SoftwareX 13 (2021) 100653

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Boosting AI applications: Labeling format for complex datasets

Marcos Nieto [*], Orti Senderos, Oihana Otaegui

Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Mikeletegi 57, 20009 San Sebastian, Spain

A R T I C L E I N F O

Article history:
Received 18 June 2020
Received in revised form 15 October 2020
Accepted 26 December 2020

Keywords:
Annotation
Dataset
Multi-sensor
Automotive

A B S T R A C T

Data labeling has become a major problem in industries aiming to create and use ground truth labels from massive multi-sensor archives to feed into Artificial Intelligence (AI) applications. Annotation of multi-sensor set-ups with multiple cameras and LIDAR is now particularly relevant for the automotive industry aiming to build Autonomous Driving (AD) functions. In this paper, we present the Video Content Description (VCD), as the first open source metadata structure and set of tools, able to structure annotations for such complex scenes, including unprecedented flexibility to label 2D and 3D objects, pixel-wise labels, actions, events, contexts, semantic relations, odometry, and calibration. Several example cases are reported to demonstrate the flexibility of the VCD.

Code metadata

Current code version	v4.2.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_220
Legal Code License	MIT License
Code versioning system used	git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	
If available Link to developer documentation/manual	For example: https://vcd.vicomtech.org/documentation/documentation-vcd
Support email for questions	info@vicomtech.org

1. Motivation and significance

In the era of Artificial Intelligence (AI), data has become the asset that enables training models that perform tasks such as detection, recognition, segmentation, or understanding. Supervised learning approaches, extended in scale by the newest Deep Learning (DL) paradigms, require the existence of massive labeled datasets which contain data accompanied by labels, so the AI can learn from the dataset how to generalize for future unlabeled incoming data. Sectors such as autonomous driving, surveillance and security, or manufacturing have suddenly turned their attention and investments into creating high-quality datasets from which obtain business advantage over competitors. The more complex the function to create, the richer the labels need to be. As a consequence, creating complex, rich and validated labels or annotations have become one of the key technical elements for success.

The usual AI pipeline includes the generation of data (logging), typically from multi-sensor set-ups such as instrumented vehicles, followed by data filtering and annotation using costly semi-manual approaches. As a result labeled data can be either dumped into databases for scene search, or to feed test procedures, AI training methods or simulation engines (see Fig. 1).

Annotation is the task of producing metadata that describes the content of data. For images or point clouds, the description may refer to the represented scene, via word tags or semantic descriptions, but also to the location or shape of objects. These labels are used to create ground truth for evaluation processes, and training sets for machine learning. The annotation task has been traditionally tackled by the scientific community, at small scale, to label the presence or location of objects of interest, such as pedestrians or cars, or pixel-wise descriptions of image regions. As a consequence, task-specific annotation formats have been proposed [1,2].

With the emergence of deep learning technologies, the number of image and point-cloud datasets has exploded in recent years [3–8]. Also, traditional industry sectors such as automotion or video-surveillance security are suddenly turning their attention to computer vision and machine learning, because of their

[*] Corresponding author.
E-mail address: mnieto@vicomtech.org (Marcos Nieto).

https://doi.org/10.1016/j.softx.2020.100653

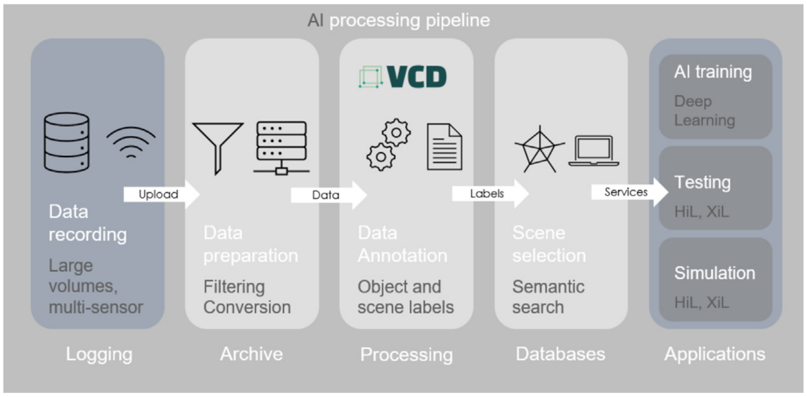

Fig. 1. AI data processing pipeline. Annotation is the central step where data is given meaning by producing labels to boost AI applications (training, testing, simulation).

Table 1
Classification of popular datasets according to the type of annotations (BB: Bounding box, PL-*N*: Pixel-level with *N* classes,), format, and application domain.

Dataset	Year	Domain	Annotation	Format
BEHAVE [9]	2007	CCTV	2D BB	XGTF
PASCAL VOC [2]	2010	General	2D BB	XML
TownCentre [10]	2011	CCTV	2D BB	CSV
KITTI obj./track. [3]	2012	AD	2D-3D BB	SSV
TRECVid [11]	2013	CCTV	Actions	XGTF
COCO [1]	2015	General	2D BB, PL-90	JSON
Cityscapes [4]	2016	AD	PL-35	PNG, JSON
Synthia [5]	2016	AD	PL-11	PNG, Text
Mapillary-Vistas [6]	2017	AD	PL-66	PNG M
BDD100K [12]	2017	AD	2D BB, PL	PNG, JSON
Youtube BB [13]	2017	General	2D BB	CSV
AD20K [8]	2018	General	PL-++	PNG, CSV
ApolloScape [14]	2018	AD	2D BB, PL	PNG, JSON
nuScenes [7]	2018	AD	3D BB	JSON
Lyft5[15]	2019	AD	3D BB	JSON
BLVD Dataset [16]	2019	AD	3D BB	TXT
Waymo [17]	2019	AD	2D-3D BB	Protobuf
Drive & Act [18]	2019	AD	Actions, 3D	CSV

increased capability to provide valuable data about the scenario in real-time. As a consequence, the investment in ground truth generation has also grown notably, creating a highly-valuable new type of data labeling companies, in order to build assets that can be used to train effective machine learning models that can automatically extract information from the data (e.g. detect pedestrians, cars, etc.). Also, annotations can take the form of ground truth content which can then be used to evaluate the performance of existing methods under specific situations (e.g. detection rates in urban or rainy environments).

In this work we describe the Video Content Description (VCD), an annotation format and set of tools that aim to cover the absence of a structured annotation model that enable labeling complex multi-sensor datasets for a variety of application domains (e.g. automotive, surveillance, medicine, sports).

2. Related work

This section overviews representative datasets with different annotation types (what is labeled) and then explores existing or commonly used annotation formats (how it is labeled).

2.1. Datasets and annotation types

In Table 1 we present an overview of representative open datasets from several domains, including the recently popular AD (Autonomous Driving) domain, specially interesting for its data and annotation complexities. As we can see, annotations frequently consist on image primitives, such as 2D bounding boxes, polygons [4,19] or pixel-level masks [4,5,8,12,19,20] (in the form of PNG images), or 3D elements (such as cuboids, etc.) [7, 17,21]. Actions or events are usually not labeled as time intervals, but as tags attached to each sample (image or point cloud scan) [18]. Some exceptions are datasets from the surveillance domain, which label actions also with certain semantic load [9, 11].

The analysis of this table illustrates the diversity of annotation types, according to purpose, metadata types, and data volume. We can conclude an annotation system would benefit in terms of interoperability and standardization from a single annotation language which aggregates all possible types of annotations, ranging from image-level bounding boxes and labels to temporal actions, and to relations between annotations. A proof of such need can be seen in the ASAM e.v. OpenLABEL standardization project,[1] started 2020, aiming to gather experts from the automotive industry to define an annotation model, format, and taxonomy for building AD functions.

2.2. Data annotation formats

As we have seen in Table 1, there is not a convention on the annotation format. Most datasets define a non-standard, task-specific annotation format, lacking extensibility, compatibility or real-time streaming.

CSV (Comma-Separated Value) and other folder-file annotation models are indeed efficient in terms of storage, but greatly lack the ability to extend or update annotations: any modification on the model implies rewriting all the parsing scripts, there is no direct way to merge different-level information (e.g. actions, relations, spatial information), and adding more details to annotations could simply be infeasible (e.g. from bounding box to polygons).

For pixel-level annotations, PNG images seem to have been selected as the best choice, given their ability to efficiently compress plain color information (e.g. 15–30 KB/image, CityScapes-fine dataset [4]). A simple RGB code table is enough to retrieve the label of a pixel. However, higher level analytics may require to query for shapes according to their geometry, size, or relative position with respect to other objects. This information is not contained in the PNG annotations and therefore an extra computational effort is required each time the information is accessed. Polygon-based annotations partially solve this problem at the cost

[1] https://www.asam.net/project-detail/scenario-storage-and-labelling.

Table 2
Comparison of annotation format features: * JSON, XML, Google Protobuf.

Format	Schema	Serialization	Mode	2D Obj.	3D Obj.	Nested attrib.	Frame interval	Ontology	Actions	Relations
Pascal-VOC [2]	No	XML	Image	Yes	No	No	No	No	No	No
COCO [1]	Yes	JSON	Image(s)	Yes	No	No	No	No	No	No
Viper-XML [11]	Yes	XML	Video, Images	Yes	No	No	Multiple	No	Yes	Yes
nuScenes [7]	Yes	DB	Multi-sensor	No	Yes	No	Single	No	No	No
VCD (our work)	Yes	Multiple*	Multi-sensor	Yes	Yes	Yes	Multiple	Yes	Yes	Yes

of producing larger annotation files (e.g. average 270 KB/image, CityScapes-fine dataset [4]).

JSON has started to be used widespread [4,7,22]. It offers similar properties as XML content, but with additional features that make it a good choice for the description of annotation. Namely, JSON focuses on numbers and text, is compatible with modern document-oriented databases, and very efficient parsers exist in different programming languages including Javascript, Python and also C++.

Recent datasets, because of being very large and complex, tend to contain folder structures with the scene data with sets of index text files that define timestamps, or other information about CAN signals, odometry or GPS data, while annotations are then described as CSV files [21]. There is not a centralized annotation file to retrieve information easily, nor a separation between annotations and data.

A complex format can be found at the nuScene dataset [7] (also used in the Lyft5 dataset), which include a complete hierarchy of labels, implemented as a set of JSON files with token entries that emulate a relational database. The format is designed to label 3D cuboids, and with timestamping information for different sensors. However, the format is not suited to label actions or semantic relations between elements easily. The tokens embedded within the annotations guarantee consistent linking between the different JSON files used, at the cost of adding a significant storage overhead.

As a conclusion, these datasets define custom, ad-hoc annotation formats, presented with development kits that simplify access and consumption, but are not suitable for extension, interoperation with other datasets (e.g. fusion or aggregation) nor longer-term operation on annotations (e.g. adding action labels for frame intervals in a later stage of annotation). In Table 2 we present a summary of the described features of the annotation formats used by at least two datasets of Table 1.

3. Software description

VCD is primarily defined as a structure of data fields (see schema illustrations in Figs. 3). For that purpose, we have used JSON-schema to define the structure.[2] In addition, we have created a VCD Python API to manage VCD payloads (i.e. load, modify, serialize) guaranteed to follow the schema, and a number of conversion tools. Additional APIs could be build in other languages (e.g. we have also implemented a Typescript version in the develop branch of the code repository), and interested users could as well write their own API with the only requirement of producing JSON content valid against the JSON schema.

`Elements`, `Frames` and `Streams` are the three main concepts of VCD (see Fig. 2). These concepts allows describing rich scenes, entirely (to serialize as files), or frame-by-frame (as messages), clearly separating static from dynamic information, structuring information per frames and per stream, such that annotations can be added in different stages, queried, and managed, and VCDs can be merged, updated and compared coherently.

[2] VCD structure can also described in other languages. We have also used Google Protocol Buffers to create a VCD proto file.

3.1. Elements

`Elements`[3] are the main containers of annotations. They can be `Objects`, `Actions`, `Events`, `Contexts` and `Relations`. Each of them have a unique identifier (UID), a name, and a semantic type, which may point to a concept in an ontology using a URI. `Elements` can contain static information but also dynamic, associated to specific frames or frame intervals.

`Object`: Contains information that can represent numerical magnitudes, such as 2D bounding boxes, 3D cuboids, polygons, points, or any array of numbers that represent arbitrary information (e.g. steer angle and speed of vehicle at a given instant).

`Action`: Temporal `Element` which represents a semantic situation, such as an activity or action of one or more `Objects`, described with a text string that can be the URI of an ontology item (e.g. `#PersonRunning`).

`Event`: A point in time that triggers some `Action` or appearance of an `Object` (e.g. `#PedestrianStartsWalking`), and that can be used to link actions in sequences.

`Context`: Any other non-spatial or non-temporal additional information of the scene that is not directly related to `Objects` or `Actions`, but that increases the semantic load of the annotations: e.g. `#Raining`, `#Night`, etc.

`Relation`: `Elements` can be connected via `Relations`. This type of annotation follows the RDF triplets definition,[4] which describes an `rdf:subject`, an `rdf:predicate` and an `rdf:object`. In VCD, RDF entries can have an arbitrary number of subjects and objects for the same predicate. Any VCD `Element` can be `rdf:subject` or `rdf:object`, depending on the `rdf:predicate`. This type of annotation is extremely useful to identify participants of scenes, their semantic implication, and for easy and fast retrieval of elements of a scene. For instance, it is possible to represent that a pedestrian or a group of cars are the participants of a certain action, such as "#PedestrianCrossing" or "#isOvertaking".

The observation of an `Object` from sensors may produce a variety of numerical descriptors. As opposed to many other description languages, which primarily focus on a single type (e.g. bounding boxes), VCD allows adding as many descriptors as desired, with a variety of types that enables the annotation of any type of information. For that purpose, an `Object` contains an arbitrary number of `ObjectData`.

`ObjectData` is the abstraction class of numerical entities such as bounding boxes, polygons, points, circles, and generic arrays of numbers. Each `ObjectData` can be named to add a semantic description inside the `Object` (e.g. bounding box "body", bounding box "head"), and also specify to which `Stream` it corresponds to.

Other `ObjectData` serve to generic purposes. For instance, `vec` can be used to define arrays of arbitrary length for magnitudes that represent some geometry or physical property of the `Object`. This flexible `ObjectData` concept is very useful to allocate space for magnitudes that can be computed or derived from

[3] In this manuscript we use `PascalCase` notation for concepts defined in the VCD schema for better visualization. The VCD JSON schema follows `snake_case` notation.

[4] https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/.

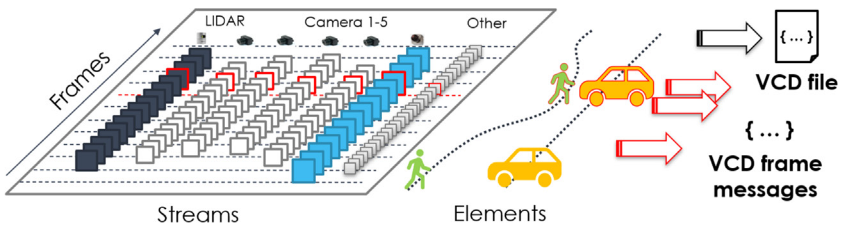

Fig. 2. VCD's `Stream`, `Frame` and `Element` concepts.

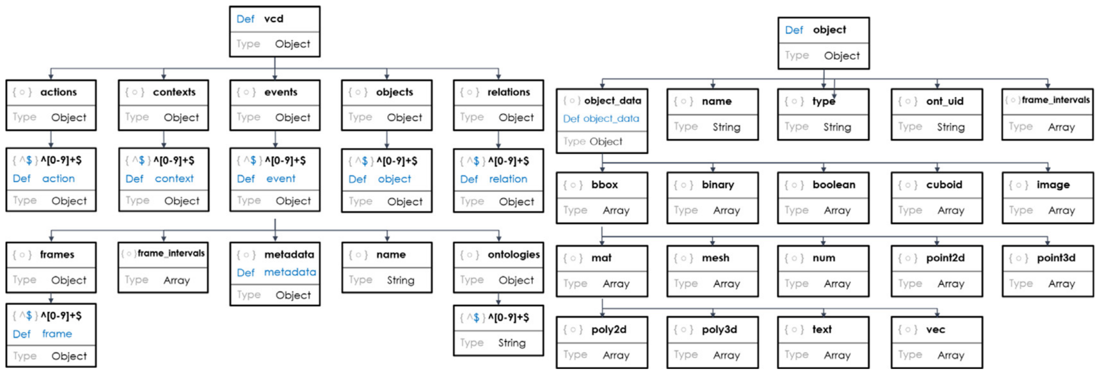

Fig. 3. VCD JSON schema: (left) `vcd`'s children nodes are the `Elements`, `Frames`, and metadata with `Streams`; (right) `Elements` like `Object` have name, type, frame intervals. `Object` also has arrays of `ObjectData`.

others, e.g. distance to other bounding boxes, color, depth, etc. The VCD API allows adding, removing, and merging `ObjectData` inside `Objects`.

Additionally, any `ObjectData` can contain attributes which are also other `ObjectData` of types `num`, `vec`, `text` or `bool`. This feature is very useful to add properties to annotations, in the form of some numbers (e.g. confidence levels), text (e.g. descriptions) or booleans (e.g. visible or not visible).

3.2. Format and tools

We have implemented the VCD as a JSON schema and as a Google Protocol Buffers proto file. A Python API has been created to manage VCD content, serialize it and to program converter tools. This implementation can be found at the github repository,[5] and VCD can be installed with `pip install vcd`. Documentation can also be found online.[6]

The repository includes a rich group of tests[7] which produce several VCD examples, for different use cases (multiple objects, multi-sensor, timestamping, actions and relations, ontologies, online operation, etc.).

4. Illustrative examples

In this section we present different example use cases where the VCD can be applied. A handful set of more than 40 VCD JSON files can also be found under folder /tests/etc in the code repository.

4.1. Bounding box and cuboid annotation

The most extended use case is the annotation of 2D bounding boxes (rectangles) in images [1,2,12,13] or 3D cuboids [7,21]. VCD provides support for `bbox` and `cuboid` as a derived classes of `ObjectData`.

For the tests, we have considered three different annotated datasets: (i) TownCentre [10], which includes 2 bounding boxes per person (body and head), (ii) KITTI object and tracking [3], for having 2D and 3D bounding boxes with nested attributes, and (iii) nuScenes [7], for its large volume of data and multi-sensor set-up (about 1.4 million 3D cuboids from 850 scenes, 20 s each).

With these four cases, we demonstrate that: (i) VCD is able to associate multiple bounding boxes to a single object, (ii) that it can compress the representation of labels using its proto schema, and (iii) that different types of `ObjectData` can be encapsulated inside `Objects`. We have created translators that read from the formats of the mentioned datasets, and have converted them into VCD JSON and Proto files. A comparison of the produced files is shown in Table 3.

As expected, the VCD JSON format is larger than the plain CSV and SSV formats, for the additional headers added to organize the content, while the VCD Proto files are smaller thanks to its binarization. In the case of nuScenes JSON format, we can observe both VCD JSON and VCD Proto result in smaller files. The volume of nuScenes in original JSON format is 2.92 MB/scene in average, while VCD JSON is 1.05 $\pm$ 0.30 MB, and VCD Proto is 0.70 $\pm$ 0.20 MB. Main reason is probably the overhead added by the tokens used in nuScene data structure to keep elements linked as a relational database.

Table 3
Sizes of annotations of different bbox and cuboid datasets compared to the size of VCD JSON and Proto files.

	Source (MB)	VCD JSON (MB)	VCD Proto (MB)
Town Center (CSV)	5.42	11.17	4.26
KITTI-object (SSV)	9.52	20.97	7.35
KITTI-tracking (SSV)	8.79	17.30	6.97
nuScenes (JSON)	2481.32	891.83	592.88

5 https://github.com/Vicomtech/video-content-description-VCD.
6 https://vcd.vicomtech.org/documentation/documentation-vcd.
7 https://github.com/Vicomtech/video-content-description-VCD/tree/master/tests.

Table 4
Average and standard deviation file size comparing source text and PNG files with VCD JSON and VCD JSON Compressed with lossless Freeman-chain code encoding. For the experiment we have randomly chosen 100 images from each dataset, repeating the test 10 times and aggregating all values. Same PC (8 GB RAM, Windows 10, Intel® Core™ i5-8250U @ 1.6 GHZ).

	Source text (KB)	PNG (KB)	VCD JSON (KB)	VCD JSON C. (KB)
Cityscapes coarse	38.33 ± 6.29	36.49 ± 4.70	6.74 ± 1.64	5.79 ± 2.19
Cityscapes fine	145.20 ± 23.39	39.56 ± 8.91	21.01 ± 7.94	13.63 ± 3.64
SYNTHIA	1407.28 ± 167.57	46.90 ± 12.12	103.90 ± 11.89	47.79 ± 5.76
Mapillary	–	59.23 ± 8.31	301.90 ± 44.21	42.64 ± 8.21

4.2. Pixel-level annotation

Pixel-wise annotations, often referred to as semantic annotations [20], determine the class (and instance number, in some cases) of each pixel of an image. These annotations are usually provided as PNG images, or as groups of polygons each defined by a set of points.

Amongst pixel-level datasets, we have selected Cityscapes [4], SYNTHIA [5] and Mapillary Vistas [6] to carry out a comparison between the two possible representations of pixel-level information: (i) PNG images, (ii) lists of polygons.

Table 4 presents the average size of the produced annotation files for the different source formats and the VCD JSON files (we have included a VCD JSON version where polygon coordinates are compressed using a variation of the Freeman chaincode lossless compression [23]). As we can see, the proposed format produces smaller files compared to the source text files due to the general compactness of VCD JSON. When compared with PNG raster content, VCD annotations achieve comparable sizes on the coarse Cityscapes annotations, while about double size for fine annotations and Mapillary (mainly because of the extreme detail of annotations and very large image resolution). The lossless compression of VCD JSON makes it even smaller than PNG in all cases.

One interesting property of VCD for storing semantic annotations is that it is possible to very efficiently retrieve information out of the polygons, and therefore create light-weight, class-specific VCDs for certain purposes. In contrast, PNG images need to be processed with image analysis tools each time to retrieve such information. As an example, reading and processing Mapillary-Vistas class and instance PNG images (4032×3024 pixels), using OpenCV's `findContours` Python function takes an average of 2.64 s in a regular PC. Reading and parsing the equivalent VCD JSON file and retrieving all polygons takes 24.93 ms when polygons are described in absolute coordinates, and 167.55 ms seconds when using a Freeman-chaincode lossless compression algorithm. As a conclusion, using VCD JSON is simultaneously better than PNG for storage and post-processing computation needs.

5. Impact and sustainability

The automotive industry is rushing to create Autonomous Driving (AD) functions using multi-sensor solutions. Labeling such complex datasets is becoming an additional and unexpected challenge, which then imposes the need for a robust, flexible and efficient annotation format. Some of the requirements for such format are:

- Annotation of multi-camera and multi-sensor recordings
- Management of timestamps for events synchronization
- Online annotation for in-vehicle processing units
- Linked to ontology-based semantics
- Extensible and scalable (good search capabilities)

To the best of our knowledge, no annotation model exists that tackles all these requirements, while VCD has been devised to satisfy them and provide practical tools related to labeling tasks.

Compared to the approaches described in Section 2, the proposed VCD format is way richer and flexible, as it can host content from multiple sensors, with syncing information, and describe both static and dynamic properties of objects, including actions as temporal entities, and semantically rich relations between elements. In addition, VCD uses linked data entries to ontologies that define the taxonomy of classes, possible attributes, and relations between classes. The VCD structure also permits labeling an entire scene in a single JSON file, using integer unique identifiers to rapidly access information about elements, frames and other entities.

VCD can label semantic concepts, going beyond traditional object-level annotation concepts, adding the possibility to label actions, attributes, and relations (in the form of RDF triplets).

In addition, VCD structure hosts nicely multi-sensor set-up properties, such as intrinsic parameters, extrinsic poses, odometry values, and timestamping.

Since VCD has an online nature, it can be used as open language to interchange messages between components of a multi-sensor application (e.g. within RTMaps or ROS frameworks), allowing customized concepts to be interchanged, including objects, action, relations. VCD has been used as labeling and messaging format in H2020 projects such as Cloud-LSVA, VI-DAS, etc., and is currently under usage by private labeling companies of the automotive sector.

VCD has been open sourced recently, aiming to reach a wider user group compared to the labeling companies at the automotive sector. VCD principles are being studied and proposed to shape the recently opened standardization project OpenLABEL from ASAM e.V.

6. Conclusions

In this paper we have presented the Video Content Description (VCD), as an annotation format suitable to create image and video annotations as spatial and temporal `Elements`, such as `Objects`, `Events`, `Actions`, `Contexts`, and their semantic relationships as `Relations`.

The VCD is specially designed to work for large scale video annotation, in sectors like Autonomous Driving or surveillance, in which large volumes of annotations are required to create ground truth and training sets for machine learning algorithms. Also, the VCD structure has been carefully crafted to support complex annotation types, such as nested attributes, multi-stream data, and compact polygon representations, which virtually support any kind of annotation use case.

A number of use cases have been presented which demonstrate the different types of annotations the VCD can handle. Comparison with existing annotations on known datasets have been provided, demonstrating that VCD content can be created from existing annotations, and that the produced annotations are equal or more compact than the source annotations. VCD JSON schema and Python API have been open sourced and can be used under MIT license.

CRediT authorship contribution statement

Marcos Nieto: Conceptualization, Methodology, Software, Writing - original draft, Formal analysis. **Orti Senderos:** Data curation, Visualization, Validation. **Oihana Otaegui:** Funding acquisition, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

This work was partially funded by the European Commission under Horizon 2020 research and innovation programme (grant agreement no. 688099, project Cloud-LSVA).

References

[1] Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft COCO: Common objects in context. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T, editors. Computer vision – ECCV 2014. Springer International Publishing; 2014, p. 740–55.
[2] Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A. The pascal visual object classes (VOC) challenge. Int J Comput Vis 2010;88(2):303–38.
[3] Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In: Conference on computer vision and pattern recognition (CVPR). 2012.
[4] Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, et al. The cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE conference on computer vision and pattern recognition (CVPR). 2016.
[5] Ros G, Sellart L, Materzynska J, Vazquez D, Lopez A. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: Proc. of the IEEE conference on computer vision and pattern recognition (CVPR). 2016.
[6] Neuhold G, Ollmann T, Rota Bulò S, Kontschieder P. The mapillary vistas dataset for semantic understanding of street scenes. In: International conference on computer vision (ICCV). 2017, https://www.mapillary.com/dataset/vistas.
[7] Caesar H, Beijbom O, Bankiti V, Lang A, Vora S, Dicle C. Nuscenes. 2018.
[8] Zhou B, Zhao H, Puig X, Fidler S, Barriuso A, Torralba A. Scene parsing through ADE20k dataset. 2017.
[9] Blunsden S, Fisher RB. The BEHAVE video dataset: ground truthed video for multi-person. 2009.
[10] Benfold B, Reid I. Stable multi-target tracking in real-time surveillance video. In: CVPR. 2011, p. 3457–64.
[11] Over P, Awad G, Michel M, Fiscus J, Sanders G, Kraaij W, et al. TRECVID 2013 - an overview of the goals, tasks, data evaluation mechanisms and metrics. In: Proceedings of TRECVID 2013, NIST, USA. 2013.
[12] Yu F, Xian W, Chen Y, Liu F, Liao M, Madhavan V, et al. BDD100k: A diverse driving video database with scalable annotation tooling. 2017.
[13] Real E, Shlens J, Mazzocchi S, Pan X, Vanhoucke V. YouTube-BoundingBoxes: A large high-precision human-annotated data set for object detection in video. 2017, arXiv:1702.00824.
[14] Huang X, Cheng X, Geng Q, Cao B, Zhou D, Wang P, et al. The apolloscape dataset for autonomous driving. 2018, CoRR abs/1803.06184.
[15] Houston J, Zuidhof G, Bergamini L, Ye Y, Jain A, Omari S, et al. One thousand and one hours: Self-driving motion prediction dataset. 2020, arXiv:2006.14480.
[16] Xue J, Fang J, Li T, Zhang B, Zhang P, Ye Z, et al. BLVD: Building A large-scale 5D semantics benchmark for autonomous driving. In: Proc. international conference on robotics and automation, p. 6685-6691, 2019.
[17] Sun P, Kretzschmar H, Dotiwalla X, Chouard A, Patnaik V, Tsui P, et al. Scalability in perception for autonomous driving: Waymo open dataset. 2019, arXiv:1912.04838.
[18] Martin M, Roitberg A, Haurilet M, Horne M, Reiß S, Voit M, et al. Drive&act: A multi-modal dataset for fine-grained driver behavior recognition in autonomous vehicles. In: The IEEE international conference on computer vision (ICCV). 2019.
[19] Russell BC, Torralba A, Murphy KP, Freeman WT. Labelme: A database and web-based tool for image annotation. Int J Comput Vision 2008;77(1–3):157–73.
[20] Ošep A, Hermans A, Engelmann F, Klostermann D, Mathias M, Leibe B. Multi-scale object candidates for generic object tracking in street scenes. In: ICRA. 2016.
[21] Patil A, Malla S, Gang H, Chen Y-T. The h3D dataset for full-surround 3D multi-object detection and tracking in crowded urban scenes. In: International conference on robotics and automation. 2019.
[22] Li X, Flohr F, Yang Y, Xiong H, Braun M, Pan S, et al. A new benchmark for vision-based cyclist detection. In: IEEE intelligent vehicles symposium. 2016, p. 1028–33.
[23] Freeman H. On the encoding of arbitrary geometric configurations. IRE Trans on Elec Comput 1961;10:260–8.