```
<system prompt>
Select the best answer to the following multiple-
     choice question based on the video.
<image>
<image>
<image>
...
Question: <Question>
Options: <Options>

Answer with the option letter from the given choices
      directly.
```

Figure 13. The template of a question answering prompt.

H.3. Prompt for Distractor Generation

Prompt Template for Distractor Options

```
<system prompt>
You are an expert in creating challenging multiple-
     choice questions.
For the question: <question> with the correct answer
      <answer> generate 4 plausible but incorrect
     distractors that are closely related to the
     correct answer in context, category, or
     characteristics, making the question more
     challenging.
Ensure that the distractors could reasonably seem
     correct to someone who is unsure of the answer.
The output format should be:
"1. \"<Distractor 1>\"",
"2. \"<Distractor 2>\"",
"3. \"<Distractor 3>\"",
"4. \"<Distractor 4>\"".
```

Broad Impact

The *T** framework provides an efficient keyframe extraction solution compatible with any model or task, with applications in video summarization, healthcare training, entertainment indexing, and real-time surveillance. Its computational efficiency reduces energy consumption, aligning with sustainability goals. Additionally, the LV-HAYSTACK bench-

Figure 11. **Comparison of uniform sampling and T* sampling for long-format video understanding.** In this example, the task involves identifying a "metal cylinder" in an hour-long video. Uniform sampling fails to include relevant frames, as it randomly selects 8 frames across the video. In contrast, T* sampling dynamically selects frames containing the metal cylinder, providing the necessary visual context for effective understanding.

mark advances standardized evaluation practices, encouraging innovation in long-form video understanding.

Our proposed dataset is also applicable to foundation models that process entire videos. With our LV-HAYSTACK dataset which consists of both training and test sets, we aim to show that keyframe supervision can act as a guiding mechanism, enabling models to first identify the most relevant keyframes from video, then use them to produce a contextually grounded answer. This approach can be more structured, efficient, and effective than directly predicting an answer from a long-form video.