A Vehicle-Centric Surveillance Benchmark for Testing Spatial Perception in Multimodal LLMs

Nora Mendoza Kareaga

June 30, 2025

Abstract

This work introduces a benchmark dataset derived from the VIRAT Ground surveillance videos for evaluating Multimodal Large Language Models (MLLMs). From 12h of video footage, a vehicle-centred subset is isolated, keeping such events and creating image–question pairs about motion direction, turning and U-turns, a total of 1,568 prompts. Multiple visual configurations add or remove temporal context, cropping, blur and trajectory overlays. Two open-source 2B parameter MLLMs (Qwen2-VL-2B, InternVL-2B) are evaluated on the full set. Humans solve all tasks easily, while models had trouble, this analysis shows the deficiencies in basic orientation and temporal sequencing tasks, crutial characteristic of video surveillance models.

Keywords: Visual-Language Models, Video-Surveillance, Vehicle-Activity Recognition, Multimodal Language Model, Spatio-Temporal Reasoning, Benchmark Dataset, VIRAT

1 Introduction

Surveillance systems are components of modern infrastructure that significantly improve public safety, law enforcement, and urban traffic management. As urban environments increasingly deploy large camera networks, they continuously generate large amounts of video data. Manually analyzing this massive volume of data is labor intensive, costly, and prone to human errors, potentially leading to delayed or inaccurate responses. Consequently, there is a growing need for automated solutions capable of rapidly and reliably interpreting video content.

Traditional Video Analytics (VA) solutions have made significant progress in efficiently processing large datasets by effectively detecting and tracking objects such as vehicles and pedestrians and annotating relevant attributes, including object location and trajectories. However, these systems heavily rely on predefined categories and rules, limiting their ability to understand complex contextual information inherent in real-world surveillance scenarios.

Large Language Models (LLMs), on the other hand, have recently revolutionized natural language processing due to their capability to generate coherent human-like text, summarize extensive textual information, and perform logical reasoning. However, the high-dimensional and voluminous nature of surveillance data poses significant computational and memory challenges for standard LLMs.

Multimodal Large Language Models (MLLMs) overcome these limitations by integrating computer vision and natural language understanding, enabling simultaneous interpretation of visual and textual data. In this report, we specifically focus on models that jointly handle images and text. The project is about a benchmakr dataset to evaluate MLLM performance and capabilities, emphasizing their effectiveness in extracting spatial and temporal information from surveillance video data, specifically using the VIRAT Video Analytics dataset.

1.1 Objectives

1. Develop a specialized benchmark dataset, derived from VIRAT, to assess the ability of MLLMs to interpret surveillance scenarios.
2. Design straightforward image–question pairs that test a model's ability to reason about direction, turning and basic action order.
3. Evaluate the capabilities of of state-of-the-art MLLMs (Qwen2-VL and InternVL) in surveillance tasks, particularly their spatial and temporal reasoning abilities.

1.2 State Of The Art

Multimodal Large Language Models (MLLMs) are AI systems that jointly process visual data and natural language,

• Contact E-mail: 1631736@uab.cat
• Supervised by: Jordi Gonzàlez (Computer Science)
• Academic Year 2024/25

integrating information from both images and textual descriptions. Recent MLLMs such as Qwen2-VL [1], LLaVA [2], InternVL [3] and MiniCPM [4] have shown able capabilities, including complex image description and multimodal reasoning.

These models have enabled new applications such as image captioning, visual question answering, multimodal reasoning, and interactive assistants. However, despite significant achievements, several critical limitations persist:

- **Visual Hallucinations.** MLLMs frequently produce incorrect perceptions, seeing things that are not present in the image, this phenomenon is known as hallucination and is a serious issue for reliability and trustworthiness of such models [5, 6, **?**].

- **Reliability.** MLLMs often struggle with specific visual reasoning tasks, such orientation perception and temporal order perception.
  Orientation perception refers to an MLLM's capability to identify objects based on their spatial positioning or alignment. Despite their advanced multimodal reasoning capabilities, recent evaluations like the BLINK benchmark [7] show that MLLMs often fail at tasks as simple as distinguishing between similar objects that differ only in orientation, for instance, identifying which arrow points differently among several visually similar arrows. Such errors show a gap in the spatial reasoning capabilities of these models, which are crucial to correctly interpret visual contexts in the real world, such as differentiating the direction of vehicle movement.
  Temporal order perception, another critical reliability aspect, involves correctly interpreting sequences of events. MLLMs often struggle to accurately understand the chronological sequence in scenarios involving multiple temporal steps. Evaluations reveal frequent errors when answering simple temporal queries, such as identifying which event occurred first from a series of temporarily ordered events [7]. This limitation poses a substantial risk, especially in surveillance contexts, where accurate reconstruction of event timelines is crutial for chronicle reconstruction.

- **Evaluation Difficulties.** Assessing multimodal models is less straightforward than evaluating text-only models, as multiple correct interpretations may exist. Having multiple valid answers makes traditional metrics like BLEU [8] only partial indicators of performance. To address this issue, in the recent years, AI and Computer Vision researchers have published multiple specialized benchmarks such as BLINK [7], LLaVA-Bench [9], POPE [5], and COCO [10], aiming to comprehensively evaluate multimodal performance.

1.3 Methodology

All the work is written in Python and most of it is stored on GitHub [1]. The code is split into three simple folders. The first folder reads the raw VIRAT videos and turns them into images, with all the data processing steps. The second folder builds the final benchmark: completes the textual and visual prompt configurations, and saves the ready-to-test dataset. The third folder runs the tests. It calls Qwen-2 VL or InternVL through PyTorch, and computes the evaluation of results.

[1]https://github.com/nora826/Vehicle-Centric-Surveillance-Benchmark-VIRAT

The project was coded in VS Code with a Python 3.11 virtual environment. Main libraries are pandas and NumPy for data, OpenCV and Pillow for images, Streamlit for the small web tool, PyTorch and Transformers for the models, and scikit-learn and Matplotlib for metrics and plots.

1.4 Planning

The work is organised as seven successive phases:

1. **Literature Review & Project Setup**. Read about existing state of the art benchmarks and models and analyse existing gaps in surveillance tasks requiring orientation perception and temporal understanding.
   Approximately 1–2 Weeks.
2. **Data Preparation.** Collect the relevant VIRAT releases, standardise annotations, and extract a clean, vehicle-focused subset.
   Approximately 4–5 Weeks.
3. **Question Generation.** Produce representative images from existing data and pair them with multiple-choice questions focused on movement and orientation.
   Approximately 2 Weeks.
4. **Dataset Configurations.** With existing cleaned and prepared image-question pairs. Prepare multiple image configurations for same input text, combining different visual cues that might help the model.
   Approximately 2–3 Weeks.
5. **Model Evaluation.** Select two state of the art and open source multimodal language models, that can work with such input and that may give good overall performance, and test them on the dataset.
   Approximately 3–5 Weeks.
6. **Result Analysis.** Test with different visual prompt configurations and evaluate the performance depending on the introduced image configuration.
   Approximately 2–3 Weeks.
7. **Writing.** Compile results and finalize the documentation, incorporating feedback and ensuring all documents meet the requirements.
   Approximately 1–2 Weeks.

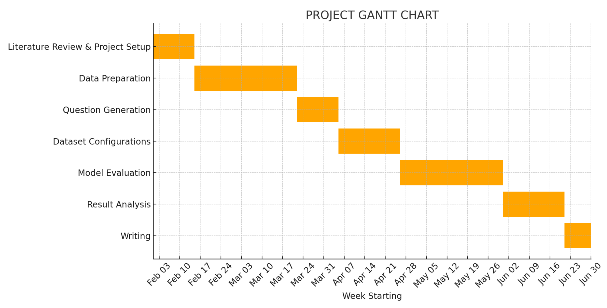


Fig. 1: Gantt Chart.

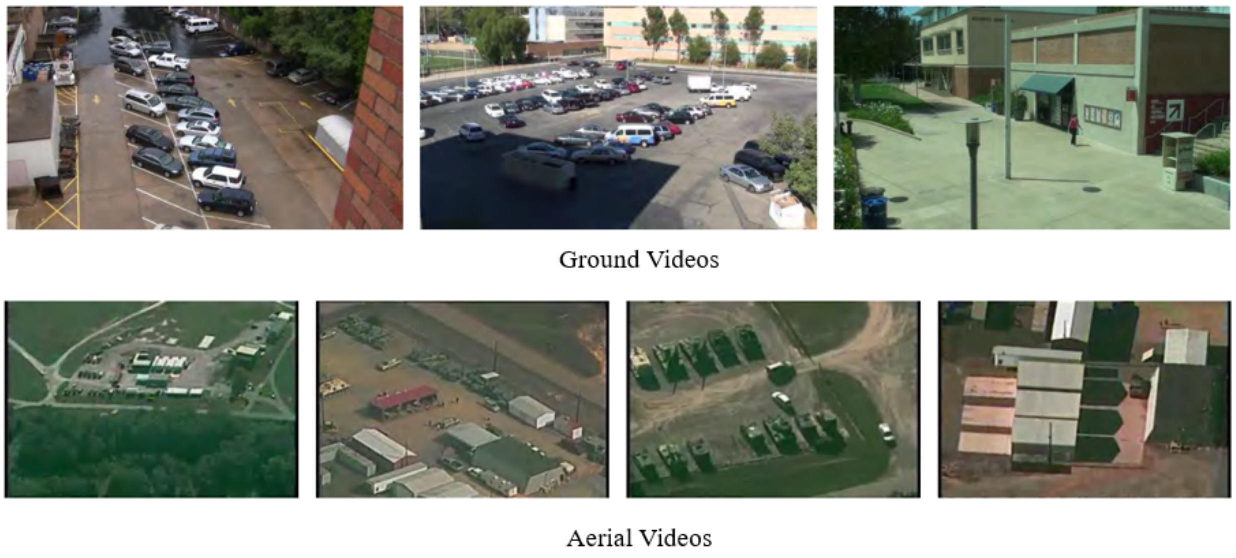


Fig. 2: Sample of Images extracted from Videos belonging to Ground and Aerial Sets.

2 Dataset Overview

2.1 Introduction

The Video Image Retrieval and Analysis Tool (VIRAT) dataset is a large-scale video collection that contains two broad categories of activities, single-object and two-objects which involve both human and vehicles. VIRAT provides extensive real-world video footage annotated with detailed descriptions. The dataset aims to facilitate research in activity detection, object tracking, event recognition, and scene understanding for surveillance purposes. It contains two distinct sets captured from different perspectives [11]:

- **Aerial Videos**: Approximately 4 hours of footage recorded from aerial vehicles. These provide broad-area coverage from elevated viewpoints, useful for tracking large-scale movements and activities.
- **Ground Videos**: 12.5 hours of publicly available videos recorded from stationary cameras positioned on top of buildings. Ground videos offer fixed perspectives, beneficial for detailed observations of human behaviours and vehicle interactions at street level. This project only uses Ground videos. See Figure 8 to see all the scenes included in ground dataset.

2.2 Versions and Evolution

2.2.1 Release 1.0 (2011)

The original release featured annotations focused on basic interactions involving people and vehicles, categorized into 6 distinct event types. Events primarily include vehicle-human interactions, see Table 1.

2.2.2 Release 2.0 (2012)

In its subsequent release, the dataset is expanded to include a wider range of activities, increasing to a total of 12 distinct event types. Annotations become more comprehensive, detailing activities such as running, entering or exiting buildings and carrying objects, see Table 1. Each event includes bounding boxes precisely labelling the spatial locations of involved objects and people while the event occurs. This release contains around 8.5 hours of video annotations for 11 of the 12 videos of different scene recordings, see Table 6.

2.2.3 Extended Release (2020)

The most recent and comprehensive version, is based on the previous release, without any new video recordings but with different annotation types and format. This release is part of the IARPA[2] DIVA[3] program.

This last release becomes more extensive in the variety and complexity of annotated events, now covering 46 distinct event types. This dataset includes advanced vehicle manoeuvrers together with granular human activities, see Table 1. However, the total annotated hours decreases significantly, from around 8.5 hours of the first release to around 4.5 hours in this one, since we have annotations only for 5 of the 12 different scene recordings, see Table 6.

V-1.0 (6)	Opening/Closing Vehicle Trunk; Loading/Unloading Object into Vehicle; Getting into/out from Vehicle
V-2.0 (12)	V-1.0; Digging, Carrying an Object, Running, Entering/Exiting Facility, Gesturing
Extended (46)	V-2.0; Interacting, Standing, Talking, Vehicle Turning Right/Left, U-Turn, Walking, Vehicle Stopping, Phone Texting, etc.

TABLE 1: Activity Types included in each VIRAT Release

[2]Intelligence Advanced Research Projects Activity
[3]Deep Intermodal Video Analytic

2.3 Advantages

- **Realistic Scenarios.** Captured in natural outdoor environments with background clutter and varied lighting conditions. Supports multi-object, multi-activity recordings within single frames, reflecting real-world complexity.
- **Detailed Annotations.** Each version provides bounding boxes and precise temporal information for activities, enabling tracking and activity recognition tasks.
- **High-Resolution Videos.** Footage is typically available in 1080p or 720p, offering detailed visual information. Down-sampled versions facilitate experimentation on devices with limited computational resources [12].
- **Low-Resolution Action Coverage.** Even in high-quality footage, actions at a distance appear in low resolution, mirroring real surveillance conditions. This contrasts VIRAT with many actor-centric datasets that focus on clearly visible, high-resolution actions [12]. See 3.

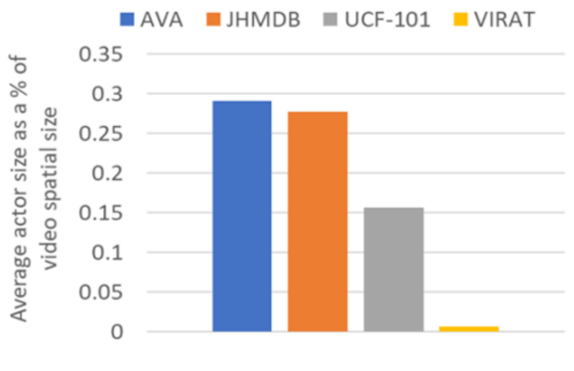


Fig. 3: Average Actor Size.

2.4 Limitations

- **Lack of Indoor Scenarios.** The dataset is limited to outdoor areas, limiting variety.
- **Limited Scene Variety.** Consists of 329 videos spanning 11 distinct scenes, primarily parking lots.
- **Smaller Scale Compared to Other Datasets.** VIRAT is smaller relative to newer large-scale datasets.
- **Real-World Challenges**
  - **Occlusions**: Frequent overlapping and partial visibility of objects.
  - **Scale Variations**: Objects can appear at different sizes.
  - **Lighting Variations**: Outdoor conditions change throughout the day.
  - **Crowded Scenes**: Multiple agents moving in different directions.

3 Benchmark Dataset Generation

3.1 Dataset Creation Process

I. Source Datasets The VIRAT dataset was initially collected to reflect realistic surveillance scenarios. The ground videos were captured by stationary RGB cameras placed at fixed outdoor locations (such as building rooftops) across multiple sites. These cameras have elevated points, capturing everyday scenes like parking lots. All footage was recorded during daylight under natural lighting conditions, containing complex and uncontrolled backgrounds with frequent incidental activities.

Scene	VIRAT 2.0	Extended Release
VIRAT_S_0000	64	1394
VIRAT_S_0001	104	0
VIRAT_S_0002	107	2437
VIRAT_S_0100	365	0
VIRAT_S_0101	59	0
VIRAT_S_0102	425	0
VIRAT_S_0400	90	1011
VIRAT_S_0401	109	2150
VIRAT_S_0500	108	712
VIRAT_S_0501	0	0
VIRAT_S_0502	70	0
VIRAT_S_0503	54	0

TABLE 2: Number of activities per scene from two different dataset versions

To enhance the diversity of activities and scenes, this project utilizes both VIRAT 2.0 release and Extended Release. Although VIRAT 2.0 contains fewer annotations per video compared to the Extended release, it provides a wider variety of scenes as indicated in 2.

- **VIRAT Release 2.0 (2012)** This dataset consists of ground camera footage totaling approximately 8.5 hours of video (in HD resolution: primarily 1080p or 720p) from 11 outdoor scenes. In practice, the footage is segmented into over 500 individual video clips. The frame rate of these videos ranges from 25 to 30 frames per second (FPS).

  Annotations: The annotation format consists of three separate text files per video:

  1. `objects.txt`: Contains bounding-box coordinates for annotated objects.
  2. `events.txt`: Lists events with timestamps (start and end times).
  3. `mapping.txt`: Connects objects to their corresponding event annotations.

  Annotations are encoded numerically and require mapping to predefined dictionaries containing 12 activity classes and 5 object classes. Additionally, bounding boxes appear exclusively within the temporal duration of annotated events. Consequently, actors that do not participate in any of the 12 specified activities remain entirely unlabeled.

  Total annotated events: 1,555 instances across 12 distinct activity types.

- **Extended Release (2020)** The Extended Release re-annotates the original VIRAT ground videos comprehensively, providing detailed multi-object tracking and significantly expanded annotations. This version tracks every moving actor continuously, from their entry into the scene to their exit, annotating all relevant activities across the entire clip. This annotation strategy results in richer, more informative data with 46 granular activity types.

  Annotations: Provided in YAML file format, structured into three files per video:

  1. `regions.yml`: Contains object and bounding-box details for specific timestamps.
  2. `activities.yml`: Details each activity instance, specifying the activity type, duration (start and end timestamps), and participating actors.
  3. `types.yml`: Maps object IDs to their semantic categories (e.g., person, car, etc.).

  This reorganization facilitates easier data interpretation and enables detailed, temporal event analysis at the frame-by-frame level.

  Total annotated events: 7,704 instances across 46 distinct activity types.

II. Unification (V0) To unify the dataset versions, two separate JSON files were generated, one for each source datasets explained before. These JSON files were structured identically, in order to later merge into a single unified dataset, referred to as *Version 0 (V0)*. The unification process involved three main steps:

- **Bounding Box Conversion**: The Release 2.0 dataset already provides bounding boxes as simple rectangular coordinates `(x, y, w, h)` for each frame, so only the midpoints need to be calculated. The Extended Release initially represents bounding areas using polygons (lists of corner points). To standardize, each polygon was converted into its bounding box format, adding six explicit fields:

  - `bbox_lefttop_x, bbox_lefttop_y`
  - `bbox_width, bbox_height`
  - `mid_x, mid_y`

  After this conversion step, bounding box representations from both datasets became fully consistent.

- **Vocabulary Alignment**: For Release 2.0, numeric codes representing activity classes and object types, were translated into textual labels. The Extended Release annotations already contained readable labels directly in the YAML files, which were used without alteration.
  Both datasets store these labels uniformly in two common fields: `activity` (for activity type) and `actor_type` (for object classification). Eliminating the need for lookup tables.

- **Dictionary Key Unification**: Events are stored using unique string keys. Each version employs a distinct key pattern, preventing conflicts:

  - **Release 2.0**:
    `videoID + eventID`
    e.g., ”000001_34”

  - **Extended Release**:
    `fileNumber + activityID`
    e.g., ”11_81”

Since key formats differ, both datasets could be safely merged. After merging, the unified *V0* dataset was represented as a single event-centric JSON file.

III. Cleaning (V1) The raw V0 set was pruned taking out trajectories that do not contain information for at least 10 frames[4].

[4]Three frames correspond to ≈0.5 s

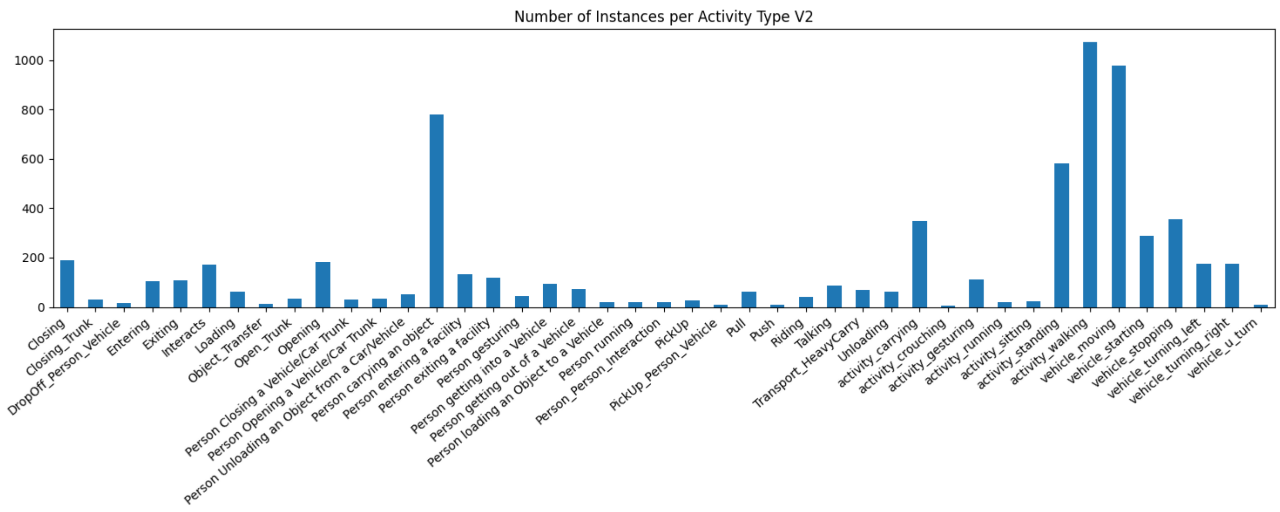


Fig. 4: Activity distribution (V2)

IV. Filtering (V2) Activities that were not interesting for the project such as ”phone texting” were discarded and only a subset of 23 activities was kept. The complete list of activity labels present in each version appears in Appendix B. This *version 2* (V2) contains 4 876 events. Figure 4 shows the per-class distribution. These labels form the basis for all subsequent relabelling and question-generation steps described later.

V. Manual Relabelling of *vehicle moving* Events (V3) As illustrated in Figure 4, activities like *vehicle moving* or *person walking* are much more present than others. The reason for this is that both are general descriptions, that encompass multiple activities within them. After analysing the activities labelled as *vehicle moving*, it became clear that this was a general label encompassing multiple vehicle manoeuvres.

Therefore, I decided to perform a manual relabelling process. This effort aimed not only to generate more instances for some of the existing labels, but also to introduce two entirely new labels. The outcome was a richer dataset, increasing occurrences of existing labels such as *turning left*, *turning right*, and *U-turn*, as well as two new labels: *vehicle moving forward* and *vehicle moving backward*. These additional labels represent fundamental vehicle movements that surveillance models should be able to distinguish.

To facilitate this relabelling process, I developed a web application making use of the streamlit library. The process followed these steps:

1. The initial file (`V2.json`) was duplicated and renamed as `V3.json`, serving as the input for the Streamlit interface.
2. The Streamlit application loaded the `V3.json` file and selected only the events labeled as *vehicle moving*.
3. For each event, the web interface displayed a snapshot of the first frame for that activity, overlaying the vehicle’s complete trajectory. Additional visual cues, such as arrowheads indicating movement direction and bounding boxes highlighting the vehicle position, were optionally provided to facilitate accurate relabelling.
4. Underneath each image, six interactive buttons allowed quick reassignment of each event to one of five predefined labels. Every action taken through the interface immediately updated the data.

Through this manual relabelling process: The number of vehicle manoeuvres significantly increased. The outcome of this relabelling process, is summarized in Figure 5.

VI. Creation of Vehicle-Centric Subset and Final Version (V4) Given that this project targets surveillance tasks involving vehicles, I generated a vehicle-centric subset of the V3 dataset. This subset retains only vehicle-related events, containing nine distinct activity labels, as detailed in Appendix B.

To finalize the dataset (producing version V4), a final label was added. In order to test the model’s ability to recognize non-vehicle activities (negative examples), 150 randomly selected instances of the activity *person walking* were added. These samples serve as negative cases providing *”None of the above”* responses, as explained in further sections.

The final version (V4) of the dataset contains eight labels, coming from VIRAT 2.0, Extended Release or as a consequence of the relabelling process. Figure 5 shows the distribution of events per label, and the source of these events.

Scene	Activity Count
VIRAT_S_0000	113
VIRAT_S_0001	10
VIRAT_S_0002	866
VIRAT_S_0101	30
VIRAT_S_0102	3
VIRAT_S_0400	171
VIRAT_S_0401	117
VIRAT_S_0500	208
VIRAT_S_0502	31
VIRAT_S_0503	14

TABLE 3: Activities per scene in final version.

3.2 Question Generation

With the final dataset version (V4) completed, a subset of vehicle-related activities, the next step was generating structured questions for each of the instances.

Each instance has an image that belongs to the frame where the activity happened, with the overlaying trajectory line, etc. (All these details are explained later in Section 3.3). Each of these images is associated with at least one question. The questions are designed either as three-option multiple-choice questions or as binary (yes/no) queries. The four question templates are the following:

1. **What turn is the vehicle making?**
   A) Left B) Right C) None of the above
2. **In which direction is the vehicle moving?**
   A) Forward B) Backward C) None of the above
3. **What is the person doing?**
   A) Entering the vehicle B) Exiting the vehicle C) None of the above
4. **Is the vehicle making a U-turn?**
   A) Yes B) No

Initially, only one question was assigned to each event. However, to increase the number of ”None of the above” cases, I used existing activity instances, assigning multiple questions to some events.

When selecting true negative samples, I need to take into account that even if an instance was labelled with a single activity, there might be other activities happening at the same time, even if they are not annotated (e.g. vehicle moving forwards and turning right). As an example there are the images shown in Figure 6, that show multiple activities happening simultaneously. Taking such information into account, the following criteria were followed to generate more questions.

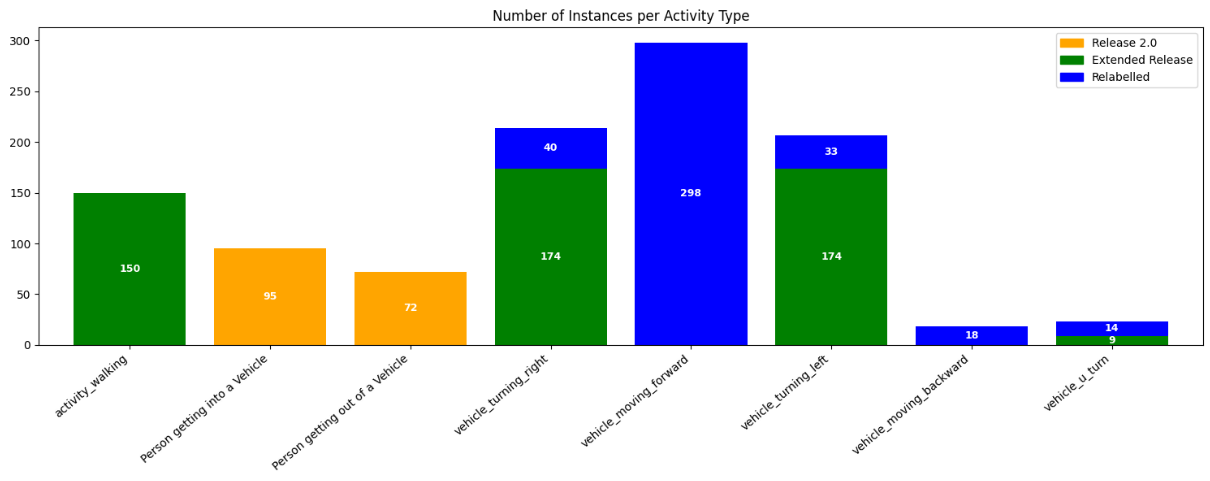
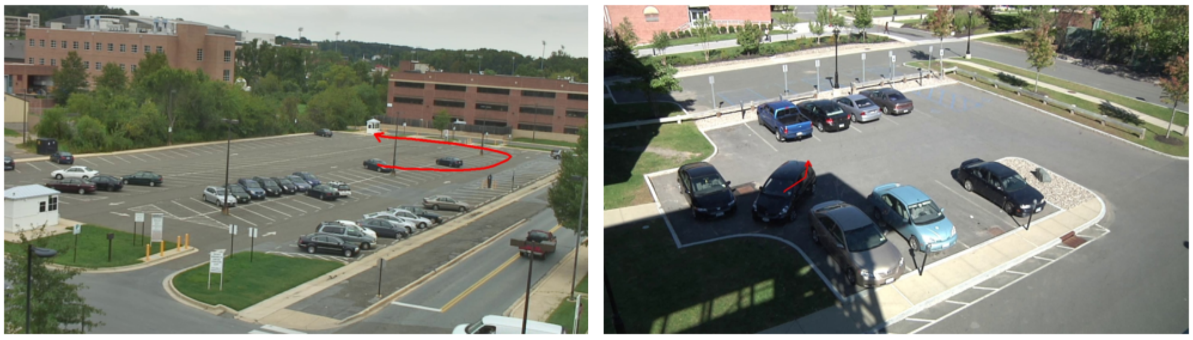


Fig. 5: Distribution of events per label.


The trajectory is labelled as **making U-turn** but the vehicle is also moving forward and turning left.

The trajectory is labelled as **moving backwards** but the vehicle is also turning right.

Fig. 6: Multiple activities within a single Trajectory.

- **Turn questions.** Clips that show a vehicle moving straight forward are used, so the correct answers are C) None of the above for question 1 and B) No for question 4.
- **Direction question.** Clips of a person entering or exiting a stationary vehicle are paired with question 2 because the car is not moving.
- **Person-action question.** 150 randomly selected *person_walking* trajectories are paired with Q3, where the correct answer is C) None of the above.

This strategy ensured that each question template was well-supported with representative negative cases, improving the overall quality and discriminative power of the benchmark dataset. Figure 10 shows the question answer distribution of the final dataset.

After generating these image-question-answer combinations, a final manual review and data-cleaning step was essential. During this stage, I identified three primary issues:

1. Some of the events that were automatically assigned multiple questions were not suitable or irrelevant for certain scenarios. For instance, a stationary vehicle scenario incorrectly assigned a question related to turning. Such inappropriate questions were carefully identified and removed, while suitable ones were retained.
2. Certain annotated events involved background activities or distant occurrences that were barely visible or identifiable, making them invalid as useful samples.
3. Additionally, during the earlier stages of dataset creation, I merged four original activity labels (two from VIRAT Release 2.0 and two from the Extended Release) into two unified labels (*entering* and *exiting*). However, these labels include not only vehicle entry and exit events but also cases related to buildings, which were beyond the scope of this project. Therefore, events depicting people entering or exiting buildings were also removed.

This manual cleaning involved reviewing each image-question pair individually, eliminating inappropriate samples. Through this review process, the total number of questions was reduced from 2297 to 1568 clear and valid image-question pairs. Examples of cases that were removed during the cleaning process are shown in Appendix C.

3.3 Final Dataset

To create a versatile dataset, an interactive web interface was designed, allowing users to personalize various aspects of the dataset, including image configurations, types of questions, and exported data fields.

This interface was implemented using the Streamlit library. When running the web application, users can customize their dataset by selecting from several configuration options, detailed as follows:

- **Question Options:**
  - Select which of four predefined questions to include.
  - Decide whether to include instances labeled as ”None of the above”. If excluded, instances with this as the correct answer are removed, and this choice is also omitted from the question text.
- **Image Options:**
  - Frame Mode:
    - **1x1 mode**: Generates a single representative frame for each event.
    - **3x1 mode**: Creates a mosaic of three frames, capturing the start, middle and end frame of the event.
  - Visual Overlays:
    - Option to overlay the main actor’s bounding box individually onto each seleced frame.
    - Option to overlay the main actor’s complete trajectory line onto each frame. There is also the possibility of adding arrowheads indicating the direction of movement. The trajectory remains the same across all frames belonging to an event.
  - Region of Interest (ROI): To emphasize the activity region, the bounding box based on the trajectory and bounding boxes of all involved actors (e.g., person and vehicle) is calculated. Then there are three optional selections to enhance the ROI:
    - **Blur non-relevant regions**, highlighting only the key event area.
    - **Crop the frame**, keeping only the defined ROI.
    - **Overlay the calculated trajectory bounding box**, marking the area of interest without cropping or blurring.
- **Export Options:** Apart from the data fields that are always stored (such as event key, image path or question), the user can opt to include additional information:
  - Bounding box of the main actor for each corresponding frame.
  - Complete trajectory line.
  - The trajectory enclosing bounding box.

Detailed instructions and descriptions of the web application’s functionalities are provided in Appendix F. After configuration selection, users can download their customized dataset. The structure and contents of the exported files are further explained in Appendix G.

4 Model Testing

4.1 Models Evaluated

Once the dataset was complete, I looked for vision–language models that could run on the small edge devices often used in surveillance.

That led me to two open-source options: **Qwen-2 VL** and **InternVL**, each with about 2 billion parameters. Models of this size fit comfortably on edge devices, where storage, energy and latency are tight constraints. Larger models (e.g Qwen2VL-7B) might give higher performance, but they would slow real-time processing, not valid for realistic surveillance scenarios.

Each model handles the images differently, Qwen-2 VL treats the entire image as one, while InternVL first slices it into multiple patches. By testing both approaches on the same benchmark we can see which architecture works better with details and gives faster, more reliable answers.

4.2 Initialization

The system prompt used is identical for both models:
”*You are a helpful assistant that answers multiple-choice questions about surveillance images. Your answer must be exactly one of the option letters.*”.

4.3 Experimental Setup

Each question was evaluated under eleven visual configurations that combine spatial cues (bounding boxes, trajectories), temporal context (single frame vs. 3×1 mosaic) and visibility help (blur, crop).

Table 4 lists the 11 configurations, images are shown in Figure 7.

Letter	Configuration setting
a	1×1, bbox, trajectory (arrowhead)
b	1×1, bbox, trajectory (plain line)
c	1×1, bbox, trajectory (arrowhead) + blur
d	1×1, bbox, trajectory (arrowhead) + crop
e	3×1, bbox only
f	3×1, bbox + blur
g	3×1, bbox, trajectory (arrowhead)
h	3×1, bbox, trajectory (arrowhead) + crop
i	3×1, bbox, trajectory (arrowhead) + blur
j	3×1, bbox, trajectory (plain line) + blur
k	3×1, bbox, trajectory (plain line) + crop

TABLE 4: Dataset configurations used in model testing.

4.4 Metrics

Each model was presented with the same textual query *eleven* times, but each time it was paired with one of the distinct image configurations listed in Table 4. Because the linguistic prompt remains constant, any change in performance is attributed to the visual information not to textual part.

To quantify performance across these configurations, two complementary metrics were computed together with class-wise confusion matrices:

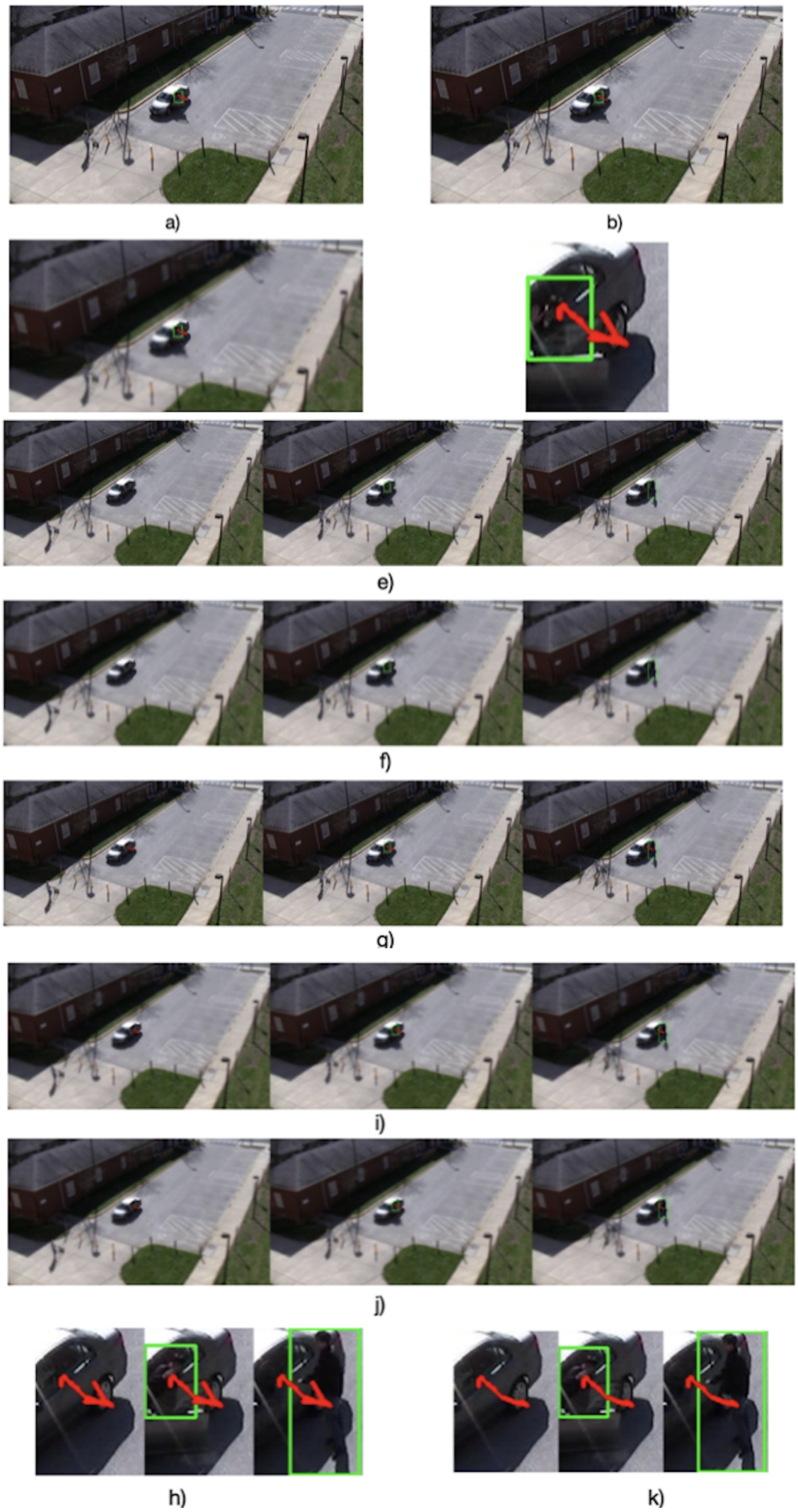


Fig. 7: Same event with different configurations

- **Accuracy.** Accuracy measures the overall proportion of predictions the model gets correct, indicating how often it chooses the right class from all possibilities:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

- **F1-Score.** The F1-score balances precision P (the proportion of correct positive predictions) and recall R (the proportion of actual positives correctly identified). It is calculated individually for each class, then averaged across all classes:

$$\text{F1} = \frac{2 \times P \times R}{P + R}.$$

  Unlike accuracy, the F1-Score treats each class equally, making sure that rare actions (e.g. *U-turn*) are just as important as frequent ones (e.g. *Moving forward*).

- **Confusion Matrix**. In addition to these two metrics, confusion matrices per question were generated to clearly identify the classes the models struggled with and to visualize common misclassification patterns.

4.5 Results

After analysing the impact of different configurations according to the confusion matrices and the accuracy and F1-Score metrics in Table 5, these are the conclusions:

- **Temporal sequence** (3×1) **vs. single frame** (1×1)**:** Using three frames instead of just one usually doesn't help. When comparing similar setups (like a–g, c–i, d–h), the single-frame option (1×1) typically performs better or at least the same. The sequence usually reduces performance, especially for Qwen2-VL.

- **Trajectory with arrowhead vs. plain line:** Arrowheads help Qwen2-VL understand the trajectory better. Removing arrowheads (as in pairs a–b, h–k, i–j) greatly lowers Qwen2's performance. InternVL, however, is almost unaffected, meaning it uses other visual clues.

- **Blurring non-relevant areas:** The effect of blurring depends on the model. Blurring hurts Qwen2-VL's performance consistently. For InternVL, blur has little effect or can even slightly improve results.

- **Cropping the region of interest:** Cropping helps Qwen2-VL perform better and doesn't harm InternVL.

		What-Turn		**U-Turn**		**Direction**		**Enter/Exit**	
		Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
a	Qwen2	0.268	0.141	0.239	0.215	0.679	0.340	0.331	0.240
	InternVL	0.266	0.183	0.969	0.492	0.581	0.367	0.331	0.232
b	Qwen2	0.268	0.141	0.090	0.089	0.681	0.270	0.346	0.211
	InternVL	0.237	0.178	0.969	0.492	0.461	0.264	0.331	0.188
c	Qwen2	0.263	0.105	0.071	0.048	0.678	0.209	0.309	0.129
	InternVL	0.217	0.198	0.969	0.492	0.575	0.296	0.397	0.368
d	Qwen2	0.266	0.140	0.325	0.274	0.681	0.349	0.346	0.266
	InternVL	0.268	0.186	0.969	0.492	0.569	0.362	0.331	0.232
e	Qwen2	0.268	0.141	0.225	0.204	0.678	0.324	0.301	0.237
	InternVL	0.248	0.192	0.969	0.492	0.443	0.243	0.309	0.167
f	Qwen2	0.263	0.105	0.139	0.089	0.476	0.189	0.316	0.172
	InternVL	0.235	0.177	0.969	0.492	0.460	0.268	0.308	0.177
g	Qwen2	0.179	0.076	0.055	0.037	0.270	0.209	0.192	0.140
	InternVL	0.176	0.177	0.969	0.492	0.270	0.176	0.176	0.177
h	Qwen2	0.268	0.141	0.144	0.138	0.681	0.270	0.331	0.187
	InternVL	0.241	0.172	0.969	0.492	0.467	0.268	0.309	0.177
i	Qwen2	0.268	0.141	0.190	0.177	0.681	0.270	0.316	0.190
	InternVL	0.266	0.190	0.969	0.492	0.587	0.354	0.331	0.222
j	Qwen2	0.263	0.105	0.055	0.037	0.669	0.209	0.316	0.139
	InternVL	0.233	0.212	0.969	0.492	0.590	0.301	0.397	0.368
k	Qwen2	0.268	0.141	0.088	0.087	0.681	0.270	0.346	0.211
	InternVL	0.245	0.189	0.969	0.492	0.473	0.269	0.331	0.187

TABLE 5: Accuracy and macro-F1 by question, configuration and model.

When testing single frames (1×1), cropping greatly improved Qwen2's performance (configuration d). In sequences (3×1), cropping also helps both models slightly by removing unnecessary background details.

These results might come due to model's image handling strategy. Qwen-2 VL turns the whole 3 × 1 into one large patch, so fine details such as small cars, arrowheads, pedestrians might be lost into a single token and become harder to recognise. That is why it benefits from anything that shrinks the scene (cropping), sharpens cues (arrowheads), or removes noise (single-frame mode), and why extra blur or extra panels usually hurt.

InternVL, in contrast, first splits the mosaic into several patches and then reduces tokens with a pixel, without shuffling. For this reason, fine details are not lost and that is probably the reason for having a better performance.

In future work, Qwen-2 VL might improve it's performance by receiving the three frames as separate images or by adding a brief system prompt that says "the left, centre, and right parts of the picture are start, middle and end-time frames."

5 Conclusions

This project focused on the development and testing of a benchmark dataset designed to evaluate how effectively Multimodal Large Language Models (MLLMs) handle apparently simple surveillance tasks. Specifically, the dataset examines orientation perception and temporal sequencing capabilities, two essential aspects for reliable surveillance analysis.

The dataset has proven to be challenging for current MLLMs despite being intuitive and easy for human interpretation. This performance gap highlights significant limitations in existing models, which shows the need for improvements in spatial and temporal reasoning.

Given that humans can accurately and effortlessly perform these tasks, the expectation is that artificial intelligence should similarly achieve human-level understanding. Therefore, this dataset can be used as a valuable benchmark for future research and development for improving robustness and reliability of multimodal language models in surveillance applications.

6 Acknowledgment

I would like to express my gratitude to Zenjie Li and Kamal Nasrollahi, tutors during my internship in Milestone Systems, for their constant supervision and for granting me the opportunity to carry out my final-degree project within the company. I am also thankful to Sergio Escalera, external tutor at the Computer Vision Center (CVC), whose guidance was really helpful throughout the research. Finally, I thank to Jordi Gonzàlez, my university tutor, for ensuring that the work satisfied all academic requirements and for the extra feedback.

REFERENCES

[1] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Tang, J. Wang, J. Yang, J. Tu, J. Zhang, J. Ma, J. Yang, J. Xu, J. Zhou, J. Bai, J. He, J. Lin, K. Dang, K. Lu, K. Chen, K. Yang, M. Li, M. Xue, N. Ni, P. Zhang, P. Wang, R. Peng, R. Men, R. Gao, R. Lin, S. Wang, S. Bai, S. Tan, T. Zhu, T. Li, T. Liu, W. Ge, X. Deng, X. Zhou, X. Ren, X. Zhang, X. Wei, X. Ren, X. Liu, Y. Fan, Y. Yao, Y. Zhang, Y. Wan, Y. Chu, Y. Liu, Z. Cui, Z. Zhang, Z. Guo, and Z. Fan, "Qwen2 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2407.10671

[2] B. Li, Y. Zhang, D. Guo, R. Zhang, F. Li, H. Zhang, K. Zhang, P. Zhang, Y. Li, Z. Liu, and C. Li, "Llava-onevision: Easy visual task transfer," 2024. [Online]. Available: https://arxiv.org/abs/2408.03326

[3] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, B. Li, P. Luo, T. Lu, Y. Qiao, and J. Dai, "Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks," 2024. [Online]. Available: https://arxiv.org/abs/2312.14238

[4] S. Hu, Y. Tu, X. Han, C. He, G. Cui, X. Long, Z. Zheng, Y. Fang, Y. Huang, W. Zhao, X. Zhang, Z. L. Thai, K. Zhang, C. Wang, Y. Yao, C. Zhao, J. Zhou, J. Cai, Z. Zhai, N. Ding, C. Jia, G. Zeng, D. Li, Z. Liu, and M. Sun, "Minicpm: Unveiling the potential of small language models with scalable training strategies," 2024. [Online]. Available: https://arxiv.org/abs/2404.06395

[5] Y. Li, Y. Du, K. Zhou, J. Wang, W. X. Zhao, and J.-R. Wen, "Evaluating object hallucination in large vision-language models," 2023. [Online]. Available: https://arxiv.org/abs/2305.10355

[6] G. Geigle, R. Timofte, and G. Glavaš, "Does object grounding really reduce hallucination of large vision-language models?" in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 2728–2742. [Online]. Available: https://aclanthology.org/2024.emnlp-main.159/

[7] X. Fu, Y. Hu, B. Li, Y. Feng, H. Wang, X. Lin, D. Roth, N. A. Smith, W.-C. Ma, and R. Krishna, "Blink: Multimodal large language models can see but not perceive," 2024. [Online]. Available: https://arxiv.org/abs/2404.12390

[8] H. Saadany and C. Orăsan, "Bleu, meteor, bertscore: Evaluation of metrics performance in assessing critical translation errors in sentiment-oriented text," in *Proceedings of the Translation and Interpreting Technology Online Conference TRITON 2021*, ser. TRITON 2021. INCOMA Ltd. Shoumen, BULGARIA, 2021, p. 48–56. [Online]. Available: http://dx.doi.org/10.26615/978-954-452-071-7$_0$06

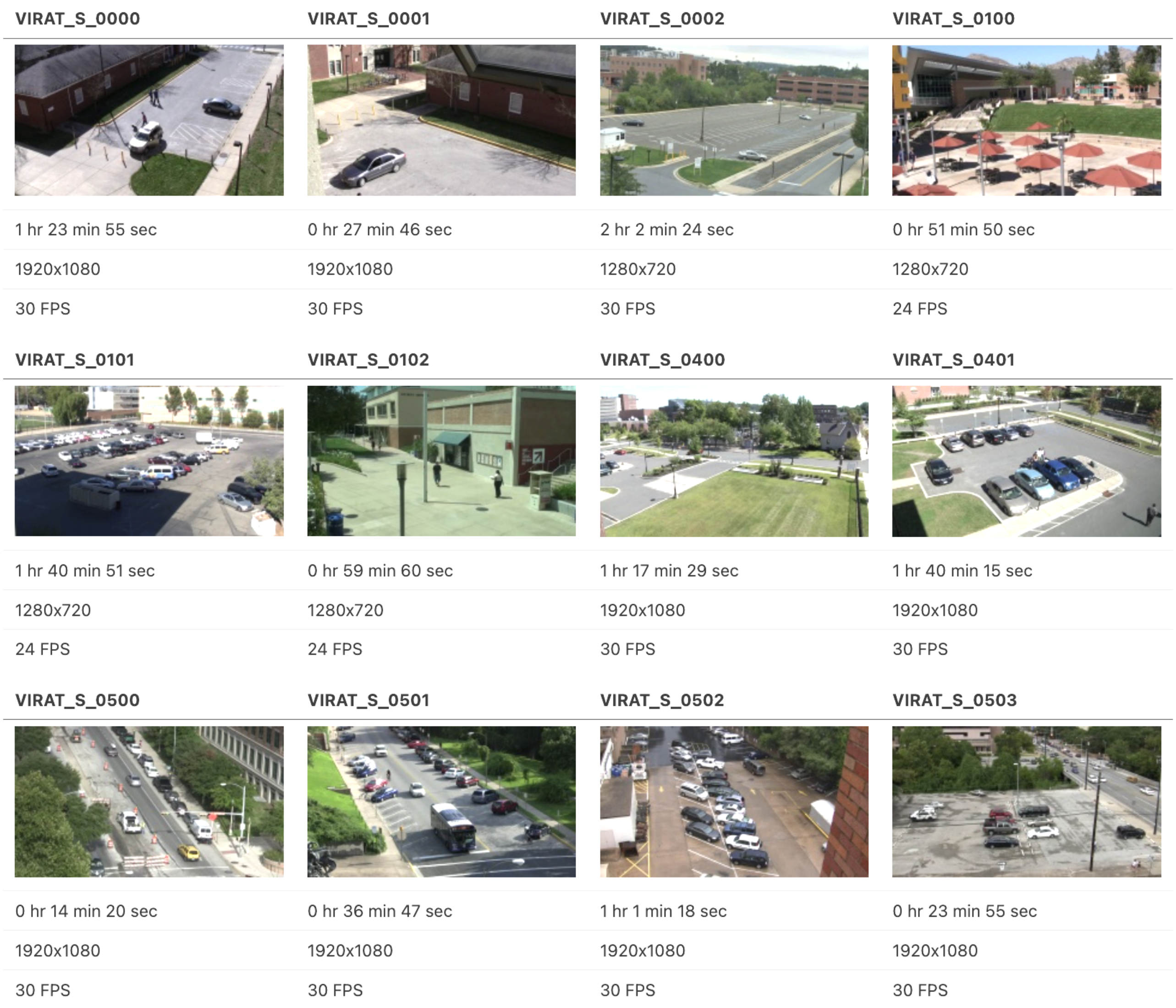
[9] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," 2023. [Online]. Available: https://arxiv.org/abs/2304.08485

[10] S. Singh, A. Yadav, J. Jain, H. Shi, J. Johnson, and K. Desai, "Benchmarking object detectors with coco: A new path forward," 2024. [Online]. Available: https://arxiv.org/abs/2403.18819

[11] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai, "A large-scale benchmark dataset for event recognition in surveillance video," in *CVPR 2011*, 2011, pp. 3153–3160.

[12] U. Demir, Y. S. Rawat, and M. Shah, "Tinyvirat: Low-resolution video action recognition," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 7387–7394.

A Video Annotation Hours Per Scene and Release



VIRAT_S_0000	VIRAT_S_0001	VIRAT_S_0002	VIRAT_S_0100
1 hr 23 min 55 sec	0 hr 27 min 46 sec	2 hr 2 min 24 sec	0 hr 51 min 50 sec
1920x1080	1920x1080	1280x720	1280x720
30 FPS	30 FPS	30 FPS	24 FPS
VIRAT_S_0101	**VIRAT_S_0102**	**VIRAT_S_0400**	**VIRAT_S_0401**
1 hr 40 min 51 sec	0 hr 59 min 60 sec	1 hr 17 min 29 sec	1 hr 40 min 15 sec
1280x720	1280x720	1920x1080	1920x1080
24 FPS	24 FPS	30 FPS	30 FPS
VIRAT_S_0500	**VIRAT_S_0501**	**VIRAT_S_0502**	**VIRAT_S_0503**
0 hr 14 min 20 sec	0 hr 36 min 47 sec	1 hr 1 min 18 sec	0 hr 23 min 55 sec
1920x1080	1920x1080	1920x1080	1920x1080
30 FPS	30 FPS	30 FPS	30 FPS

Fig. 8: Vidoes of VIRAT initial Release 1.0, also used for Release 2.0 and Extended Release.

Camera	# videos 2020	Video durations 2020	# videos V2	Video durations V2	Total video duration
VIRAT_S_0000	6	0 hr 52 min 33 sec	5	0 hr 47 min 53 sec	1 hr 23 min 55 sec
VIRAT_S_0001	0	0	2	0 hr 27 min 46 sec	0 hr 27 min 46 sec
VIRAT_S_0002	47	1 hr 11 min 14 sec	31	1 hr 15 min 23 sec	2 hr 2 min 24 sec
VIRAT_S_0100	0	0	58	0 hr 51 min 50 sec	0 hr 51 min 50 sec
VIRAT_S_0101	0	0	46	0 hr 44 min 7 sec	1 hr 40 min 51 sec
VIRAT_S_0102	0	0	76	0 hr 59 min 60 sec	0 hr 59 min 60 sec
VIRAT_S_0400	22	0 hr 56 min 39 sec	28	1 hr 17 min 29 sec	1 hr 17 min 29 sec
VIRAT_S_0401	34	1 hr 12 min 9 sec	17	0 hr 32 min 26 sec	1 hr 40 min 15 sec
VIRAT_S_0500	10	0 hr 10 min 38 sec	14	0 hr 14 min 20 sec	0 hr 14 min 20 sec
VIRAT_S_0501	0	0	0	0	0 hr 36 min 47 sec
VIRAT_S_0502	0	0	24	0 hr 45 min 2 sec	1 hr 1 min 18 sec
VIRAT_S_0503	0	0	14	0 hr 23 min 55 sec	0 hr 23 min 55 sec
Total	**119**	**4 hr 23 min 12 sec**	**315**	**8 hr 20 min 10 sec**	**12 hr 40 min 49 sec**

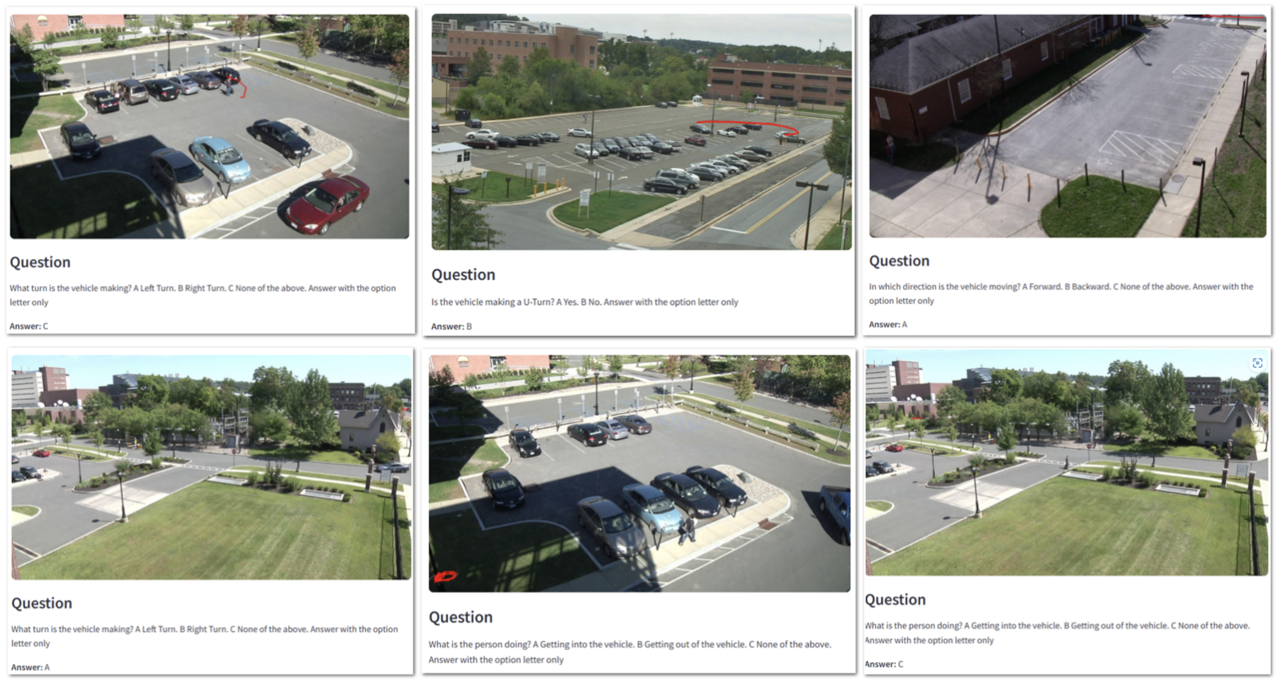
TABLE 6: Camera-wise Video Statistics for VIRAT Dataset

B ACTIVITIES INCLUDED IN EACH VERSION

Dataset	Scope	Activities
V0	Activtiies in VIRAT 2.0 and Extended Release	Closing, Closing Trunk, Drop, DropOff Person Vehicle, Entering, Exiting, Interacts, Loading, Misc, Object Transfer, Open Trunk, Opening, Person Closing a Vehicle/Car Trunk, Person Opening a Vehicle/Car Trunk, Person Unloading an Object from a Car/Vehicle, Person carrying an object, Person entering a facility, Person exiting a facility, Person gesturing, Person getting into a Vehicle, Person getting out of a Vehicle, Person loading an Object to a Vehicle, Person running, Person Person Interaction, PickUp, PickUp Person Vehicle, Pull, Push, Riding, SetDown, Talking, Transport HeavyCarry, Unloading, activity carrying, activity crouching, activity gesturing, activity running, activity sitting, activity standing, activity walking, specialized miscellaneous, specialized talking phone, specialized texting phone, specialized throwing, specialized umbrella, specialized using tool, vehicle moving, vehicle starting, vehicle stopping, vehicle turning left, vehicle turning right, vehicle u turn.
V1	All from V0 (after cleaning out the non-valid instances)	””
V2	Only the activities that might be relevant	Person loading object to vehicle; Person getting into vehicle; Person unloading object from vehicle; Person getting out of vehicle; Person entering facility; Person exiting facility; Person running; Exiting; Opening; Closing; activity standing; vehicle stopping; activity walking; vehicle turning right; vehicle moving; activity carrying; vehicle starting; vehicle turning left; activity running; Unloading; Loading; Entering; vehicle u turn
V3	Previous activities without vehicle moving and two new labels generated during relabelling	(same as V2) plus: vehicle moving forward; vehicle moving backward
V3.2	Vehicle-Centric Subset	Person getting into a vehicle; Person getting out of a vehicle; Exiting; vehicle turning right; vehicle moving forward; vehicle turning left; vehicle moving backward; Entering; vehicle u turn
V4	Vehicle-Centric with Negative Instances for Questions	Exiting; vehicle turning right; vehicle moving forward; vehicle turning left; vehicle moving backward; Entering; vehicle u turn; activity walking

TABLE 7: Activities kept in each dataset version

C CLEANING PROCESS



Question

What turn is the vehicle making? A Left Turn. B Right Turn. C None of the above. Answer with the option letter only

Answer: C

Question

Is the vehicle making a U-Turn? A Yes. B No. Answer with the option letter only

Answer: B

Question

In which direction is the vehicle moving? A Forward. B Backward. C None of the above. Answer with the option letter only

Answer: A

Question

What turn is the vehicle making? A Left Turn. B Right Turn. C None of the above. Answer with the option letter only

Answer: A

Question

What is the person doing? A Getting into the vehicle. B Getting out of the vehicle. C None of the above. Answer with the option letter only

Question

What is the person doing? A Getting into the vehicle. B Getting out of the vehicle. C None of the above. Answer with the option letter only

Answer: C

Fig. 9: Example of removed activities during cleaning process.

D TECHNICAL DETAILS ON FINAL DATASET

Activity	Instances	Avg Duration (s)	Avg Actor Area (px) [5]
backward	16	9.2896	38911.3125
entering vehicle	65	3.3974	11102.1385
exiting vehicle	61	2.5607	12577.1311
forward	225	6.6221	17173.2222
left turn	172	4.2647	24397.5233
none	982	5.5012	19348.8147
right turn	191	5.0574	22561.2775
u turn	13	19.0359	16636.0769
walking	149	19.5667	4880.2953

TABLE 8: Activity Metrics for Final Version

[5]Calculated using the bounding box corresponding to the best frame for each event.

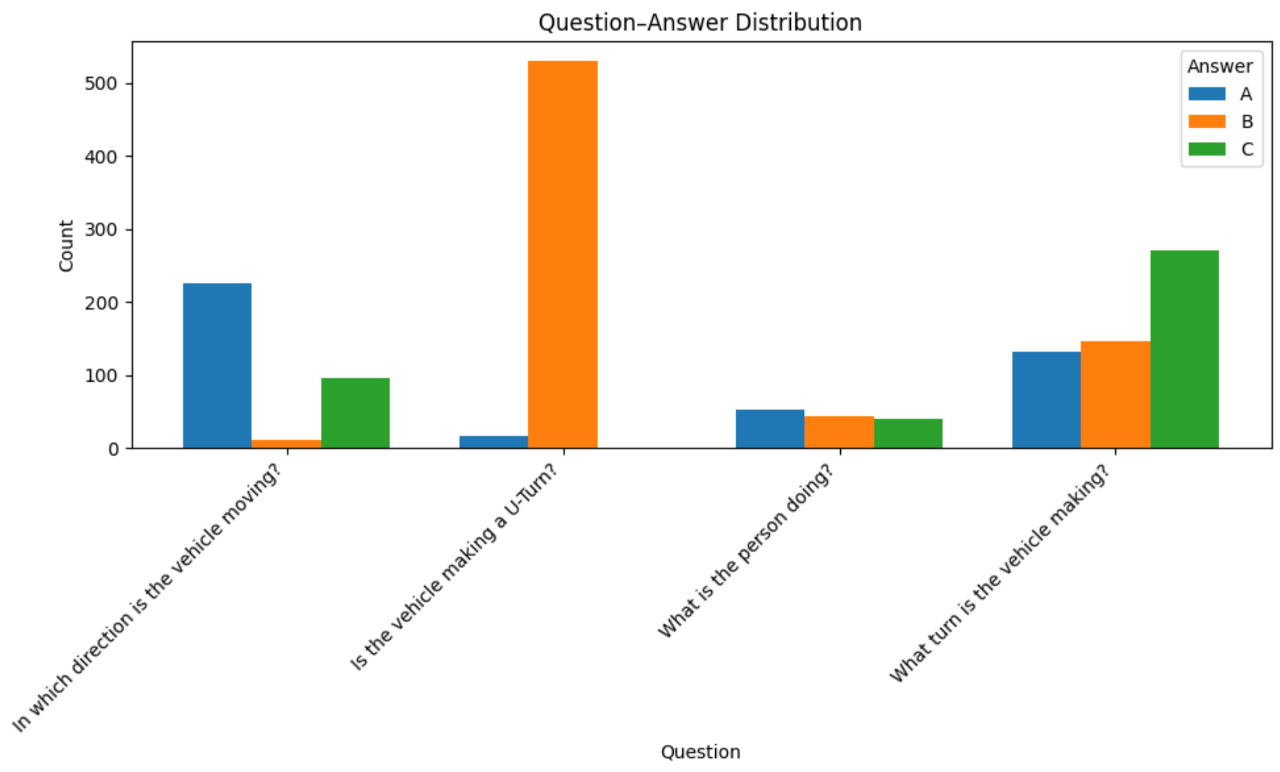


Fig. 10: Distribution of Answers per Question

E Annotation Format

Attribute	Description
Event Key	Unique identifier for each event, combining dataset and event specifics.
Question	Natural language question associated with the event, including the correct answer on the second line.
Image	File path to the image showing the event.
Source	Indicates the original dataset source (e.g., VIRAT).
Frame Number	List of one or three frame indices used for rendering (depending on the layout).
Object Bounding Box	*(optional)* For `save_bbox=True`, a JSON-formatted list of bounding boxes per frame (one box for single-frame mode, or three boxes for 3×1 mode).
Trajectory Bounding Box	*(optional)* For `save_traj_bbox=True`, the single bounding box that encloses the entire trajectory of the primary actor.
Trajectory	*(optional)* For `save_traj=True`, a list of (frame, x, y) centre coordinates per frame.

TABLE 9: Annotation Attributes for Each Dataset Event

F Dataset Generation with Desired Configuration

The dataset is generated using a Streamlit-based interactive app that takes as input a *raw dataset* explained in Appendix H. The webapp is initialized running this command:

```
streamlit run configurations_dataset.py
```

This opens the interface where the user can specify which questions to include and select visual/rendering options.

F.0.1 Sidebar Controls

- **Include "None of the above"**: Checkbox (`include_none`) to mantain or remove questions/entries whose correct answer is "C". In case of not including it, the multiple-choice questions will also not include the choice C, just A and B.
- **Question Selection**: One checkbox per question. Only events with selected questions will be included.

F.0.2 Display Options

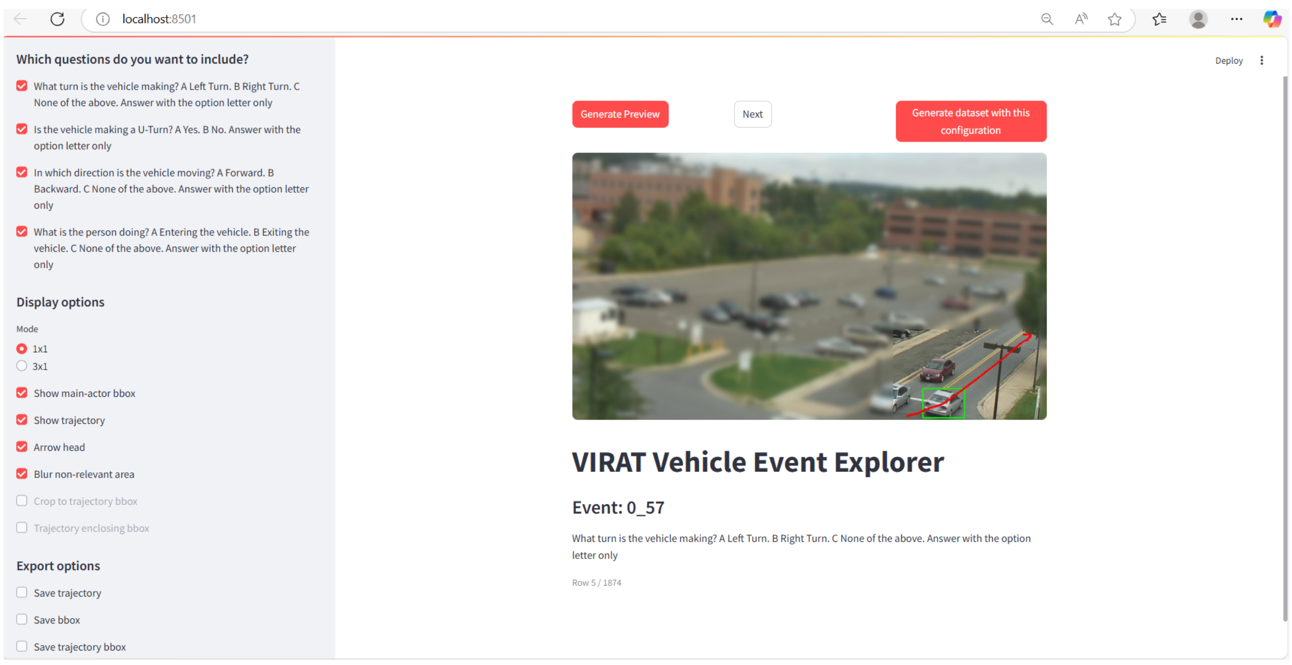
- **Mode**:
  - *1×1*: Extracts the single best frame (`img_best`).
  - *3×1*: Extracts start, middle, and end frames generating a single horizontal mosaic image (`img_start`, `img_mid`, `img_end`).
- **Show main-actor bbox**: Draws the bounding box around the actor at the chosen frame(s).
- **Show trajectory**: Overlays the trajectory line in red, extracted from actors bounding boxes across frames.
- **Arrow head** (`show_arrow`): Adds an arrow at the end of the trajectory.
- **Blur non-relevant area**: Applies a Gaussian blur outside the trajectory bounding box.
- **Crop to trajectory bbox**: Crops the image to the tight trajectory bounding box.
- **Trajectory enclosing bbox**: Draws a box around the full trajectory (used to compute blur or crop)

F.0.3 Export Options

- **Save trajectory**: Includes the raw trajectory centres (frame, x, y) in the TSV.
- **Save bbox**: Includes the list of object bounding boxes per frame (one or three frames) in the TSV column (depending on the selected mode).
- **Save trajectory bbox**: Includes the trajectory-enclosing bounding box in the TSV.

F.0.4 Preview and Navigation

- **Generate Preview**: Generates the current row event with the chosen options.
- **Next**: Goes to the next event.
- **Generate dataset with this configuration**: Applies the current filters and image generation options to all events and generates images together with `data.tsv` and `configuration_info.txt`.



Which questions do you want to include?

- ☑ What turn is the vehicle making? A Left Turn. B Right Turn. C None of the above. Answer with the option letter only
- ☑ Is the vehicle making a U-Turn? A Yes. B No. Answer with the option letter only
- ☑ In which direction is the vehicle moving? A Forward. B Backward. C None of the above. Answer with the option letter only
- ☑ What is the person doing? A Entering the vehicle. B Exiting the vehicle. C None of the above. Answer with the option letter only

Display options

Mode
- ● 1x1
- ○ 3x1
- ☑ Show main-actor bbox
- ☑ Show trajectory
- ☑ Arrow head
- ☑ Blur non-relevant area
- ☐ Crop to trajectory bbox
- ☐ Trajectory enclosing bbox

Export options

- ☐ Save trajectory
- ☐ Save bbox
- ☐ Save trajectory bbox

VIRAT Vehicle Event Explorer

Event: 0_57

What turn is the vehicle making? A Left Turn. B Right Turn. C None of the above. Answer with the option letter only

Row 5 / 1874

Fig. 11: Streamlit interface for custom dataset generation.

G Final Dataset Structure

After the dataset generation, the directory `configuration_1/` contains:

```
configuration_1/
    images/
    data.tsv
    configuration_info.txt
```

G.0.1 configuration_info.txt

Shows the configuration that was chose.

```
mode:               1x1
bbox:               False
trajectory:         True
arrow:              True
traj_bbox:          False
blur_area:          False
crop_image:         True
save_traj:          False
save_bbox:          False
save_traj_bbox:     False
include_none:       True
questions_included: ['What turn is the vehicle making? A Left Turn. etc.]
```

G.0.2 data.tsv

It contains one row per exported image. The columns are the following:

- `event_key`: Unique activity identifier.
- `question`: Question text (with "\nCorrect Answer" at the end).
- `img`: Path to the generated JPG.
- `source`: Original dataset origin (e.g., VIRAT).
- `frame_number`: Comma-separated list of 1 or 3 frame indices used for final image generation.
- `object_bbox` (optional): Contains the list of bounding boxes (1 or 3) in JSON format.

- `trajectory_bbox` (optional): Single JSON list [x0, y0, w, h].
- `trajectory` (optional): List of (frame, x, y) centres in JSON format.

H Raw Dataset

The Streamlit `configurations_dataset.py` file takes as input a raw dataset. This raw dataset is generated by the script `generate_raw_dataset.py`, which processes the last version of the generated dataset 3.json (explained in dataset generation section).

- Four plain JPEG frames per event (`start`, `middle`, `end`, and `best`),
- Full trajectory information (list of `(frame, x, y)` centres) in JSON format,
- Object bounding boxes per frame (smallest actor bbox) when available,
- A TSV file (`data.tsv`) that consolidates Q&A prompts, frame paths, bounding boxes, trajectory-bbox and raw trajectory data.

H.0.1 Individual Processing

For each event in the input JSON:

1. The script reads the start and end frame of the activity. According to that it extracts the corresponding frames from the source video.
   - Start frame: $\texttt{first\_frame} + 1$
   - End frame: $\texttt{last\_frame} - 1$
   - Middle frame: $(\texttt{first\_frame} + \texttt{last\_frame})/2$
   - Best frame: the frame whose actor bounding box has the largest area and lies completely inside the image (i.e., not touching the edges). If no such frame exists, the start frame is chosen.
2. For each of the four selected frames, the actor bounding box is extracted. In case of multiple actors, such as person entering the vehicle, the smallest actor "person" is selected.
3. Independently, the information of the trajectory is extracted, based on the actor's bounding box across the frames. It creates a sorted list of $(\texttt{frame}, \texttt{mid\_x}, \texttt{mid\_y})$.
4. With all the information already created and stored, it finally generates the questions as explained in the *Dataset Generation* section. In case one event has more than one questions, in the output tsv file, the row corresponding to that event is duplicated, having the exact same information for image paths, trajectory, bounding box, etc. except for the question.

This `data.tsv` and its images create the “raw dataset” that is then used by `configurations_dataset.py`.