Spatio-Temporal Grounding of Large Language Models from Perception Streams

Anonymous Author(s)
Affiliation
Address
email

Abstract

Embodied-AI agents must reason about how objects move and interact in 3-D space over time, yet existing smaller frontier Large Language Models (LLMs) still mis-handle fine-grained spatial relations, metric distances, and temporal orderings. We introduce the general framework Formally Explainable Spatio-Temporal Scenes (FESTS) that injects verifiable spatio-temporal supervision into an LLM by compiling natural-language queries into Spatial Regular Expression (SpRE) — a language combining regular expression syntax with $S4_u$ spatial logic and extended here with universal and existential quantification. The pipeline matches each SpRE against any structured video log and exports aligned (query, frames, match, explanation) tuples, enabling unlimited training data without manual labels. Training a 3-billion-parameter model on 27k such tuples boosts frame-level F1 from 48.5% to 87.5%, matching GPT-4.1 on complex spatio-temporal reasoning while remaining two orders of magnitude smaller, and, hence, enabling spatio-temporal intelligence for Video LLM.

1 Introduction

The ability to comprehend and reason about how a dynamic, three-dimensional world evolves over time is fundamental to embodied AI—spanning household robotics, autonomous driving, and assistive manipulation. To train and evaluate such systems we also need tooling that can query and annotate spatio-temporal events in video perception logs. LLMs and Visual Language Models (VLMs)[1] already show promise as task-and-motion planners [25, 19] and low-cost annotators [14]. Yet a growing body of work demonstrates that frontier models remain brittle: they mis-judge fine-grained spatial relations [24, 27, 13, 23], lose track of temporal dynamics [11], and struggle when both aspects matter simultaneously [8, 9]. For instance, VLMs often confuse relative object ordering, fail to distinguish identical instances, and cannot reason about metric distance—shortcomings that translate directly into failure modes.

In this paper, we present FESTS, a framework that injects rich, verifiable spatio-temporal supervision into an LLM, enabling it to answer – and explain – complex video queries. Our key idea is to leverage SpREs [2], a language that fuses regular-expression syntax with $S4_u$ spatial logic, to generate large numbers of self-verifiable queries and corresponding ground-truth matches. These queries can express properties such as "*find all frames in which a car and a bus start at least 10 m apart and come within 1 m of each other within 20 frames*," which go beyond multiple-choice QA, and naturally scale to 2-D or 3-D data. Crucially, we extend SpREs to support universal and existential

[1]"Visual" refers to any (potentially multi-modal) model that accepts an image or sequence of images as input.

Submitted to Conference on Neural Information Processing Systems (NeurIPS 2025).

quantification over objects to track entities across time and encode behaviors like "*every pedestrian is at least 1m away from the truck.*"

Recently, Li et al. [8] showed that Video LLMs [30, 20] – models coupling a video encoder with a language decoder – can improve reasoning skills through purely textual fine-tuning. Their evidence suggests that temporal-reasoning bottlenecks lie in the LLM component rather than the video encoder, implying that stronger textual supervision can improve reasoning. Our framework capitalises on this insight: by generating arbitrarily many SpREs-grounded (query, frames, match, explanation) tuples from any perception dataset, we fine-tune the LLM component to reason about both temporal orderings and spatial relations.

In more detail, given textual video object annotation data which must include object classes and bounding box information, and which may include unique object identifiers, pixel depth information, or other attributes of interest, e.g., color. Our goal is to fine tune an LLM to be able to reason about arbitrary spatio-temporal patterns which can be encoded with SpREs. We present a framework which automates query generation and data annotation with the goal of producing any desired size training dataset. It is important to highlight three benefits of our framework. First, our framework can be utilized on both real data and artificially generated data. Second, and most importantly, with a given video data set or perception data, we can generate an arbitrary number of spatio-temporal queries for training and fine tuning. Third, our framework can also produce natural language explanations on why a pattern was matched on the annotated dataset. This additional information can be fed as part of the training process, or even be used in a chain-of-thought spatial reasoning framework as in [21]. To our knowledge, no dataset exists that couples complex queries to spatio-temporal reasoning capabilities of models. Virtually all the prior works on spatio or spatio-temporal fine tuning use multiple choice question and answering for fine-tuning with much simpler spatio-temporal properties.

Using our benchmark dataset, we show that with just 27k training examples (each paired with explanations), we boosted a 3-billion-parameter model to be competitive against a state-of-the-art GPT-4.1 model on our training and evaluation dataset. This establishes that our framework has the potential to enhance Video LLMs [30, 20] with new spatio-temporal reasoning capabilities since we enable some more complex patterns than [21]. Although our fine-tuned model consistently achieves substantial improvements across varied query complexities and frame lengths, there remains strategic room for further enhancement, particularly in existential queries that involve extended object tracking across frames, where GPT-4.1 currently maintains an advantage.

Contributions Our paper makes the following contributions:

1. **Dataset:** We release FESTS benchmark dataset, the first automatically-annotated video corpus whose labels are derived from verifiable spatio-temporal queries rather than crowd-sourced labels.
2. **End-to-end pipeline:** FESTS ships code to (i) synthesize diverse SpRE queries, (ii) match them against structured perception logs, and (iii) export aligned (`query, frames, match, explanation`) tuples for training or evaluation.
3. **Pattern matching language extension:** We add existential and universal quantifiers to SpRE, enabling persistent object tracking
4. **Empirical improvements:** Using the resulting "Query→Explain" supervision, we fine-tune a 3B-parameter LLM (Qwen-2.5-Coder-Instruct) from 48.5 % to 87.5 frame-level F1, keeping competitive with GPT-4.1 on complex spatio-temporal reasoning with orders of magnitude fewer parameters.

Collectively, these results show that spatio-temporal fine-tuning, powered by logically-grounded synthetic supervision, can endow LLM with reasoning skills well beyond what multiple-choice QA alone affords.

2 Related Work

Spatial reasoning with LLM and VLM. A series of recent papers show that frontier models still lack spatial reasoning capabilities and propose various model enhancements. Chen et al.'s SpatialVLM [6], Cai et al.'s SpatialBot [5], Cheng et al.'s SpatialRGPT [7], Ma et al.'s 3D-aware SpatialLLM [22], and Zhang et al.'s COMFORT [31] all attempt to patch these gaps with geometric priors or object-centered prompts. BLINK [13] proposes "visual" commonsense benchmark problems that humans can answer within seconds, e.g., multi-view reasoning, depth estimation, and reflectance estimation. Yet the underlying benchmarks remain limited to local or static relations. FESTS subsumes this scope by compiling natural-language prompts into Quantified-Spatial Regular Expression (q-SpRE) that permit metric constraints, set operations, and universal / existential quantification.

Spatial benchmarks for LLM and VLM. The works [27] and [24] propose benchmarks that can evaluate whether frontier models poses spatial intelligence which is natural among animals. GRASP [27] demonstrates that cutting edge LLM cannot produce plans given a spatial reasoning problem. SPACE [24] exposes failures of LLM and VLM to produce a mental map of the environment when traversing it. It also demonstrates that foundation models cannot perform smaller-scale reasoning about object shapes and layouts. FESTS has orthogonal goals and evaluation criteria to GRASP and SPACE. However, it would be interesting to evaluate if FESTS can also improve spatial intelligence in frontier models.

Video-LLM benchmarks and temporal reasoning. Temporal understanding has progressed from early captioning datasets to full video-LLM challenges. Ju et al. [16] prompt VLMs for temporal localization and reveal poor clip-level accuracy. Li et al.[8] demonstrate that purely textual fine-tuning lifts ordering performance and temporal localization. V-STaR benchmark [9] assesses spatio-temporal reasoning ability in answering questions in the context of "when", "where", and "what". Mementos [28] stresses sequence reasoning over image sets, while PaLM-E [12] proposes and evaluates embodied language models with additional sensing modalities. The work in [30] shows that by simply expanding context windows improves performance in performance on long video question-answering benchmarks. NSVS-TL [11] shows that current VLM fail at long-term reasoning across frames and propose a temporal logic based framework for temporal reasoning. Nearly all approaches (besides NSVS-TL) produce benchmarks based on multiple-choice labels or short captions and question-answering. Even though the aforementioned approaches focus on temporal relations across frames, they do not really consider spatial reasoning at the same fidelity as FESTS. q-SpRE instead produces verifiable `(query, frames, match, explanation)` tuples that jointly stress spatial and temporal reasoning, and its generator can wrap any perception log—including the clips used by other benchmarks.

3 Preliminaries

This section reviews the Spatio-Temporal Regular Expression Matching (STREM) framework [2], highlights its limitations, and presents our contributions to it.

3.1 The Spatio-Temporal Regular Expression Matching Framework

The STREM framework [2] is designed to match queries over perception data streams. The queries are expressed as SpREs, which combine Regular Expressions (REs) with the spatial logic $S4_u$ [18], enabling patterns to capture both temporal and spatial relationships among objects. The matching procedure uses a formal-methods approach based on Deterministic Finite Automata (DFA), which determines whether a perception stream satisfies a given query.

3.1.1 Limitations

In the current variation, there are several limitations to the STREM framework that do not support the ability to perform more complex temporal queries.

In the current version, SpRE queries such as, a simple "Find all frames where the same pedestrian is present for five frames", or more complicated, "Find all frames where the same pedestrian overlaps with any car or bus for five frames" are not possible. Furthermore, reasoning over all kinds of objects at a specific point in time across multiple points of time is not possible and thus queries such as, "Find all frames where all cars are more than 500 units away from any pedestrian for three frames" do not have any inherit support. These limitations enforce a per-frame reasoning query to

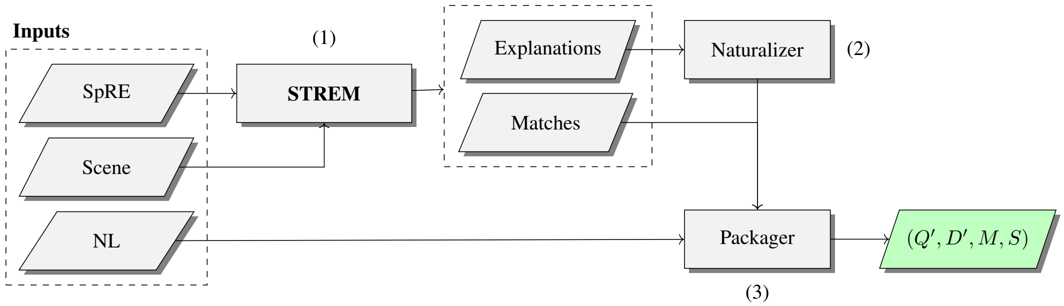

Figure 1: The FESTS framework begins with (1) which processes the SpRE and perception stream inputs to produce the two formal method-based results; (2) processes the explanation to improve readability for LLMs; and (3) packages this into a distributable data formats.

be formed by the user and thus does not enable a wide range of multi-frame temporal reasoning expressions that would otherwise strengthen the capabilities of the querying language overall.

3.2 Quantification Support

To support operations of quantification-based queries to extend the capabilities of LLMs, we first adjust the syntax of the SpRE grammar to include RE-level quantifiers. The modified SpRE syntax is shown in Eq. 1 below.

$$Q \coloneqq \phi \mid Q_1\, Q_2 \mid Q_1 | Q_2 \mid Q^* \mid \exists x.Q \mid \forall x.Q \tag{1}$$

where $\exists$ and $\forall$ correspond to the new existential and universal quantifier and $\forall$ corresponds to the new universal quantifier introduced. The syntactic definitions of the other operators may be reviewed in [2]. For a formal review of the semantics, see Sect. 10;

To support the semantics of these quantification operators, we integrate a new matching algorithm alongside additional components to support the semantics of the existential and universal quantifiers within SpRE queries.

4 Formally Explainable Spatio-Temporal Scenes

The FESTS framework (see Fig. 1) accepts as input a data stream D of downstream perception-based data such as object annotations; examples of pre-existing datasets containing such information include Woven Perception [17] or nuScenes [4]. As output, the FESTS data pipeline returns a perception stream of with each entry organized as follows:

$$(Q', D', M, S) \tag{2}$$

where Q' is the Natural Language (NL) variant of the SpRE query Q, $D' = (F_i, F_{i+1}, \ldots, F_j) \subseteq D$ is the sampled data stream, M is the set of matches from STREM, and S is the set of NL explanations linearized from the set of explanations E.

Let us consider the following NL query written for an Autonomous Vehicle (AV) system affixed with image-based sensors and a downstream object detector:

> *Find all frames where the bounding box of the same car intersects with a bounding box of a bus for two frames.*

From this query, the goal is to identify frames from the perception stream that match the properties outlined. This query is composed of both spatial properties such as *intersection* as well as temporal properties such as *sequences*. However, while current LLMs such as GPT-4o [15] initially showcase positive performance on single property-based queries, queries containing a mix of both spatial and temporal elements begin to demonstrate failures. These failures consist of hallucinations in the perception stream, incorrect ranges, and reduced accuracy over longer traces as concluded in [10]. To improve upon these limitations, we utilize fine-tuning of LLMs through a formal methods-based approach to the data generated for training and fine-tuning of the models.

If the query above is processed through our framework, the resulting output would be as follows:


```
{
  "input": {
    "input": "You identify video scenes matching a natural language query
    ↪  using frame-level object detections.\nInput XML
    ↪  structure:\n<root>\n  <query>Natural language scene
    ↪  description.</query>\n  <data>\n
    ↪  frame,identifier,label,score,xmin,ymin,xmax,ymax\n
    ↪  0,AB,pedestrian,1.0,1254,603,269,101\n
    ↪  1,AC,car,0.9,1300,600,280,110\n    ...\n  </data>\n</root>\nOutput
    ↪  format:\n-Matched frames as lists of consecutive indices in
    ↪  <result> tags.\n-Brief explanation inside <reasoning> tags.\n-If no
    ↪  match, output: <result>[]</result><reasoning></reasoning>\nExample
    ↪  output:\n<result>[[1,2,3],[7,8]]</result>\n<reasoning>Frames 1-3
    ↪  and 7-8 matched due to presence of pedestrians crossing the
    ↪  road.</reasoning>\nNo extra text outside <result> and <reasoning>
    ↪  tags.---\n<root>\n\t<query>Find all frames where the bounding box
    ↪  of the same car intersects with a bounding box of a bus for two
    ↪  frames.</query>\n<data>\nindex,identifier,class,xmin,ymin,xmax,yma
    ↪  x\n23,aa,bus,232,538,307,571\n23,ba,car,323,504,518,643\n23,ca,car
    ↪  ,558,508,741,672\n23,da,car,488,517,570,579\n23,ea,car,893,517,101
    ↪  1,554\n23,fa,car,285,525,366,578\n23,ga,car,480,521,537,562\n23,ha
    ↪  ,car,265,526,407,604\n24,ga,car,485,521,540,561\n24,ia,car,39,528,
    ↪  258,623\n24,da,car,497,517,574,576\n24,ca,car,564,507,736,662\n24,
    ↪  ha,car,281,526,415,600\n24,aa,bus,217,538,293,570\n24,ba,car,343,5
    ↪  05,523,636\n24,fa,car,293,525,366,576\n24,ea,car,893,516,1011,554\
    ↪  n</data>\n</root>\n"
  },
  "output": [
    [
      23,
      24
    ]
  ],
  "explanations": [
    "From index 23 to 24, area of the bounding box of a car with id ba
    ↪  overlaps with a bus."
  ]
}
```


5 Experiments

To evaluate the effectiveness of our approach, we fine-tune an LLM, Qwen2.5-3B-Instruct [29], on the outputs of our framework from an AV perception dataset, Woven Perception [17]. In the following sections, the dataset composition, fine-tuning procedure, evaluation metrics, and results are presented.

5.1 Dataset Composition

To fine-tune an LLM on the outputs of our framework, a perception stream source is required. The Woven Perception [17] dataset was chosen for its comprehensive selection of perception streams and high-quality, hand-labeled object annotations. This dataset is comprised of 180 different scenes with each scene containing a stream of 126 frames from 7 different monocular camera sensors, which provides 1.2K+ perception streams to process with our framework.

To generate the data for fine-tuning, the perception streams were sampled at incremental frame lengths of 1, 2, 4, 6, 8, 10, 12, 14, and 16 to gradually increase the difficulty for the LLM. For each sample, our framework joins the satisfaction result and explanation from the STREM framework with the corresponding NL query and perception stream data for 15 templated queries. This procedures yields 27K+ outputs as the inputs to fine-tune the LLM on.

5.1.1 Query Types

The queries we fine-tune the model on can be grouped into five distinct categories. These categories and considerations of each are outlined below:

1. **Sequence**: A query containing multiple temporally adjacent events.
2. **Spatial**: A query that contains operations such as intersection of bounding boxes.
3. **Temporal**: A query that contains eventual events.
4. **Metric**: A query that contains measurement-based operations.
5. **Existential**: A query that contains reasoning on the same or all objects over time.

5.2 Models and Fine-Tuning Configurations

The fine-tuning was performed entirely on the LLM, Qwen2.5-3B-Instruct [29]. This model was selected for several reasons: (1) publicly and readily available, (2) small parameter size, (3) ideal for task completion and fine-tuning, and (4) size of context-length. The model was fine-tuned under the following two training configurations:

- C1. **Supervised Fine-Tuning**: The model was trained exclusively on the *query* and *match* outputs of our framework, with no *explanation* field. The Parameter-Efficient Fine-Tuning (PEFT) using the Low Rank Adaptation (LoRA) method was applied to the attention and MLP layers with a rank of 16, scaling of 32, and a dropout of 0.05; trained for 5 epochs with an effective batch size of 60; optimized with AdamW (8-bit) with a learning rate of 1×10^{-5} and cosine scheduling.
- C2. **Supervised Fine-Tuning with Reinforcement Learning**: The model was pre-trained from the C1 configuration. The Reinforcement Learning (RL) with Proximal Policy Optimization (PPO) used where the PPO used a custom hierarchical-based reward function (see Sect. 5.3); trained for 1 PPO epoch with 4 optimization epochs per PPO batch; optimized with AdamW (8-bit) with a learning rate of 1×10^{-6}, effective batch size of 4, a KL divergence coefficient of 0.05, and upper bound of 512 tokens for rollouts.

In addition, the fine-tuned models were compared against the GPT-4.1[2] [1] model representing the state-of-the-art and the Qwen2.5-Coder-3B-Instruct model [29] representing the baseline.

5.3 Evaluation Metrics

To evaluate the model during fine-tuning, we developed two methods distinct for each fine-tuning configuration in Sect. 5.2.

For the C1 configuration, the causal language modeling objective is optimized using cross-entropy loss, minimizing differences between the predicted and ground-truth token probabilities such that all tokens except the results are masked.

For the C2 configuration, a hierarchical-based reward function is used. This reward function evaluates several properties including: (1) structural validity such as XML formatting; (2) match accuracy with mAP IoU and exact match; and (3) reasoning fidelity, which assesses semantic similarity to ground-truth explanations using a sentence transformer from [26] and numerical IoU of referenced frames. The penalties of the reward function account for excessive response length, spurious text outside delimited tags, and invalid formats.

[2]This model was accessed and used for evaluation on 05/01/2025.

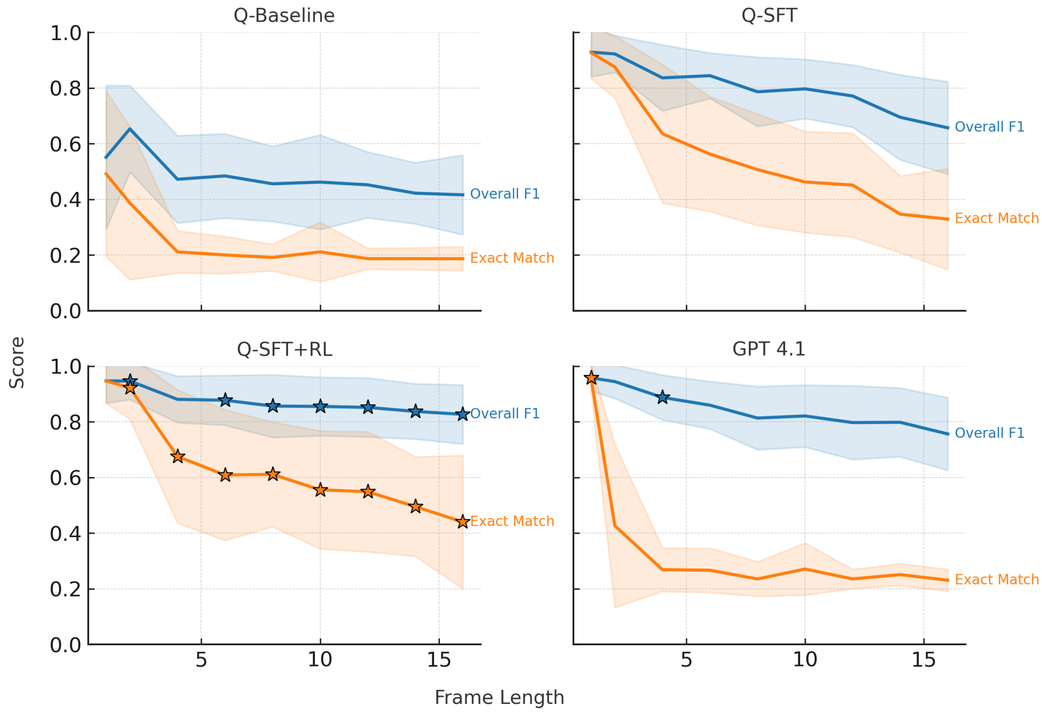

Figure 2: Average Performance across frame lengths. Blue = Overall F1, orange = Exact Match; shaded σ. Starred points denote the best-performing model for each frame length.

While the reasoning fidelity guides the RL training, it noted that primary performance metrics in Sect. 5.4 focus on the accuracy of the predicted frames.

5.4 Results

Main Findings. Table 1 summarizes key results. Adding explanations yields large improvements. Supervised fine-tuning on query–answer pairs (Q-SFT) improves overall Frame F1 from 48.5% to 80.4%. Reinforcement learning on top of those explanations (Q-SFT+RL) pushes Frame F1 to 87.5%, just above GPT-4.1 at 84.8%. The jump is primarily driven by recall (+28 points during SFT) and then by precision (+2.2 points during RL). Exact-match rises from 25.0% (Baseline) to 56.6% (Q-SFT) and 64.5% (Q-SFT+RL).

Impact of Query Length. Fig. 2 shows that gains scale with query length. For 16-frame inputs Frame F1 climbs from 65.7 % (Q-SFT) to 82.7 % (Q-SFT+RL), a +17-point jump. Exact-match improves by +39.5 points over baseline and by +29.5 points over GPT-4.1 (64.5 % vs. 35.0 %). Most of the improvement comes from SFT. RL improves a further 7.9 points.

Table 1: Performance comparison across models and target lengths. Metrics: Frame F1 ($F1_f$), Exact Match (EM), and Segment F1 ($F1_s$). Best results per column are in **bold**.

	4			8			12			16			Overall		
Model	$F1_f$	EM	$F1_s$	$F1_f$	EM	$F1_s$	$F1_f$	EM	$F1_s$	$F1_f$	EM	$F1_s$	$F1_f$	EM	$F1_s$
Q-Baseline	0.472	0.211	0.325	0.456	0.191	0.258	0.452	0.187	0.231	0.416	0.187	0.215	0.485	0.250	0.326
Q-SFT	0.836	0.636	0.644	0.786	0.507	0.582	0.771	0.451	0.565	0.657	0.329	0.454	0.804	0.566	0.636
Q-SFT+RL	0.881	**0.676**	**0.694**	**0.856**	**0.611**	**0.686**	**0.852**	**0.549**	**0.659**	**0.827**	**0.440**	**0.629**	**0.875**	**0.645**	**0.723**
GPT-4.1	**0.888**	0.269	0.640	0.813	0.236	0.528	0.797	0.236	0.497	0.756	0.231	0.447	0.848	0.350	0.610

Per-Query Type Performance. Table 2 confirms the pattern across five query types. SFT improves F_1 by +0.35 (Sequence), +0.35 (Spatial), +0.31 (Temporal), +0.31 (Metric), and +0.28 (Existential). RL improves recall above 0.90 for Sequence, Metric, and Existential and raises F_1 to 0.90+ for Sequence, Metric, and Temporal. Spatial and Existential stay at 0.82–0.83 F_1. Exact-match is 0.821 for Sequence and 0.706 for Metric but does not perform as well at 0.531 for Existential queries.

Summary. The query-explanation–RL pipeline delivers consistent, order-of-magnitude improvements with minimal extra annotation, suggesting strong potential for transfer to other video-language tasks governed by symbolic logic.

Our analysis refrains from comparing against *reasoning* models, as this introduces an additional learning signal and thus another source of variance, complicating attribution in experimental studies. Moreover, reasoning models exacerbate the inherent textual-context limitations of LMMs, necessitating truncated analyses or aggressive token pruning to accommodate input data within memory constraints.

Table 2: Precision (P), Recall (R), F_1, and Exact-Match (EM) for each query type. Values are averaged across 8-, 12-, and 16-frame lengths.

Query type	Base				Stage-I: SFT				Stage-II: SFT + RL			
	P	R	F_1	EM	P	R	F_1	EM	P	R	F_1	EM
Sequence	0.875	0.668	0.570	0.206	0.931	0.961	0.923	0.715	**0.971**	**0.978**	**0.962**	**0.821**
Spatial	0.785	0.557	0.415	0.190	0.825	0.853	0.761	0.502	**0.845**	**0.905**	**0.819**	**0.552**
Temporal	0.863	0.597	0.504	0.348	0.883	0.857	0.810	0.553	**0.910**	**0.902**	**0.866**	**0.615**
Metric	0.872	0.551	0.457	0.200	0.903	0.830	0.769	0.597	**0.917**	**0.960**	**0.903**	**0.706**
Existential	**0.875**	0.574	0.480	0.306	0.817	0.861	0.757	0.463	0.826	**0.937**	**0.828**	**0.531**
Average	0.854	0.589	0.485	0.250	0.872	0.872	0.804	0.566	**0.894**	**0.936**	**0.876**	**0.645**

6 Limitations

The current set of limitations of this work includes: (1) missing component to automate the translation between NL-based queries into SpRE counterparts; (2) the perception stream strictly requires pre-labeled data; (3) accuracy of the results depends on the quality of the sourced labels; (4) manual curation and creation of the queries are required; and (5) evaluation on a single model.

7 Conclusions

In this work, we developed FESTS, a benchmark dataset leveraging verifiable, logically-grounded queries for automated annotation and explainability, substantially advancing video-language model training without reliance on crowd-sourced labels. Through systematic fine-tuning of the Qwen-3B model using our FESTS dataset, we achieved a notable improvement in frame-level F1 performance, from 48.5% to 87.5%, achieving similar performance with GPT-4.1. Despite these results, the generalization capabilities of our model still trail behind state-of-the-art models such as GPT-4.1, particularly on Existential and Spatial queries.

For future work, the following items are of immediate interest: (1) develop methods for synthetic generation of queries and subsequent perception streams to improve generalizability, (2) perform additional comparisons against a wider range of model configurations, and (3) incorporate other spatial information such as point clouds or depth maps.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Jacob Anderson, Georgios Fainekos, Bardh Hoxha, Hideki Okamoto, and Danil Prokhorov. Pattern matching for perception streams. In *International Conference on Runtime Verification*, pages 251–270. Springer, 2023.

[3] David Basin, Felix Klaedtke, Samuel Müller, and Eugen Zălinescu. Monitoring metric first-order temporal properties. *Journal of the ACM (JACM)*, 62(2):1–45, 2015.

[4] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[5] Wenxiao Cai, Iaroslav Ponomarenko, Jianhao Yuan, Xiaoqi Li, Wankou Yang, Hao Dong, and Bo Zhao. Spatialbot: Precise spatial understanding with vision language models. *arXiv preprint arXiv:2406.13642*, 2024.

[6] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465, 2024.

[7] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. Spatialrgpt: Grounded spatial reasoning in vision language models. *arXiv preprint arXiv:2406.01584*, 2024.

[8] Zixu Cheng, Jian Hu, Ziquan Liu, Chenyang Si, Wei Li, and Shaogang Gong. Lei li and yuanxin liu and linli yao and peiyuan zhang and chenxin an and lean wang and xu sun and lingpeng kong and qi liu. In *International Conference on Learning Representations (ICLR)*, 2025.

[9] Zixu Cheng, Jian Hu, Ziquan Liu, Chenyang Si, Wei Li, and Shaogang Gong. V-star: Benchmarking video-llms on video spatio-temporal reasoning. *arXiv preprint arXiv:2503.11495*, 2025.

[10] Minkyu Choi, Harsh Goel, Mohammad Omama, Yunhao Yang, Sahil Shah, and Sandeep Chinchali. Towards neuro-symbolic video understanding. In *European Conference on Computer Vision*, pages 220–236. Springer, 2024.

[11] Minkyu Choi, Harsh Goel, Mohammad Omama, Yunhao Yang, Sahil Shah, and Sandeep Chinchali. Towards neuro-symbolic video understanding. In *Computer Vision (ECCV 2024)*, pages 220–236. Springer, 2025.

[12] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. In *International Conference on Machine Learning*, pages 8469–8488. PMLR, 2023.

[13] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision*, pages 148–166. Springer, 2024.

[14] Noriaki Hirose, Catherine Glossop, Ajay Sridhar, Dhruv Shah, Oier Mees, and Sergey Levine. Lelan: Learning a language-conditioned navigation policy from in-the-wild videos. *arXiv preprint arXiv:2410.03603*, 2024.

[15] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[16] Chen Ju, Tengda Han, Kunhao Zheng, Ya Zhang, and Weidi Xie. Prompting visual-language models for efficient video understanding. In *European Conference on Computer Vision*, pages 105–124. Springer, 2022.

[17] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Woven planet perception dataset 2020, 2019.

[18] Roman Kontchakov, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. Spatial logic+ temporal logic=? *Handbook of spatial logics*, pages 497–564, 2007.

[19] Teyun Kwon, Norman Di Palo, and Edward Johns. Language models as zero-shot trajectory generators. *IEEE Robotics and Automation Letters*, 2024.

[20] Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023.

[21] Yuecheng Liu, Dafeng Chi, Shiguang Wu, Zhanguang Zhang, Yaochen Hu, Lingfeng Zhang, Yingxue Zhang, Shuang Wu, Tongtong Cao, Guowei Huang, et al. Spatialcot: Advancing spatial reasoning through coordinate alignment and chain-of-thought for embodied task planning. *arXiv preprint arXiv:2501.10074*, 2025.

[22] Wufei Ma, Luoxin Ye, Nessa McWeeney, Celso M. de Melo, Alan L. Yuille, and Jieneng Chen. Spatialllm: A compound 3d-informed design towards spatially-intelligent large multimodal models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[23] Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, et al. Openeqa: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16488–16498, 2024.

[24] Santhosh Kumar Ramakrishnan, Erik Wijmans, Philipp Kraehenbuehl, and Vladlen Koltun. Does spatial cognition emerge in frontier models? *arXiv preprint arXiv:2410.06468*, 2024.

[25] Zachary Ravichandran, Varun Murali, Mariliza Tzes, George J Pappas, and Vijay Kumar. Spine: Online semantic planning for missions with incomplete natural language specifications in unstructured environments. *arXiv preprint arXiv:2410.03035*, 2024.

[26] Sentence-Transformers. all-mpnet-base-v2. https://huggingface.co/sentence-transformers/all-mpnet-base-v2, 2021. Apache-2.0 license.

[27] Zhisheng Tang and Mayank Kejriwal. Grasp: A grid-based benchmark for evaluating commonsense spatial reasoning. *arXiv preprint arXiv:2407.01892*, 2024.

[28] Xiyao Wang, Yuhang Zhou, Xiaoyu Liu, Hongjin Lu, Yuancheng Xu, Feihong He, Jaehong Yoon, Taixi Lu, Gedas Bertasius, Mohit Bansal, et al. Mementos: A comprehensive benchmark for multimodal large language model reasoning over image sequences. *arXiv preprint arXiv:2401.10529*, 2024.

[29] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[30] Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. In *International Conference on Learning Representations (ICLR)*, 2025.

[31] Zheyuan Zhang, Fengyuan Hu, Jayjun Lee, Freda Shi, Parisa Kordjamshidi, Joyce Chai, and Ziqiao Ma. Do vision-language models represent space and how? evaluating spatial frame of reference under ambiguities. In *International Conference on Learning Representations (ICLR)*, 2025.

8 Reinforcement Learning Fine-Tuning Algorithm

The model is fine-tuned using a simplified Proximal Policy Optimization (PPO) variant. The objective is to optimize the policy π_θ to maximize an expected custom reward, regularized by the KL divergence from a frozen reference policy π_{ref} (initialized from the SFT model).

Algorithm 1 Abstract PPO for Structured Output Generation

Input: Initial policy parameters θ_0; dataset $\mathcal{D}$ of $(\mathbf{x}, \text{ans}_{\text{gt}}, \text{exp}_{\text{gt}})$ pairs; hyperparameters.
Initialize: Policy π_θ with $\theta \leftarrow \theta_0$; reference π_{ref} with θ_0 (frozen); Optimizer for θ.

for epoch $e = 1$ to E_{PPO} **do**
  for each batch $\{(\mathbf{x}_j, \text{ans}_{\text{gt}_j}, \text{exp}_{\text{gt}_j})\}_{j=1}^{B}$ from $\mathcal{D}$ **do**
    Sample responses (trajectories) $\mathbf{y}_j \sim \pi_\theta(\cdot|\mathbf{x}_j)$ for each $\mathbf{x}_j$.
    Compute rewards $R_j = \text{RewardFunction}(\mathbf{x}_j, \mathbf{y}_j, \text{ans}_{\text{gt}_j}, \text{exp}_{\text{gt}_j})$. (Described below)

    For each trajectory $\mathbf{y}_j$:
      Estimate $D_{\text{KL},j} := \mathbb{E}_{t\sim\text{Unif}(1,|\mathbf{y}_j|)}[\log \pi_{\text{ref}}(y_{j,t}|\mathbf{x}_j, y_{j,<t}) - \log \pi_\theta(y_{j,t}|\mathbf{x}_j, y_{j,<t})]$.

    Define batch loss $\mathcal{L}(\theta) := -\frac{1}{B}\sum_j R_j + \beta_{\text{KL}} \cdot \frac{1}{B}\sum_j D_{\text{KL},j}$.
    Update θ using $\nabla_\theta \mathcal{L}(\theta)$ (e.g., AdamW, with gradient accumulation).
Output: Fine-tuned policy parameters θ.

Reward Function

The $\text{RewardFunction}(\mathbf{x}, \mathbf{y}, \text{ans}_{\text{gt}}, \text{exp}_{\text{gt}})$ is a composite function. It first evaluates the generated response $\mathbf{y}$ against the prompt $\mathbf{x}$ and ground truth data $(\text{ans}_{\text{gt}}, \text{exp}_{\text{gt}})$ to produce a set of detailed metrics $\mathcal{M}$. These metrics assess:

- **Format Validity**: Correctness of required XML-like tags (e.g., <reasoning>, <result>) and structural validity of content within tags.
- **Answer Accuracy**: Structural Exact Match (EM) and mean Average Precision (mAP_IoU) of the <result> content against ans_{gt}.
- **Reasoning Faithfulness**: Evaluates the text content $\mathbf{y}_{\text{reasoning}}$ within the <reasoning> tag against exp_{gt}. This score is a product of:
  1. Numerical IoU: Intersection over Union of key numerical entities (e.g., frame numbers) mentioned in $\mathbf{y}_{\text{reasoning}}$ versus those in exp_{gt}.
  2. Semantic Relevance: A binary value indicating if $\mathbf{y}_{\text{reasoning}}$ is semantically similar to exp_{gt}. This similarity is determined by:
     (a) Encoding both texts into vector embeddings using the Sentence Transformer model sentence-transformers/all-mpnet-base-v2.
     (b) Calculating the cosine similarity between these embeddings.
     (c) Checking if this similarity score meets or exceeds a predefined threshold τ_{sim} (set to 0.5 in our experiments).
- **Penalties**: For excessive length, spurious content outside defined tags, and invalid answer formats.

These metrics $\mathcal{M}$ are then combined into a scalar reward R using a hierarchical weighting scheme. Good formatting acts as a primary gate, scaling the rewards obtained from answer accuracy and reasoning quality. This encourages the model to first produce well-structured outputs before optimizing for content accuracy and faithfulness.

9 Queries

Table 3: Query-set taxonomy used for training and evaluation. Each category is assigned to a structural class (Type). `SPre` is the symbolic pattern and *NL description* the natural-language equivalent.

<table>
<tr><th>Category</th><th>Type</th><th>SPre pattern</th><th>NL description</th></tr>
<tr><td>A1</td><td>Sequence</td><td><code>[[:pedestrian:]]{1,5}</code></td><td>Find all frames where a pedestrian is present for 1–5 frames.</td></tr>
<tr><td>A2</td><td>Sequence</td><td><code>[[:car:]]{1,8}</code></td><td>Find all frames where a car is present for 1–8 frames.</td></tr>
<tr><td>A3</td><td>Sequence</td><td><code>[[:car:] & [:pedestrian:]]{1,7}</code></td><td>Find all frames where a car and pedestrian are present for ≤ 7 frames.</td></tr>
<tr><td>B1</td><td>Spatial</td><td><code>[NE([:pedestrian:] & [:car:])]{1,3}</code></td><td>Find frames where a pedestrian's box overlaps a car for 1–3 frames.</td></tr>
<tr><td>B2</td><td>Spatial</td><td><code>[NE([:bus:] & [:car:])]{1,2}</code></td><td>Find frames where a bus overlaps a car for exactly 1–2 frames.</td></tr>
<tr><td>B3</td><td>Spatial</td><td><code>[NE([:pedestrian:] & ([:truck:] | [:bus:] | [:car:]))]{1,5}</code></td><td>Find 1–5 frames where a pedestrian overlaps a truck, bus, or car.</td></tr>
<tr><td>C1</td><td>Temporal</td><td><code>[[:car:]][[:pedestrian:]][[:car:]]</code></td><td>Find car–pedestrian–car triplets in consecutive frames.</td></tr>
<tr><td>C2</td><td>Temporal</td><td><code>[[:pedestrian:]]{3}[[:car:]]</code></td><td>Find a 3-frame pedestrian run followed by a car.</td></tr>
<tr><td>C3</td><td>Temporal</td><td><code>[[:car:]][NE([:bus:] & [:car:])]{2,4}</code></td><td>Find a car followed by a bus–car overlap lasting 2–4 frames.</td></tr>
<tr><td>D1</td><td>Metric</td><td><code>[@dist([:bus:], [:car:]) < 500.0]{1,3}</code></td><td>Find 1–3 frames where bus–car distance < 500 px.</td></tr>
<tr><td>D2</td><td>Metric</td><td><code>[@area([:car:]) > 5000.0]{1,2}</code></td><td>Find 1–2 frames where a car's area > 5000 px.</td></tr>
<tr><td>D3</td><td>Metric</td><td><code>[@x([:car:]) < @x([:bus:])]{1,4}</code></td><td>Find 1–4 frames where a car is left of a bus.</td></tr>
<tr><td>E1</td><td>Existential</td><td><code>E(v := [:pedestrian:])([v]{2,5})</code></td><td>Find repeated sightings (2–5 frames) of the same pedestrian.</td></tr>
<tr><td>E2</td><td>Existential</td><td><code>E(v := [:car:])([NE(v & ([:car:] | [:pedestrian:]))]{1,4})</code></td><td>Find a tracked car that intersects a car or pedestrian for 1–4 frames.</td></tr>
<tr><td>E3</td><td>Existential</td><td><code>E(v := [:car:])([NE(v & [:bus:])]{1,3})</code></td><td>Find frames where the same car intersects a bus for 1–3 frames.</td></tr>
</table>

- **S4 (Set-based)**: enables set operations such as intersection, union, complement, interior, and closure over spatial regions.
- **Metric (Real number-based)**: arithmetic operations, like addition, subtraction, multiplication, division, exponentiation, and common unary/binary functions.
- **S4u (Boolean-based)**: conjunctions, disjunctions, relational operators, esp. characterized by spatial constraints.
- **Regular Expressions (Symbol-based)**: characterized by concatenation, alternation, and Kleene-star.

10 Regular–Expression Semantics with Quantification and Storage

In this section, we present our extension of regular expressions to regular expressions with data variable quantification operations. For simplicity in the presentation, we do not present the spatial reasoning syntax and semantics, but rather we focus on the semantics of the quantifiers. We refer the reader to [2] for the definitions and implementation of the spatial operators in SpREs.

Alphabet, variables, and storage. Fix a finite data alphabet Σ and a finite set of variables $\mathcal{V}$. A *storage function* is a (partial) mapping

$$\sigma : \mathcal{V} \rightharpoonup \Sigma,$$

which we update with the usual functional override $\sigma[x \mapsto d]$ for $x \in \mathcal{V}$ and $d \in \Sigma$.

Extended syntax.

$$r ::= \emptyset \mid \varepsilon \mid a \quad (a \in \Sigma) \mid x \quad (x \in \mathcal{V}) \mid r_1 \,|\, r_2 \mid r_1 \cdot r_2 \mid r^* \mid \exists x.\, r \mid \forall x.\, r.$$

Intuitively, the atomic expression x matches the *current symbol* iff that symbol equals the value stored for x.

Indexed satisfaction with storage. For a finite trace $w = w_0 \dots w_{n-1}$ and a storage σ, we write

$$(w, i, j, \sigma) \;\models\; r$$

meaning "the sub-trace $w[i : j)$ satisfies r under environment σ." The rules below generalise the regular expression matching to qunatified regular expressions. The generalization is needed in order to define how values are stored and retrieved in the variables.

Base cases

$$\begin{aligned}
(w, i, j, \sigma) &\models \emptyset &&: \quad \text{never true;}\\
(w, i, j, \sigma) &\models \varepsilon &&: \quad i = j;\\
(w, i, j, \sigma) &\models a &&: \quad i + 1 = j \wedge w_i = a;\\
(w, i, j, \sigma) &\models x &&: \quad i + 1 = j \wedge \sigma(x){\downarrow} \wedge w_i = \sigma(x),
\end{aligned}$$

where "$\sigma(x){\downarrow}$" means that x is defined in σ.

Boolean and Kleene cases (the environment is threaded unchanged):

$$\begin{aligned}
(w, i, j, \sigma) \models r_1 \,|\, r_2 &\iff (w, i, j, \sigma) \models r_1 \vee (w, i, j, \sigma) \models r_2;\\
(w, i, k, \sigma) \models r_1 \cdot r_2 &\iff \exists\, j \,\big(i \le j \le k \wedge (w, i, j, \sigma) \models r_1 \wedge (w, j, k, \sigma) \models r_2\big);\\
(w, i, j, \sigma) \models r^* &\iff \exists m \ge 0,\ i = k_0 \le k_1 \le \ldots \le k_m = j \text{ s.t. } \forall \ell < m, (w, k_\ell, k_{\ell+1}, \sigma) \models r.
\end{aligned}$$

Quantifiers (bind a fresh value and store it for the continuation):

$$\begin{aligned}
(w, i, j, \sigma) \models \exists x.\, r &\iff \exists d \in \Sigma.\ (w, i, j, \sigma[x \mapsto d]) \models r;\\
(w, i, j, \sigma) \models \forall x.\, r &\iff \forall d \in \Sigma.\ (w, i, j, \sigma[x \mapsto d]) \models r.
\end{aligned}$$

Word-level semantics (closed expressions). For a *closed* expression (no free variables) we can start with the empty storage:

$$w \models r \iff (w, 0, |w|, \varnothing) \models r.$$

Remark. The storage function σ behaves exactly like the valuation environment used in first-order temporal logics [3]: it *stores* every value chosen by a quantifier so that subsequent occurrences of the corresponding variable x can test equality with the previously stored data symbol.

11 User Guide

This guide provides the basics to build, install, and run the FESTS framework for generating spatio-temporal data based on the Woven Perception dataset.

11.1 The Formally Explainable Spatio-Temporal Scenes Framework

This framework can be found under the `fests/` folder provided in the supplementary materials. To use this tool, you must first have completed the following pre-requisites:

1. Install the STREM tool (v0.1.1)
2. Install the Python interpreter (v3.10)
3. Install the FESTS library (v0.1.0)

Note: For the following subsections below (Sects. 11.1.1 to 11.1.3), all commands are ran from the root project directory as the working directory—the directory where these instructions reside.

11.1.1 Installing the STREM Tool

The STREM tool is used to generate the formally verifiable match results from the perception data provided to be appended in the creation of the FESTS dataset.

Pre-Requisites The STREM tool is a Rust-based command-line interface tool which relies on the Rust compiler and toolchain to build and install, correctly. Therefore, installation of the toolchain is required, this can be done by running the following command:

```
$ curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

Installation To install the STREM tool to your system to be ran as a command-line tool, run the following command from the root directory, accordingly:

```
$ cargo install --path strem/
```

11.1.2 Installing the Python Interpreter

In this demonstration, a Linux-based environment is assumed for proper installation of the correct Python interpreter. For this guide, we will assume an Ubuntu-based system. To install the correct Python version, run the following command from the root directory, accordingly:

```
$ sudo apt update && \
sudo apt install software-properties-common -y

$ sudo add-apt-repository ppa:deadsnakes/ppa && \
sudo apt update

$ sudo apt install python3.10 python3.10-venv python3.10-dev
```

11.1.3 Install the FESTS Library

To install the FESTS library for FESTS-based dataset generation, run the following command from the root directory, accordingly:

```
$ pip install fests/
```

To verify successful installation, run the following commands:

```
$ strem --version && \
python --version && \
python fests/scripts/process.py --help
```

11.1.4 Running the FESTS Framework Tool

To run the tool and generate a uniformly sampled FESTS dataset, run the following command from the root directory, accordingly:

```
$ python fests/scripts/process.py \
--output="output/" \
--recursive \
--jobs=32 \
--context="fests/data/" \
fests/data/woven/
```

After running the command above, the results will be saved to the `output/data/` folder. From this, the format of such a file may look as follows:

```
{
  "input": {
    "input": "You identify video scenes matching a natural language query
    ↪  using frame-level object detections.\nInput XML
    ↪  structure:\n<root>\n  <query>Natural language scene
    ↪  description.</query>\n  <data>\n
    ↪  frame,identifier,label,score,xmin,ymin,xmax,ymax\n
    ↪  0,AB,pedestrian,1.0,1254,603,269,101\n
    ↪  1,AC,car,0.9,1300,600,280,110\n    ...\n  </data>\n</root>\nOutput
    ↪  format:\n-Matched frames as lists of consecutive indices in
    ↪  <result> tags.\n-Brief explanation inside <reasoning> tags.\n-If no
    ↪  match, output: <result>[]</result><reasoning></reasoning>\nExample
    ↪  output:\n<result>[[1,2,3],[7,8]]</result>\n<reasoning>Frames 1-3
    ↪  and 7-8 matched due to presence of pedestrians crossing the
    ↪  road.</reasoning>\nNo extra text outside <result> and <reasoning>
    ↪  tags.---\n<root>\n\t<query>Find all instances where the area of a
    ↪  car is greater than 5000 pixels for one or two frames.</query>\n<d⌋
    ↪  ata>\nindex,identifier,class,xmin,ymin,xmax,ymax\n23,aa,car,232,53⌋
    ↪  8,307,571\n23,ba,car,323,504,518,643\n23,ca,car,558,508,741,672\n2⌋
    ↪  3,da,car,488,517,570,579\n23,ea,car,893,517,1011,554\n23,fa,car,28⌋
    ↪  5,525,366,578\n23,ga,car,480,521,537,562\n23,ha,car,265,526,407,60⌋
    ↪  4\n24,ga,car,485,521,540,561\n24,ia,car,39,528,258,623\n24,da,car,⌋
    ↪  497,517,574,576\n24,ca,car,564,507,736,662\n24,ha,car,281,526,415,⌋
    ↪  600\n24,aa,car,217,538,293,570\n24,ba,car,343,505,523,636\n24,fa,c⌋
    ↪  ar,293,525,366,576\n24,ea,car,893,516,1011,554\n</data>\n</root>\n"
  },
  "output": [
    [
      23,
      24
    ]
  ],
  "explanations": [
    "From index 23 to 24, area of the bounding box of a car is greater than
    ↪  5000."
  ]
}
```

In addition, you may view the `output/stats.json` to view a set of overall and per-query dataset statistics such as the elapsed time, seed, number of files processed, percentage of files that have non-empty matches, etc.

11.1.5 Important

The table below highlights some important resources provided in the supplementary materials and their purposes for inspection:

Resource	**Details**
`strem/`	The STREM tool source code with modifications
`fests/`	The FESTS framework source code
`fests/data/woven/`	A sample from the Woven Perception dataset
`fests/data/prompt.txt`	The LLM prompt used for fine-tuning the LLM model
`fests/data/queries.json`	The set of SpRE and NL queries fine-tuned with
`dataset/fests/`	A sample FESTS dataset generated from the Woven dataset

Table 4: A set of important resources and locations.