Multimedia Tools and Applications (2024) 83:26505–26524
https://doi.org/10.1007/s11042-023-16664-4

WebLabel: OpenLABEL-compliant multi-sensor labelling

Itziar Urbieta[1,2] · Andoni Mujika[1] · Gonzalo Piérola[1] · Eider Irigoyen[1] · Marcos Nieto[1] · Estibaliz Loyo[1] · Naiara Aginako[2]

Received: 5 December 2022 / Revised: 20 July 2023 / Accepted: 23 August 2023 /
Published online: 31 August 2023

Abstract
Annotated datasets have become crucial for training Machine Learning (ML) models for developing Autonomous Vehicles (AVs) and their functions. Generating these datasets usually involves a complex coordination of automation and manual effort. Moreover, most available labelling tools focus on specific media types (e.g., images or video). Consequently, they cannot perform complex labelling tasks for multi-sensor setups. Recently, ASAM published OpenLABEL, a standard designed to specify an annotation format flexible enough to support the development of automated driving features and to guarantee interoperability among different systems and providers. In this work, we present WebLabel, the first multipurpose web application tool for labelling complex multi-sensor data that is fully compliant with OpenLABEL 1.0. The proposed work analyses several labelling use cases demonstrating the standard's benefits and the application's flexibility to cover various heterogeneous requirements: image labelling, multi-view video object annotation, point-cloud view-based labelling for 3D geometries and action recognition.

Keywords OpenLABEL · Ground truth · Video labelling · Point cloud · Tracking · 3D

1 Introduction

Machine Learning (ML) is fuelling the development of Autonomous Vehicles (AVs). As in other domains, ML has grown exponentially in reliability and performance, providing techniques to recognise what objects are present, localising the objects in 2D and 3D, determining the objects' and scene's attributes, characterising relationships between objects and providing a semantic description of the scene [1]. Such capabilities enable the creation of perception functions as the basis upon which AV functions can be built, such as lane-keeping systems, lane change assist, automatic braking, parking, etc.

✉ Itziar Urbieta
iurbieta@vicomtech.org

1 Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Mikeletegi 57, 20009 Donostia-San Sebastián, Spain

2 University of the Basque Country (EHU/UPV), Donostia-San Sebastián, Spain

As part of safety-related functions in AV, ML functions shall be validated before being deployed into real roads and vehicles. To that end, it is necessary to demonstrate its behaviour, e.g., by measuring its detection accuracy or equivalent metric, in all those situations where the function is supposed to work correctly [2, 3]. In the automotive industry, one major approach is to collect a variety of scenes with instrumented vehicles which capture relevant scenes in as many situations as desired. That often huge content requires a rigorous labelling process to create ground truth, i.e., identify, classify, and detail such situations so that the output of the function (e.g., a classification algorithm, a measurement of the distance to a lane marking) can be compared to the ground truth to extract objective measures of performance (aka Key Performance Indicators, KPI).

These recordings are usually multi-sensorial datasets, which combine images with point clouds, odometry and vehicle data. Multi-sensor data labelling is crucial in the automotive industry as it is essential for developing and improving autonomous driving technologies. Autonomous vehicles rely on data from multiple sensors such as cameras, lidars, radars, and GPS to make decisions and navigate complex environments. Labelling this data will improve accuracy by combining data from multiple sensors assuring that autonomous vehicles can have a complete understanding of their environment, leading to better decision-making and safer driving. Labelling actions may then require obtaining specific information about the objects in the scene, such as pedestrians or vehicles, along with scenery (e.g., lanes or traffic signs), behavioural information (e.g., status or actions carried out by the driver or other road participants), and even contextual data about the scene (e.g., road type, country, illumination conditions, etc.). Multi-sensorial data implies that multiple objects may appear at each sensor payload; thus, calibration and synchronisation requirements must be considered during the labelling process.

OpenLABEL 1.0 was released at the end of 2021 by ASAM e.V. [4] to address the need for a harmonised description of a data format that covers the abovementioned multi-sensor object labelling requirements. The standard was conceived as flexible to support a variety of labelling use cases, including simple and classic 2D image labelling types (e.g., bounding boxes, polygons, pixel-wise), 3D point clouds (e.g., cuboids, polylines), but also time intervals as semantically rich periods where something is happening (e.g., an action or event). Thanks to its structure and utilisation of synchronisation and calibration information, OpenLABEL guarantees that objects can receive a single unique identifier despite its many potential representations at each sensor (e.g., a single car projected in 4–8 cameras or LIDAR).

The complexity of 2D, 3D and temporal annotations cannot still be easily solved with automated algorithms, which may give reasonably good detection results, but are not 100% accurate or reliable. Manual supervision is a must, and for vast volumes of data to label, labelling tools need to be crafted to support this Herculean task.

In this work, we present WebLabel, as the first labelling tool that fulfils the requirements for massive labelling of OpenLABEL-compliant complex multi-sensor data. The tool has been devised to structure internal data following the OpenLABEL schema using the open-source library Video Content Description (VCD) [5] in its Typescript version, available as an NPM package. This engine accelerates data management, projection across views, synchronisation, and support for multiple geometries. Most importantly, it guarantees compliance with the OpenLABEL standard. As a web application, WebLabel can easily be deployed and run in cloud platforms to orchestrate the labour of hundreds or thousands of annotators, which can be presented with pre-annotated content run by ML models. We showcase the ability of our tool to support several popular labelling use cases in the automotive industry.

2 Related work

The efforts employed by the stakeholders of the automotive industry focus on the development of autonomous systems. Hence, the sector faces multiple challenges in providing intelligence to these systems. Due to the need for Artificial Intelligence (AI) algorithms, the requirement for multimedia content is proportionally growing.

Throughout the last decade, multiple datasets have been openly released so stakeholders can feed their deep learning (DL) models and develop intelligent systems for developing connected and autonomous vehicles. Common Objects in Context (COCO) [1] is the primary object detection dataset for computer vision that stores the information as JSON files. Pascal Visual Object Classes (Pascal VOC) [5] also contain annotated images with everyday objects. Though following this format, an XML file is created for each image. Besides, the bounding box information is represented differently in both. Most images are manually annotated, such as the ImageNet dataset [7] containing 14.197.122 annotated images hierarchically ordered according to the WordNet. Even if the aim of these datasets is mainly the same, interoperability is not assured since the formats are not standard, and the terminology needs to be formalised.

Many labelling tools are available for object annotation to prepare these datasets. Some of them are mentioned in the analysis done by Kaur and Singh [6]. However, this is completed by further exploring the tools that share some of the tool's features presented in this article.

Roboflow [7] and Toronto Annotation Suite (TORAS) [8] are web-based tools fitting for collaborative image labelling and instance segmentation of objects. Another web-based application is the open-source software published by MIT under the name LabelMe [9], endorsed by a database for browsing and image querying. For online video annotation, one of the alternatives is VATIC [10]. This tool is suitable for objects; though, it also considers other elements, such as Actions understood as temporal attributes of these objects. A further tool that supports video annotation is LabelBox [13]. It contemplates the importance of relationships and allows connecting two annotations by assigning labels and setting whether the relation between them is unidirectional or bidirectional. These element types still need to be fully supported and are only anticipated as object-related.

Additionally, 3D-Bat [11] is focused on labelling objects with bounding boxes in 3D point clouds. Nonetheless, the interoperability problem has reappeared with these tools. Each supports a different format. Some tools are compatible with some of the mentioned datasets; for instance, CVAT [12] is an image labelling tool that supports the structure of the COCO dataset. Due to this, developers usually require extra effort to adapt the data to the required format and translate the terminology.

Furthermore, the structure used by the well-known YOLO [13] neural network is considered an annotation format [14]. For example, among the tools that support both formats, one can find two that are open source. On the one hand, LabelImg [18] is a tool written in Python that is prepared for image annotation. On the other hand, Label Studio [15] is prepared for multiple data types (image, audio, text, time series). It can be integrated with machine learning models for working with pre-labelled data. This application is also provided with converters among eminent machine learning libraries.

Most of the available data and tools are focused on objects. However, the new developments demand more awareness about their surroundings. This concludes by requiring data with a complex semantic description to feed the training processes. In this domain, this is the need to provide a complete description of every element necessary to interpret a scenario. For instance, M.Buechel [16] proposed a method for a semantic description of traffic scenes using ontologies and for improving situational awareness by using semantic classification conditions. Some works have already stated the importance of event or action analysis for developing many applications [17].

Aiming for such interoperability, in this paper, we present our work building WebLabel, a tool that fully supports the OpenLABEL standard. The adoption of the standard can be combined with the use of ontologies for formalising and conceptualising these definitions to assure interoperability. Hence, this paper continues the work presented in [18]. That tool was based on the VCD format, the predecessor of OpenLABEL. We use the new standard's more flexible frame-interval definition to present a more versatile annotation tool.

3 Tool overview

WebLabel is a web application designed to label multimedia content (videos, images, point clouds, etc.). The user interface (UI) can be configured to enable tagging all the element types defined in the ASAM OpenLABEL standard [4]: objects, actions, events, contexts, and relations. Hence, the resulting file obtained with the application is generated according to the standard.

3.1 User interface (UI)

The presented application supports different data source formats; hence, several use cases have been deployed (see Section 6). Yet, the tool has specific characteristics that persist throughout all the configuration cases. For instance, the main parts of the interface are listed and referenced in Fig. 1 and are part of the nomenclature used in the presented work.

A. Upper menu: buttons with essential functions such as saving the changes locally or on the server.

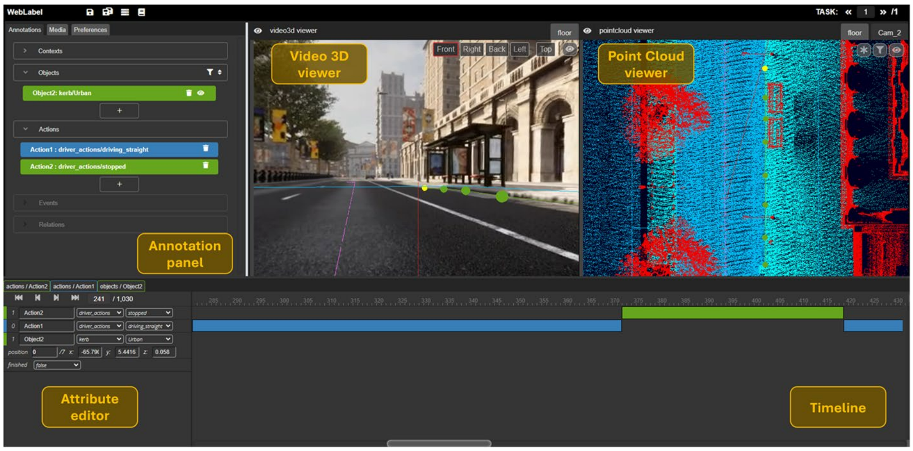


Fig. 1 WebLabel UI main modules

B. Annotations menu: displays all the annotations of the scene classified according to the above-mentioned main OpenLABEL element types. This panel provides options for filtering (by type, time range, current frame, etc.) and ordering (by creation date, frame interval or type/subtype). Each element can be selected and deleted from the panel. Furthermore, they can be marked as 'finished' to exclude them from the general list. This helps the users in their labelling strategy since annotations that do not require further modifications will not obstruct the work.
C. Stream viewer: is the visualiser of the provided data source. The panel layout is configurable, defining the number of viewer panels and the layer types. Several functionalities have been set to visualise the data and navigate along the viewers:

a Zoom in/out on the viewer to see the objects in detail.
b Move along the image, video, or point cloud.
c Change the camera view (*left, front, right, rear*) using the tabs. Depending on the strategy used by the user, for instance, different views can be visualised at once or the same view but having one of them zoomed in.
d Two additional panels can be defined with four cameras, combining all the video views: 360º view navigation and top view tab.
e Additional viewer components: Region Of Interest (ROI) area delimitation, historic ROI, sight ray, etc. (further information about these elements is in Section 6.3).

D. Timeline: provides the means to analyse and modify the duration of each annotation in the scene. The timeline is also the tool to navigate along the sequence and update the viewers to annotate the different elements and the location of each object frame by frame.
To enhance the tool, automatic annotation features are included. When the interpolation mode is activated, interpolation algorithms automatically place the intermediate geometries.
E. Attribute editor: this section is prepared to define the class and all the attributes of the elements. There is also an option to set the element as 'finished' when the labelling is completed.

3.2 Architecture

The architecture of WebLabel is devised to be a multipurpose web-based application compatible with the OpenLABEL standard for labelling objects and scenarios. Besides, the tool is powered by the VCD Toolkit, which allows the translation of the actions done by the user into the human–machine understandable OpenLABEL format.

Figure 2 represents the general architecture of the WebLabel application. Considering the function of the module or file, this architecture can be divided into six main parts:

- Data Tools: these modules oversee the data's loading and management, referring to the annotations shown and edited in WebLabel. Among the modules of this group, behold the files that read, validate, and edit OpenLABEL files; modules to read the configuration files of the application and the ones that serve as a data interface for all the other modules.
- Content Tools: modules that load and process the multimedia content (images, point clouds (PCs), videos, image sequences, etc.) that will be visualised in WebLabel.

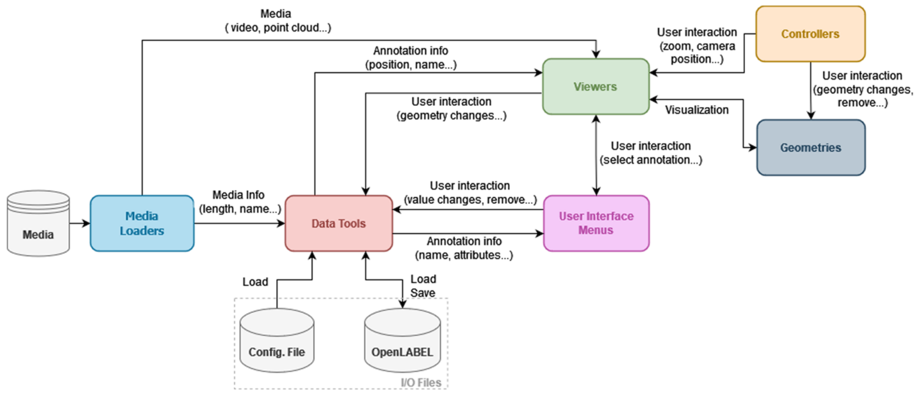


Fig. 2 General architecture of the WebLabel application

- Viewers: these modules manage all the functionalities of the viewers that show the multimedia content and the geometry of the annotations. Each type of multimedia content has its visualisation module and controller, with other generic modules supporting them. Moreover, each geometry type that represents annotations has its module.
- Menus: the functionalities of the different panels (represented in Fig. 1) that compose the web application are controlled by these modules.
- Controllers: each type of viewer has its controller. The controller takes the user's actions with the mouse and the keyboard, transforms them, and distributes them properly. For example, suppose the user drags a point of any geometry. In that case, the controller communicates how the geometry must be changed to the corresponding geometry module. Also, if the user scrolls the mouse wheel in a video, the controller will transfer this information to the corresponding viewer to zoom in or out of the image.
- Geometries: each type of geometry defined in OpenLABEL has its geometry module in WebLabel. These modules control how the geometry is painted in the viewers and handle the movements and changes from the controllers.

Figure 3 depicts the software stack of the WebLabel application. In this figure, we can see which technologies the main parts of the application make use of:

Like any other web application, WebLabel is based on the operating system of the computer and the HTML 5 and JavaScript engine of the browser. In our case, WebLabel has been tested in Google Chrome, working on Windows and Linux. Nevertheless, using 3D environments for the Viewers of the application makes the fundamental part of the software stack different. All the actors in the 3D rendering are depicted in orange. We use the JavaScript library Three.js [19] which uses WebGL [20] to render animated 3D computer graphics. Like any WebGL-based library, it uses the GPU of the computer for rendering.

The Viewers, their connected modules (Controllers and Geometries), the User Interface Menus and, in general, the whole application are orchestrated with AngularJS [21]. This is a web framework for developing single-page applications.

Finally, the Data Tools, the modules that handle and save the user's annotation changes, use the VCD library. This library fully complies with the OpenLABEL format and allows loading, saving, and reading annotation information with simple functions.

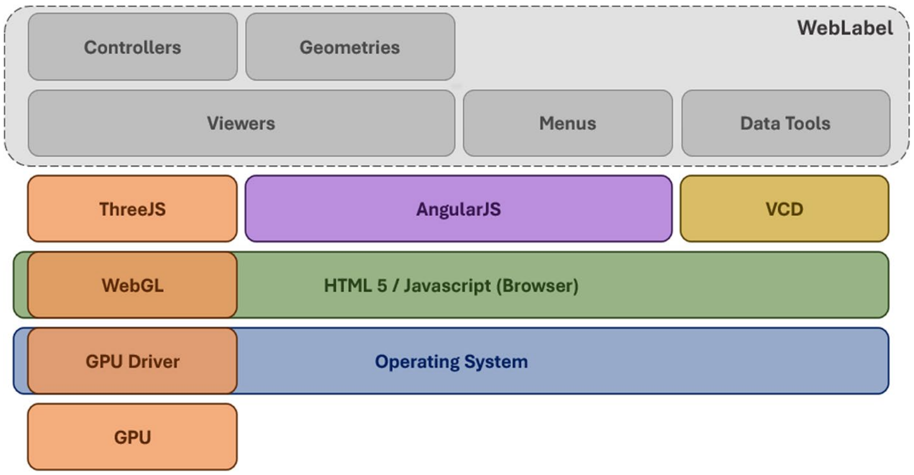


Fig. 3 Software stack of the WebLabel application

3.2.1 Input/Output interfaces

WebLabel requires the following input data:

A **Configuration file:** this file is used to define the layout of the UI and the use case requirements:

I. Task definition: WebLabel can load and display more than one annotation task at once. This is helpful when each task is a relatively simple job. For example, users can load many images simultaneously when they only have to set whether each image is a meme. For each task, the configuration file defines the following:

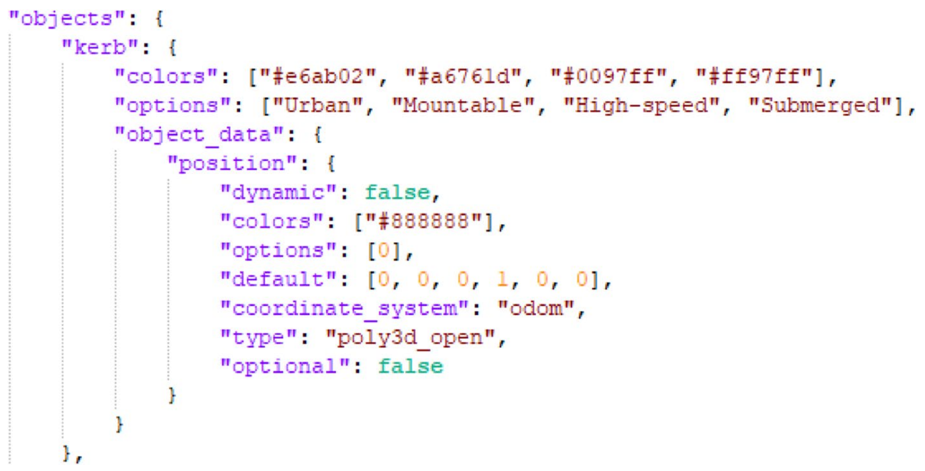
- Viewers layout: the viewer panel of the UI can be divided into multiple layers, and this section of the configuration file is used to define the distribution of the viewer. For example, as depicted in Fig. 1, the viewer can be divided into two main layers: the left side has five tabs containing the four camera views and a top view combining all of them. The viewer's right side can be configured to show the camera view again; however, it shows the point cloud view in this case. This section also points to the source file and the overlays for each view.

  Each task can have a different annotation configuration (cars in one task and pedestrians in another).

II. Element configuration: defines the element types that can be labelled in each use case. In addition, each class will have its related attributes and characteristics that will affect the distribution of the tool.

As an example, Fig. 4 shows the 'Kerb' object class definition, and the relevant information for the user that can be gathered is:

- Each element is defined with specific subtype options to avoid free text. In this case, 'Kerb' has four possible subclasses: 'Urban', 'Mountable', 'High-speed', and 'Submerged'.



```
"objects": {
    "kerb": {
        "colors": ["#e6ab02", "#a6761d", "#0097ff", "#ff97ff"],
        "options": ["Urban", "Mountable", "High-speed", "Submerged"],
        "object_data": {
            "position": {
                "dynamic": false,
                "colors": ["#888888"],
                "options": [0],
                "default": [0, 0, 0, 1, 0, 0],
                "coordinate_system": "odom",
                "type": "poly3d_open",
                "optional": false
            }
        }
    },
```

Fig. 4 Screenshot of the ‘Kerb’ Object class definition in the configuration file

- The object has a related ‘position’ attribute set as ‘static’. Therefore, the labelling will be done on top of a point cloud (see Section 6.3) instead of frame by frame. Moreover, this attribute is not optional, meaning each ‘Kerb’ object must have an assigned position.
- The chosen geometry is ‘poly3d open’, an open polygon in three dimensions to locate the object.

III. Overlays: these are additional elements that can be included in the viewer to help the user during the performance of the labelling task—for example, the region of interest (ROI) or the ego-vehicle trajectory. Further information about these is given in Section 6.3.

IV. Source files: the data sources are established in this section. The input data type can be an image, video, point cloud or a combination.

B **Preferences file**: the file is prepared for configuring the preferences tab, which gives some flexibility to the tool features (select annotation when created, add default interval when creating a new annotation, etc.). Sets the default values for every configurable parameter. This can be modified and saved from the UI tab according to the strategy used by the user

C **Annotations file**: the tool is prepared to work from scratch or have pre-annotations. In the second case scenario, the information is provided as part of the input files and must be formatted according to the latest version of OpenLABEL.

After completing the tasks, the generated data is outputted as an annotations file in OpenLABEL format. The whole sequence is assembled as a unique output file.

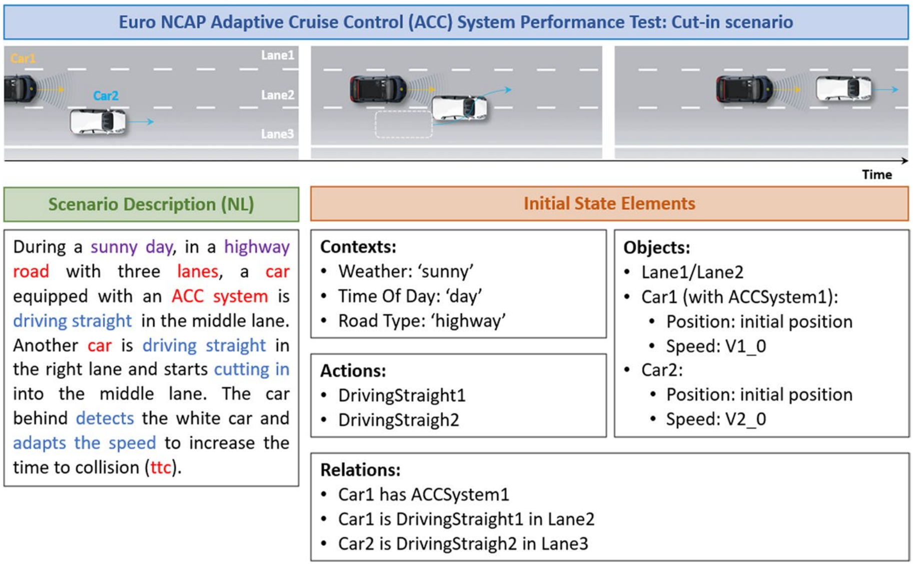
4 OpenLABEL-based functioning: General Labelling Method

OpenLABEL is a standardised annotation format developed mainly focused on the automotive domain. Nevertheless, the structure of the standard is flexible enough to host use cases related to other disciplines, ranging from object-level image annotations to more complex multi-stream semantically enriched scene labelling. The last case could involve considering any element crucial for describing a scene and the relations between them. To illustrate this, a Euro NCAP [22] adaptive cruise control (ACC) system test-case scenario has been considered, and the sequence is divided into three frames in Fig. 5. Likewise, a general description of the scenario chronology is done in natural language, and the different element types are highlighted with distinctive colours. The cut-in scenario description has been completed for the initial state, where each element relevant to the complete definition has been divided by type.

Since the WebLabel application aims to be fully compatible with OpenLABEL, the user interface design is closely linked to the fields and features OpenLABEL contains. Furthermore, the general labelling method is defined according to these characteristics. Thus, we will go through the fields composing the OpenLABEL format in the following, explaining how each affects WebLabel functioning and consequently, how the labelling method is performed.

4.1 Element-based labelling

OpenLABEL defines an Element as an entity that can be labelled with a name, type, unique identifier, and other properties. Moreover, OpenLABEL has a primary classification of Elements (Objects, Actions, Events, Contexts and Relations) [23], which is why the Annotation Panel (see Fig. 1) is divided into five submenus, one for each type of Element.


Euro NCAP Adaptive Cruise Control (ACC) System Performance Test: Cut-in scenario
Car1
Car2
Lane1
Lane2
Lane3
Time
Scenario Description (NL)
Initial State Elements

During a sunny day, in a highway road with three lanes, a car equipped with an ACC system is driving straight in the middle lane. Another car is driving straight in the right lane and starts cutting in into the middle lane. The car behind detects the white car and adapts the speed to increase the time to collision (ttc).

Contexts:
- Weather: 'sunny'
- Time Of Day: 'day'
- Road Type: 'highway'

Actions:
- DrivingStraight1
- DrivingStraigh2

Objects:
- Lane1/Lane2
- Car1 (with ACCSystem1):
  - Position: initial position
  - Speed: V1_0
- Car2:
  - Position: initial position
  - Speed: V2_0

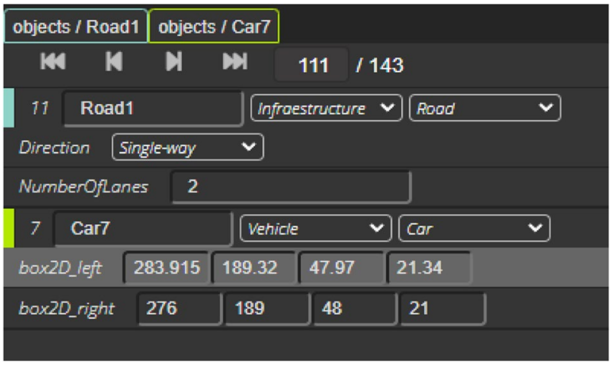
Relations:
- Car1 has ACCSystem1
- Car1 is DrivingStraight1 in Lane2
- Car2 is DrivingStraigh2 in Lane3

Fig. 5 Description of the initial state of the Euro NCAP cut-in scenario for ACC system testing

The user can add, select/deselect, or delete annotations in this menu. It also offers filtering and ordering features to ease users' work.

Once an Element is created and selected via the Annotation Panel, the corresponding information is displayed in the Attribute Editor and the Timeline (see Fig. 1). The information shown in the Attribute Editor is closely related to the attributes of the Elements defined in the OpenLABEL format as depicted in Fig. 6. Each element must have at least the listed associated features. Additionally, the editable ones will be set in the UI (see Fig. 6) the others will be internally managed by the tool when adding new elements or defined in the configuration file.

- **uid** (not editable): a unique identifier is assigned to each element by the VCD toolkit as unsigned integers. These are visible in the UI but are not editable to assure persistence and uniqueness through time.
- **name** (editable): this property is defined as a friendly identifier and must be assigned by the user. It is visible in the annotations panel and on top of the geometries for rapid identification of the objects.
- **type** (editable): defines the semantic type or class of the labelled element, and it can be concretised by assigning a subtype. This will depend on the ontology supporting the data, which orders all the classes of interest for the domain hierarchically and the needs of the use case, so it is only editable by a dropdown menu.
- **ontology id** (not editable): an identifier related to the ontology URL that supports the classes used for labelling is also provided.
- Other attributes (editable): each type of Element will have its attributes. Depending on the type of data (text, number, Boolean, vector…), the Attribute Editor will display the corresponding edition mode (dropdown menu, text field, number input, etc.). Related to this, the standard supports a limited number of primitive geometry types to define the geometry labelling as shown in Fig. 6: 2D bounding box (bbox), 2D rotated bounding box (rbbox), 3D cuboid (cuboid), point in 2D space (point2d), point in 3D space (point3d), 2D polygon defined by a sequence of 2D points (poly2d), 3D polygon defined by a sequence of 3D points (poly3d), reference to an area (area_reference), reference to a line (line_reference), 3D mesh of points, vertex and areas (mesh). The format is set in the tool's configuration file according to the objects of interest for each use case.


objects / Road1 | objects / Car7

111 / 143

11 Road1 | Infraestructure | Road

Direction: Single-way

NumberOfLanes: 2

7 Car7 | Vehicle | Car

box2D_left: 283.915 | 189.32 | 47.97 | 21.34

box2D_right: 276 | 189 | 48 | 21

Fig. 6 'Attribute editor' component of the WebLabel

4.2 Additional concepts for scenario description

WebLabel is prepared to label dynamic data and complete scenarios, which imply a temporal representation of the elements. As a result, additional concepts to elements must be understood and configured for completing the labelling method.

4.2.1 Frames

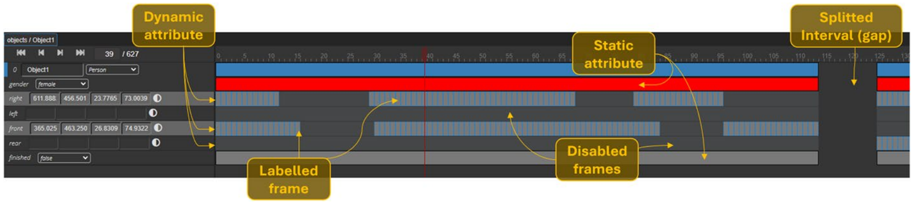
Any information about elements can be static or dynamic, i.e., it can be, by definition, equal for the whole scene, or it can suffer changes during the scene. Indeed, the same element can have both static and dynamic attributes. For example, the colour of a vehicle will be the same for the whole scene, but its position in the video (annotated as a bounding box) will change frame by frame.

In OpenLABEL format, the static information is encoded in the 'element_data' field inside the 'Element' field. The dynamic information instead is defined in the 'Frames' field since the attributes can have a different value in each frame.

Moreover, only some of the annotations will span the whole scene. Usually, an annotation will last only an interval of the scene. It may even have intermittent behaviour. Besides, in the second use case described below, it can happen that while the annotation persists in the scene, one of the attributes disappears for some frames.

This temporal variability of the annotations or their attributes is encoded in OpenLABEL, which defines the static information in the elements field and the dynamic information in the frames field. Thus, this complex behaviour has to be represented and has to be editable in WebLabel. The crucial part of the application for this representation is the timeline.

Both attribute types are depicted in Fig. 7. Conversely, the static attributes (e.g., position, the object's colour, element type, etc.) do not change throughout the scene's duration. Therefore, the annotator will set these attributes once, representing them as a whole block without frames in the timeline. For example, see the 'gender' attribute for 'Object1' in Fig. 7. Conversely, every dynamic element will have a temporal description with frame intervals. The element-specific qualities (e.g., number of lanes, road surface conditions, position, etc.) are set per frame as their value changes along the scene. For instance, the position of the objects is set by placing the corresponding geometry on top of the things visible in the images at every frame, as represented by every blue cell in Fig. 7. Additionally, if the same object is not visible, these frames must be disabled to illustrate that it disappears during that time interval this will be noticeable in the timeline since the edges of the frame are not coloured. Furthermore, in case an object disappears from the scene for a time slot, the frame-interval can be split (see Fig. 7).



objects / Object1
39 / 627
0 Object1 Person
gender female
right 611.888 456.501 23.7765 73.0039
left
front 365.025 463.250 26.8309 74.9322
rear
finished false

Fig. 7 Screenshot of the timeline panel

4.2.2 Streams and coordinate systems

These fields contain information about the recording setup that obtained all the contents that will be shown in WebLabel: cameras, lidars, etc. Properly managing data derived from sensors involves understanding the position and orientation of these devices. Furthermore, the transforms among a source and target coordinate system are also described.

Since this information is fixed in the input annotation file and cannot be edited, WebLabel does not offer the option to alter it.

4.2.3 Metadata

This section summarises basic descriptive information about the file: file or format version, author, file name or any other required comment. The metadata section provides flexibility since it is additional information that could be interpreted by the user and may be used for classification purposes. Nevertheless, it is optional information that does not affect the OpenLABEL functionality.

4.2.4 Ontologies

The standard enables linkage to ontologies via pointers in the annotation files. In this regard, every element can be related to a formally described concept in a knowledge repository. Consequently, if semantic relations are annotated, the applications can be configured for reasoning and inference of further information.

5 Comparison with existing labelling tools

Due to the importance of data labelling to support AI developments, many tools have emerged to provide image, video, and point cloud labelling capabilities. A summary of the comparison done for this work is presented in Table 1. Hence, there are several labelling tools available; nonetheless, to this day, WebLabel is the only one compatible with the OpenLABEL standard, as represented in Table 1. Consequently, it is the only tool that allows annotating all the elements required for a complete description of an automotive scenario. Some tool developments are already working towards being able

Table 1 Comparison table of available labelling tools

Name	Multipurpose	2D and 3D annotations	Multiple geometries	OpenLABEL compliant	(Semi)automatic annotation
WebLabel	🟩	🟩	🟩	🟩	🟩
Roboflow	🟩	🟥	🟩	🟥	🟩
Label Studio	🟩	🟥	🟩	🟥	🟩
CVAT	🟩	🟩	🟩	🟥	🟩
VATIC	🟥	🟥	🟥	🟥	🟩
LabelMe	🟩	🟥	🟩	🟥	🟥
UltimateLabeling	🟩	🟥	🟩	🟥	🟩
LabelBox	🟩	🟥	🟩	🟥	🟩
LabelImg	🟥	🟥	🟥	🟥	🟥
3D-BAT	🟩	🟩	🟥	🟥	🟩
TORAS	🟩	🟥	🟩	🟥	🟩

to label elements other than objects. For example, VATIC contemplates actions as temporal attributes of objects instead of considering them are elements, reducing the information that can be associated with them. For instance, the actions could be defined with attributes and relations that associate them with multiple elements, such as objects or events. LabelBox instead has focused on relations, and currently, the coverage is limited to related objects. On another note, Label Studio considered other element types, this tool labels emotions or sentiments concretely. With OpenLABEL, these elements could be included as contexts in case the labelling of the feeling is done on an image and as an attribute of a person if it must be related to a concrete object. On top of that, in WebLabel, these attributes can be treated as dynamic attributes and annotate the changes over time.

Likewise, most of the analysed tools are prepared for labelling objects using multiple geometries, and most are repeated from one to another. Nevertheless, only CVAT and 3D-BAT support geometries related to 2D and 3D representations. Both tools mentioned for 3D annotation only consider cuboids, whereas the presented tool works with point3d and poly3d. Considering this aspect must be highlighted that WebLabel is the tool that supports more geometries, and that any other application does not consider some of them. This is due to the effort put into developing the 3D use case to support the point cloud (PC) data recorded by LiDar systems that are vital for the autonomous systems being developed in the industry.

Furthermore, most tools allow exporting the data into a typical network or different dataset formats such as COCO, Pascal VOC, YOLO, etc. However, usually, the user must use a converter to obtain the data into the format of interest or use it in other tools. This interoperability problem could be solved by agreeing to use a standardised annotation format such as OpenLABEL.

In terms of functionality, it isn't easy to objectively evaluate which is friendlier for the user or has a better performance for each data source without spending time testing each one. Nonetheless, one of the aspects that is known as helpful for the user is having semi-automatic or automatic annotating features. Most of them cover this feature, even if it is to a different extent. LabelMe and LabelImg tools' default configuration does not provide automatic annotation options. Though, a semi-automatic tool based on the last mentioned and YOLOv5 is available [24]. Otherwise, Roboflow, Label Studio, UltimateLabeling, LabelBox and CVAT allow hosting machine learning models (pre-built or custom) to supply pre-labels and save annotation costs. This is currently done externally in WebLabel; however, we provide pre-labelled data for all the use cases presented in Section 6 using custom-trained models. Besides, WebLabel, as well as CVAT, VATIC and 3D-BAT, have integrated features to ease the typical annotation actions: draw a bounding box, interpolation, filters, and other shortcuts.

6 Use cases

Due to the versatility of OpenLABEL format and, consequently WebLabel, several use cases have been tested during the development of the tool. This section intends to represent such configuration variability and flexibility by describing how the application handles the peculiarities of each use case. The proposed cases are mainly conditioned by the geometry defined for labelling the scene's elements.

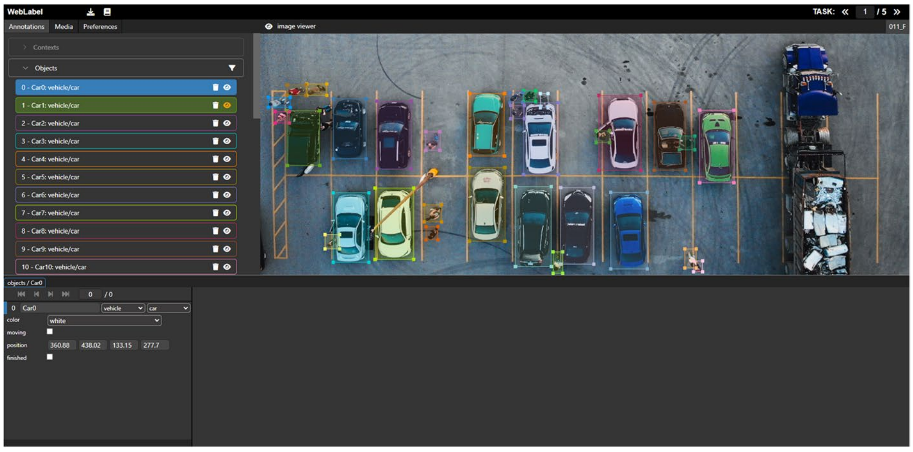
6.1 Image labelling

Image labelling is one of the most extended use cases with the most straightforward functionality. Consequently, WebLabel does not provide significant innovations for this use case, as with other tools, it is prepared to load one or multiple image(s) at once using the tasks feature. Once again, the bounding box should be placed on top of the object of interest and adjusted to its dimensions to perform the labelling task. An example of the UI configuration for this use case is shown in Fig. 8. Other attributes can be added if more information about the elements is required. Features such as bringing a bounding box to the front or sending it to the back have been included for when an overlapping of geometries happens, as shown in Fig. 1. In addition, WebLabel offers the capability of cloning geometries, enabling the automatic generation of a new object with a unique ID and identical properties to the original object. This functionality proves especially beneficial when labelling parking lots in Fig. 8, as it significantly reduces the time required for the task.

6.2 Multi-view object labelling

This use case aims to provide the means to label objects having video as the data source. As represented in Fig. 9, the viewer can be configured to visualise multiple camera views at once. In contrast to usual video players, the viewer can envision the scenario's elements with further detail by zooming in/out on the panel. Thanks to these features, the user can define a personalised labelling strategy according to the needs of the use case. For instance, one of the working options provided is to visualise different camera views to label multiple views with a persistent identifier (see Fig. 9). With this feature, the tracking of the objects of a recording can be done by creating a unique entity and setting the position attributes for all the associated views. This is possible because the identifier of the objects is unique (uuid).

Another working mode could be visualising the same camera view twice but having one zoomed in to have a more precise idea of the objects and the other with a general view of



0 - Car0: vehicle/car
1 - Car1: vehicle/car
2 - Car2: vehicle/car
3 - Car3: vehicle/car
4 - Car4: vehicle/car
5 - Car5: vehicle/car
6 - Car6: vehicle/car
7 - Car7: vehicle/car
8 - Car8: vehicle/car
9 - Car9: vehicle/car
10 - Car10: vehicle/car

objects / Car0

0 / 0

0 Car0 vehicle car
color white
moving
position 360.88 438.02 133.15 277.7
finished

Fig. 8 UI configuration for Image labelling

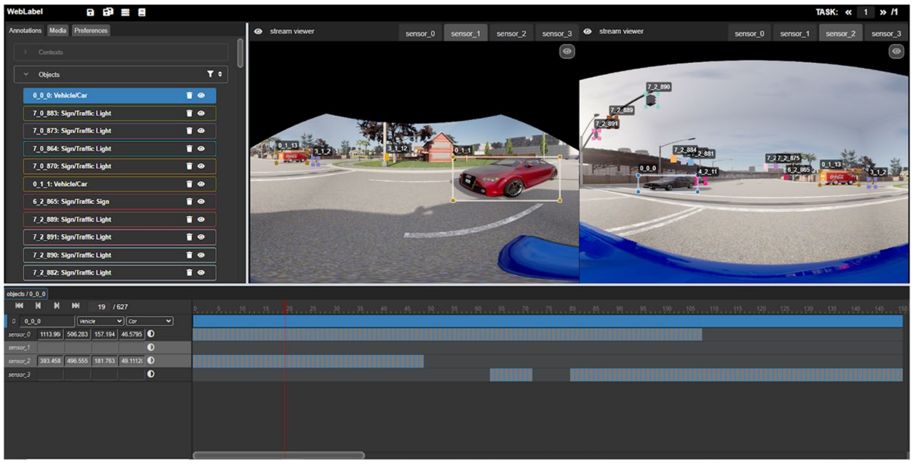

WebLabel
TASK: « 1 » /1
Annotations
Media
Preferences
Contexts
Objects
0_0_0: Vehicle/Car
7_0_883: Sign/Traffic Light
7_0_873: Sign/Traffic Light
7_0_864: Sign/Traffic Light
7_0_870: Sign/Traffic Light
0_1_1: Vehicle/Car
6_2_865: Sign/Traffic Sign
7_2_889: Sign/Traffic Light
7_2_891: Sign/Traffic Light
7_2_890: Sign/Traffic Light
7_2_882: Sign/Traffic Light
stream viewer
sensor_0
sensor_1
sensor_2
sensor_3
objects / 0_0_0
19 / 627
Vehicle
Car

Fig. 9 UI configuration for the multi-view object labelling use case

the scene. Furthermore, the videos can be reproduced and visualised on the stream viewer, and the annotations corresponding to each instant will be shown on top of the images.

To perform the labelling task, the user should define the position of the objects by frame and for every camera view, which provides a very detailed labelling result. For this purpose, the assigned geometries are bounding boxes. However, any geometry of the OpenLABEL format can be used if needed. Moreover, to set the position for a whole frame interval it is helpful to use the interpolation method and then make small changes to the frames that are not correctly assigned with this automated method.

Since the automotive domain is the area of interest in this use case, a synthetic scenario generated with the Carla simulator [28] has been used for multi-view labelling. An ego-vehicle with four cameras has been used for the simulation, and the annotations shown in Fig. 9 have a persistent identifier. As an example of tacking the objects, note that the red van in Fig. 9 has the same name, '0_1_13', in both camera views. Additionally, the configuration of the object classes to annotate is done based on the definitions of the Automotive Global Ontology (AGO) [25]. Hence, the selected categories are: 'Vehicles' with 'Car', 'Motorcycle' and 'Bus' subclasses; 'Pedestrian' and 'Signs' considering the 'Traffic light' and 'Traffic sign' subtypes.

6.3 Point cloud view-based labelling (3D)

Two use cases show how point-cloud-based labelling has been implemented in WebLabel: 'Curb Stones Detection (CSD)', using an open polygon in 3D [26] and 'Parking Space Detection (PSD)' with a closed polygon in 3D. Indeed, as these use cases are in 3D, a height attribute is given to each point of the geometry to complete the position information of the objects. An example is depicted below in Fig. 10.

These labelling tasks are performed on the point cloud or the provided top view. Moreover, the tool offers a 360° visualisation constructed by combining the videos of the four cameras (front, rear, right and left).

The general view of the top view contains the following elements:

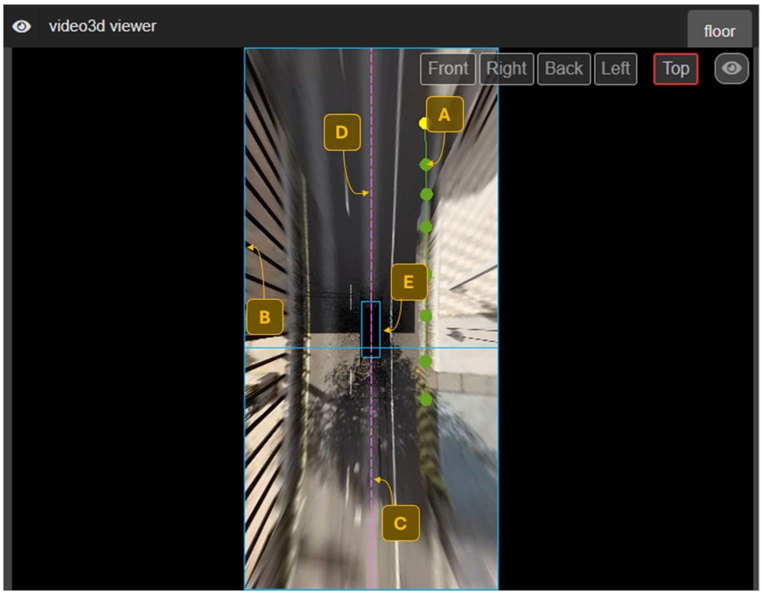


Fig. 10 Example of the Top view tab in the video 3D viewer of the WebLabel

A. Curb (CSD) annotations
B. ROI: the region of interest determines the annotation area. Follows the movement of the ego vehicle.
C. Ego-vehicle trajectory: represents the path of the vehicle along the scene.
D. Sight Ray: represents the camera heading.
E. Ego: depicts the contour of the ego-vehicle.
F. Historic ROI: represents the evolution of the ROI over time for the scene's duration. The red section in Fig. 11 represents the historic ROI.

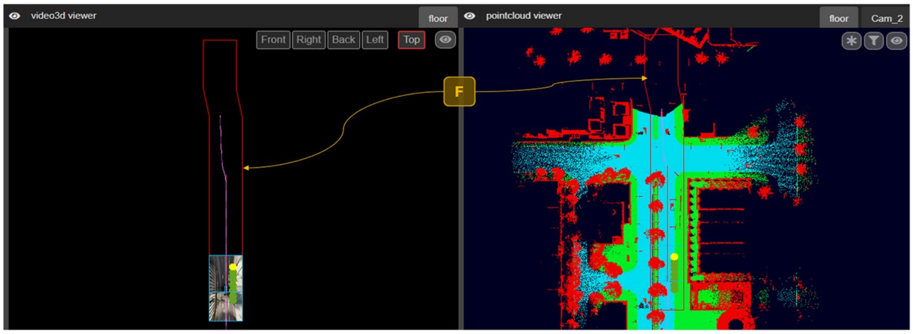


Fig. 11 Top view and Point cloud view zoomed-out representation to visualise the historic ROI

6.4 Action recognition

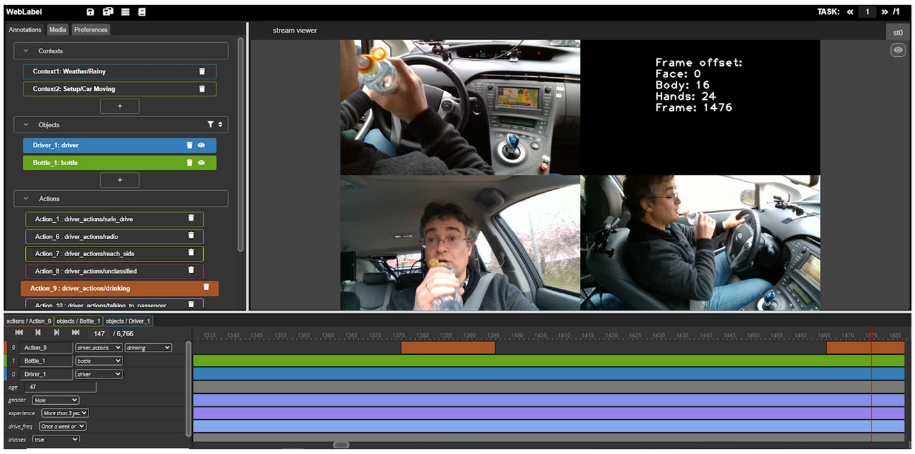
Although the use cases described above were all focused-on object annotation, OpenLABEL, and WebLabel, can handle other element types. The annotation process of the Driving Monitoring Dataset (DMD) [27], which was partly done with WebLabel (see Fig. 12), can show how the application can be used to annotate Contexts, Objects and Actions.

Considering that the information provided by the DMD dataset is related to the occurrence of concrete actions, one of the features that may help analyse the results is to zoom in and out on the timeline. This feature gives the user a global view of the whole sequence. Besides, the timeline has also split and merging options to set the duration of each action. When dividing an interval, gaps will be visible in the timeline, as represented with 'Action_9' in Fig. 12.

In addition, WebLabel can interrelate the different element types to enrich the labelled data semantically. The elements defined as 'Relations' in OpenLABEL are used for this purpose. To define these elements, after specifying the relation's name, the triple's subject and object should be identified. They will be chosen from the general lists of contexts, objects, actions or events. Hence, frame intervals can be also assigned to the relation.

7 Conclusion

The present work's main contribution is presenting the first OpenLABEL-compatible labelling tool. Considering this feature, WebLabel can be used for two main objectives: multisensory labelling for ground-truth generation and tag annotation for labelling scenarios. Both cases provide a solution to the needs of the latest technological developments. On the one hand, generating datasets and ground-truth data is crucial for training AI models and enhancing algorithms. On the other hand, the automotive sector has manifested a rising interest in simulation-based testing and formalising a pipeline, for it requires providing formalised complete scenario descriptions. Hence, with WebLabel,



Context1: Weather/Rainy
Context2: Setup/Car Moving
Driver_1: driver
Bottle_1: bottle
Action_1 : driver_actions/safe_drive
Action_6 : driver_actions/radio
Action_7 : driver_actions/reach_side
Action_8 : driver_actions/unclassified
Action_9 : driver_actions/drinking
Action_10 : driver_actions/talking_to_passenger

```
Frame offset:
Face: 0
Body: 16
Hands: 24
Frame: 1476
```

actions / Action_9 | objects / Bottle_1 | objects / Driver_1
147 / 6,766
9 Action_9 driver_actions drinking
1 Bottle_1 bottle
0 Driver_1 driver
age 47
gender Male
experience More than 3 yea
drive_freq Once a week or
glasses true

Fig. 12 UI configuration for the Action recognition case with DMD data

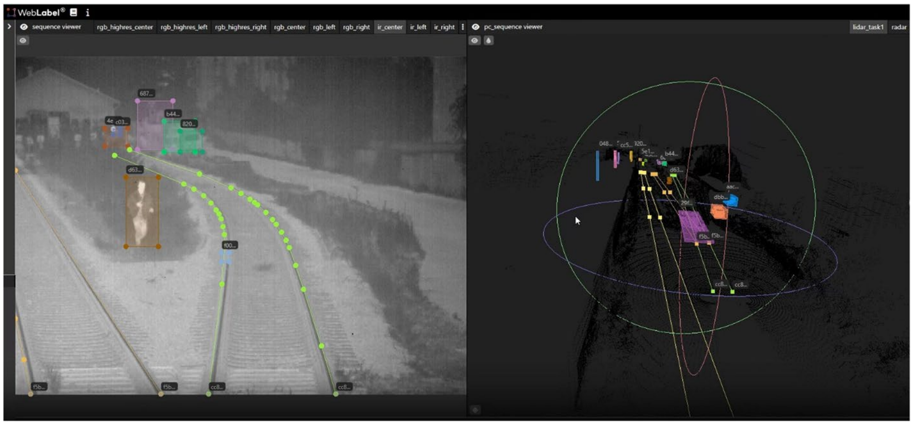


Fig. 13 Snippet of the "Open Sensor Data for Rail 2023" (OSDaR23) dataset [30] visualization using WebLabel

data captured by equipped vehicles can be labelled to define concrete scenarios. These files can be further processed to ingest the data into a simulation for testing Automated Driving Systems (ADS).

On top of that, considering the comparison done in Section 5, along with CVAT, WebLabel is the most flexible tool. It is multi-purpose, and several tasks can be loaded simultaneously to ease the managing process and back-end configuration requirements to set up the tool for the final users. Besides, 2D and 3D labelling is supported by different geometry types. The results of the annotation tasks performed with this tool can be exported in multiple formats. Nonetheless, it does not support OpenLABEL, the standard annotation format. In this aspect, up-to-date WebLabel seems to be the only labelling tool fully compatible with OpenLABEL. Therefore, it appears to be the only tool to label elements other than objects over time. In this case, the element types supported by WebLabel are contexts, objects, actions, events and relations.

Consequently, this will enable generating heterogeneous datasets for the training algorithms and enhance the development of ADS systems. On top of that, these processes increasingly require information with related elements as complete scenario descriptions. To our knowledge, WebLabel is the only tool that enables full labelling of recordings for a thorough understanding of the scenes.

Moreover, the tool's usability has been tested for different use cases using actual recordings and images. The presented cases are related to automotion; however, the UI is easily configurable, and the geometries supported by the tool can be used for other domains.

WebLabel is ready to use, in consequence, a Github repository [28] has been created to showcase its applications. Recently an open dataset created by German Centre for Rail Traffic Research at the Federal Railway Authority (DZSF), Digitale Schiene Deutschland / DB Netz AG, and FusionSystems GmbH has been published [29]. The dataset consists of 45 sequences of annotated multi-sensor data and the WebLabel can be used to visualize them [30] as shown in Fig. 13. As part of future work, two new use cases will be tested: image segmentation (pixel-wise) and point-cloud segmentation.

Acknowledgements This work has received funding from the European Union's Horizon Europe Research and Innovation Programme under Grant Agreement No 101076754 - AIthena project.

Funding This work was funded by the Horizon Europe programme of the European Union, under grant agreement 101076754 (project AITHENA). Funded by the European Union. Views and opinions expressed here are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

Data availability Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

Declarations

Financial interests The authors declare they have no financial interests.

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Lin T-Y et al (2014) Microsoft COCO: Common Objects in Context. https://doi.org/10.48550/arxiv.1405.0312
2. ASAM e.V. (2021) ASAM OpenODD Concept Project, ASAM OpenODD Concept Project. https://www.asam.net/project-detail/asam-openodd. Accessed 11 Sep 2022
3. ASAM e.V. (2022) ASAM OpenSCENARIO: User Guide. https://www.asam.net/index.php?eID=dumpFile&t=f&f=4908&token=ae9d9b44ab9257e817072a653b5d5e98ee0babf8. Accessed 21 Aug 2022
4. ASAM e.V. (2020) ASAM OpenLABEL Concept Paper. https://www.asam.net/index.php?eID=dumpFile&t=f&f=3876&token=413e8c85031ae64cc35cf42d0768627514868b2f. Accessed 25 Sep 2022
5. Everingham M, Eslami SMA, van Gool L, Williams CKI, Winn J, Zisserman A (2015) The Pascal Visual Object Classes Challenge: A Retrospective. Int J Comput Vis 111(1):98–136. https://doi.org/10.1007/s11263-014-0733-5
6. Kaur J, Singh W (2022) Tools, techniques, datasets and application areas for object detection in an image: a review. Multimed Tools Appl. https://doi.org/10.1007/s11042-022-13153-y
7. Lin Q, Ye G, Wang J, Liu H (2022) RoboFlow: a Data-centric Workflow Management System for Developing AI-enhanced Robots, in Proceedings of the 5th Conference on Robot Learning, A. Faust, D. Hsu, and G. Neumann, Eds., in Proceedings of Machine Learning Research, vol. 164. PMLR, pp. 1789–1794. [Online]. Available: https://proceedings.mlr.press/v164/lin22c.html
8. Kar A et al (2021) Toronto Annotation Suite (TORAS). https://aidemos.cs.toronto.edu/toras. Accessed 05 Oct 2022
9. Torralba A, Russell BC, Yuen J (2010) LabelMe: Online Image Annotation and Applications. Proc IEEE 98(8):1467–1484. https://doi.org/10.1109/JPROC.2010.2050290
10. Vondrick C, Ramanan D, Patterson D (2010) Efficiently Scaling Up Video Annotation with Crowdsourced Marketplaces, in Computer Vision – ECCV 2010, K. Daniilidis, P. Maragos, and N. Paragios, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 610–623
11. Zimmer W, Rangesh A, Trivedi MM (2019) 3D BAT: A Semi-Automatic, Web-based 3D Annotation Toolbox for Full-Surround, Multi-Modal Data Streams, CoRR, vol. abs/1905.00525, [Online]. Available: http://arxiv.org/abs/1905.00525
12. Sekachev B et al (2020) opencv/cvat: v1.1.0. Zenodo. https://doi.org/10.5281/zenodo.4009388
13. Redmon J, Divvala S, Girshick R, Farhadi A (2015) You Only Look Once: Unified, Real-Time Object Detection. arXiv. https://doi.org/10.48550/ARXIV.1506.02640

14. Pokhrel S (2020) Image Data Labelling and Annotation - Everything you need to know. https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1. Accessed 27 Aug 2022
15. Tkachenko M, Malyuk M, Holmanyuk A, Liubimov N (2022) Label Studio: Data labeling software, Open source software available from https://github.com/heartexlabs/label-studio, https://github.com/heartexlabs/label-studio. Accessed 12 Aug 2022
16. Buechel M, Hinz G, Ruehl F, Schroth H, Györi C, Knoll A (2017) Ontology-Based Traffic Scene Modeling, Traffic Regulations Dependent Situational Awareness and Decision-Making for Automated Vehicles. https://doi.org/10.1109/IVS.2017.7995917
17. Doulamis ND (2010) Coupled multi-object tracking and labeling for vehicle trajectory estimation and matching. Multimed Tools Appl 50(1):173–198. https://doi.org/10.1007/s11042-009-0370-0
18. Mujika A et al (2019) Web-based Video-Assisted Point Cloud Annotation for ADAS validation. https://doi.org/10.1145/3329714.3338128
19. Three.js - JavaScript 3D Library. https://threejs.org/
20. The Khronos Group Inc., WebGL. https://www.khronos.org/api/webgl
21. Jain N, Bhansali A, Mehta D (2014) AngularJS: A modern MVC framework in JavaScript. J Global Res Comput Sci 5(12): 17–23. Accessed 3 May 2023. [Online]. Available: http://www.rroij.com/open-access/angularjs-a-modern-mvc-framework-in-javascript.pdf
22. (2020) Euro NCAP Assisted Driving Test and Assessment. Highway Assist Systems. Protocol v1.0. Accessed 04 Oct 2022. [Online]. Available: https://cdn.euroncap.com/media/58813/euro-ncap-ad-test-and-assessment-protocol-v10.pdf
23. Nieto M, Senderos O, Otaegui O (2021) Boosting AI applications: Labeling format for complex datasets. SoftwareX 13:100653. https://doi.org/10.1016/j.softx.2020.100653
24. (2021) labelGo, Github. https://github.com/cnyvfang/labelGo-Yolov5AutoLabelImg. Accessed 01 Oct 2022
25. Urbieta I, Nieto M, García M, Otaegui O (2021) Design and Implementation of an Ontology for Semantic Labeling and Testing: Automotive Global Ontology (AGO). Appl Sci 11(17). https://doi.org/10.3390/app11177782
26. Apellániz JL, García M, Aranjuelo N, Barandiaran J, Nieto M (2023) LiDAR-based curb detection for ground truth annotation in AD validation, in Proc. of IEEE Intelligent Transport Systems Conference, Bilbao. Just Accepted
27. Ortega JD et al (2020) DMD: A Large-Scale Multi-Modal Driver Monitoring Dataset for Attention and Alertness Analysis, CoRR, vol. abs/2008.12085, [Online]. Available: https://arxiv.org/abs/2008.12085
28. Vicomtech (2023) WebLabel, Github. https://github.com/Vicomtech/weblabel#use-case. Accessed 19 Jul 2023
29. Deutsches Zentrum für Schienenverkehrsforschung (DZSF) (2023) Open Sensor Data for Rail 2023. https://data.fid-move.de/dataset/osdar23#. Accessed 20 Jul 2023
30. R. Tagiew et al (2023) OSDaR23: Open Sensor Data for Rail 2023, [Online]. Available: http://arxiv.org/abs/2305.03001