vicomtech

your R&D partner
for smart digital solutions

Semantic labelling for AD development & validation

Dr. Oihana Otaegui – Head of Department ITS & Engineering
Dr. Marcos Nieto – Principal Researcher

vicomtech

ABOUT US

Non-for-Profit Private Research Institution in Artificial Intelligence, Visual Computing & Interaction

Applied Research & Technology Transfer to the Industry

www. vicomtech.org

+160
RESEARCHERS

+60
Ph.D

24%
H2020 & INTERNATIONAL

46%
INDUSTRIAL R&D

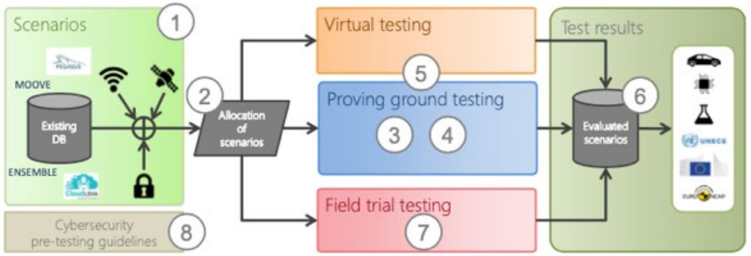
Semantic labelling for AD development & validation

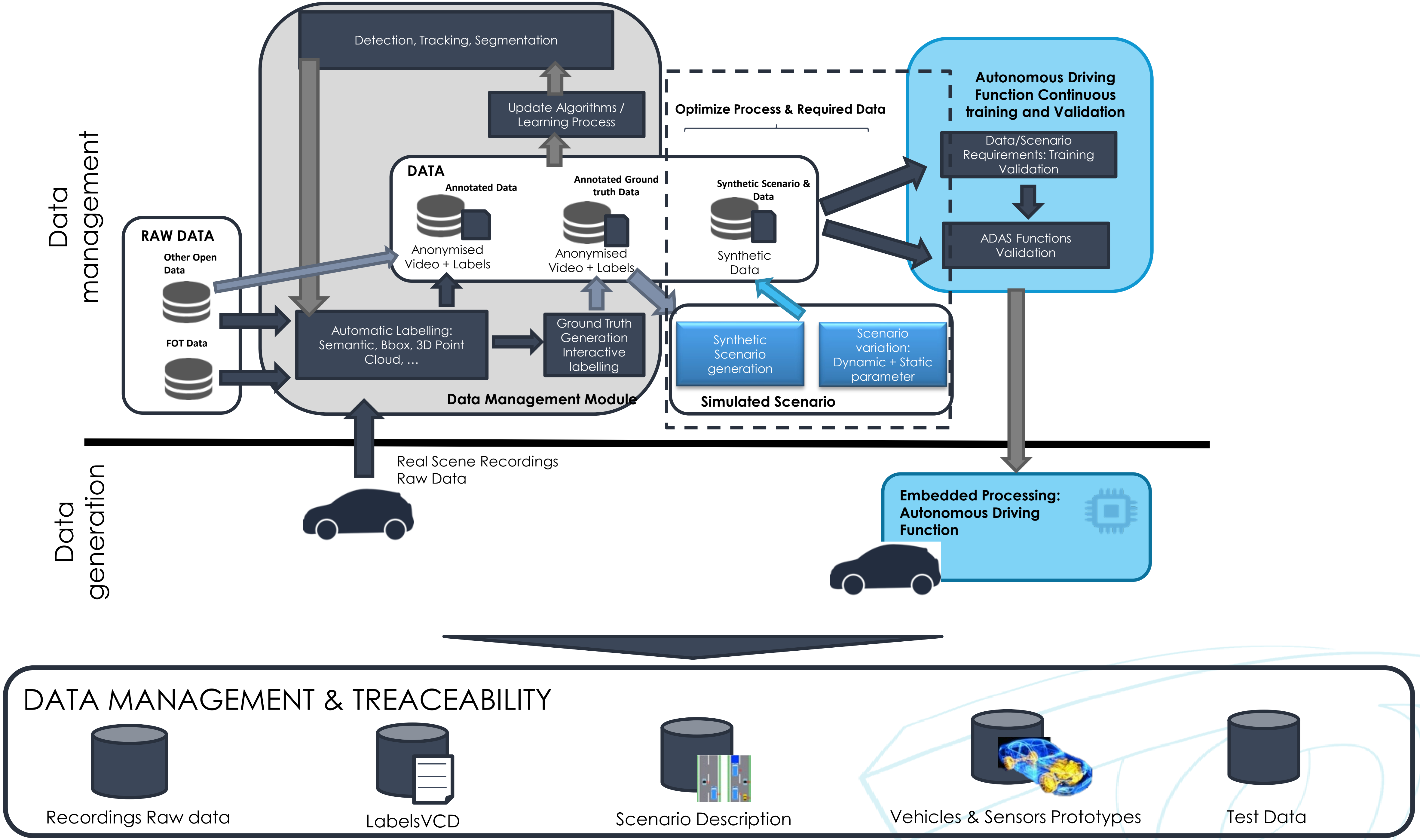
Context

Cloud-LSVA will create **Big Data technologies** to address the open problem of a lack of software tools, and hardware platforms, to **annotate very large-scale** datasets in the context of **ADAS** and **Digital Cartography to:**

- **Create large training datasets** of visual samples for training models using supervised learning to be used in vision-based detection.
- **Generate ground truth scene descriptions** based on objects (spatio-temporal) and events (temporal logic actions) to evaluate the performance of algorithms and systems that aim to detect or provide such descriptions.

HEADSTART will define **testing and validation procedures** of CAD functions including its key enabling technologies (i.e. communication, cyber-security, positioning) by cross-linking of all test instances such as **simulation**, **proving ground** and **real world field tests** to validate safety and security performance according to the needs of key user groups (technology developers, consumer testing and type approval).





Data management

RAW DATA

Other Open Data

FOT Data

Data Management Module

Detection, Tracking, Segmentation

Update Algorithms / Learning Process

DATA

Annotated Data

Anonymised Video + Labels

Annotated Ground truth Data

Anonymised Video + Labels

Automatic Labelling: Semantic, Bbox, 3D Point Cloud, …

Ground Truth Generation Interactive labelling

Optimize Process & Required Data

Synthetic Scenario & Data

Synthetic Data

Simulated Scenario

Synthetic Scenario generation

Scenario variation: Dynamic + Static parameter

Autonomous Driving Function Continuous training and Validation

Data/Scenario Requirements: Training Validation

ADAS Functions Validation

Data generation

Real Scene Recordings Raw Data

Embedded Processing: Autonomous Driving Function

DATA MANAGEMENT & TREEACEABILITY

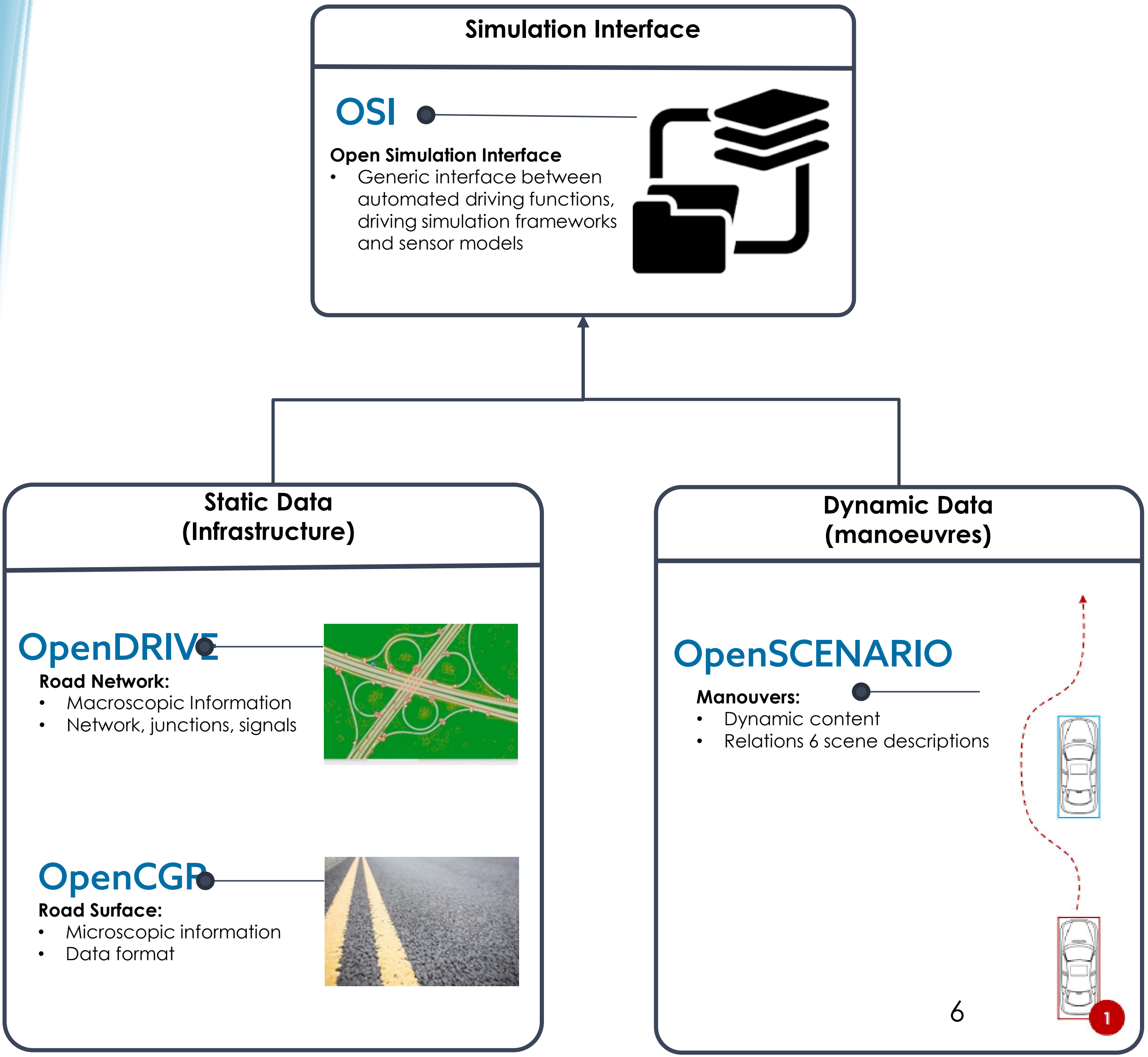
Recordings Raw data

LabelsVCD

Scenario Description

Vehicles & Sensors Prototypes

Test Data



Simulation Interface

OSI

Open Simulation Interface
- Generic interface between automated driving functions, driving simulation frameworks and sensor models

Static Data (Infrastructure)

OpenDRIVE

Road Network:
- Macroscopic Information
- Network, junctions, signals

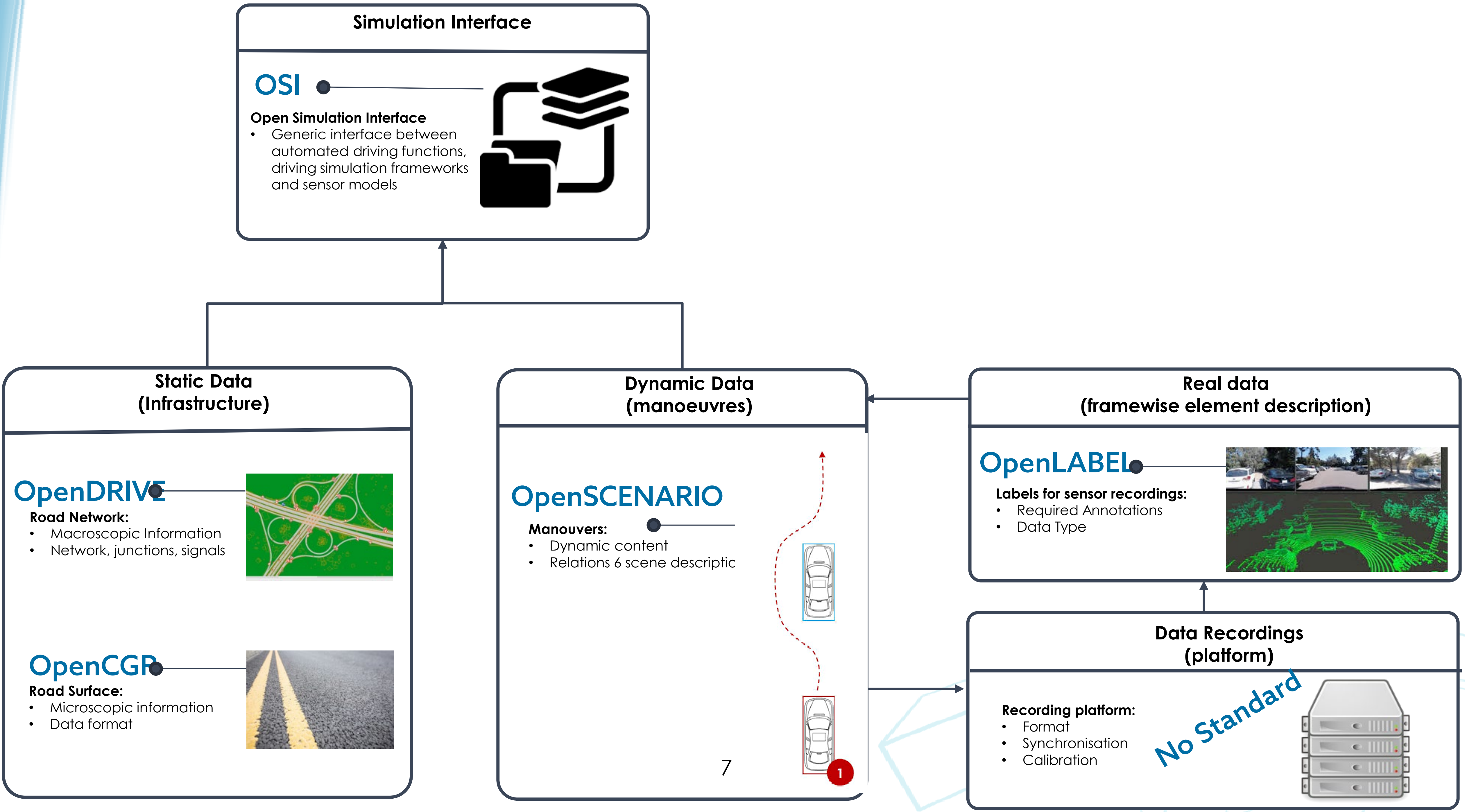
OpenCGR

Road Surface:
- Microscopic information
- Data format

Dynamic Data (manoeuvres)

OpenSCENARIO

Manouvers:
- Dynamic content
- Relations 6 scene descriptions



Simulation Interface

OSI

Open Simulation Interface
- Generic interface between automated driving functions, driving simulation frameworks and sensor models

Static Data (Infrastructure)

OpenDRIVE

Road Network:
- Macroscopic Information
- Network, junctions, signals

OpenCGR

Road Surface:
- Microscopic information
- Data format

Dynamic Data (manoeuvres)

OpenSCENARIO

Manouvers:
- Dynamic content
- Relations 6 scene descriptic

Real data (framewise element description)

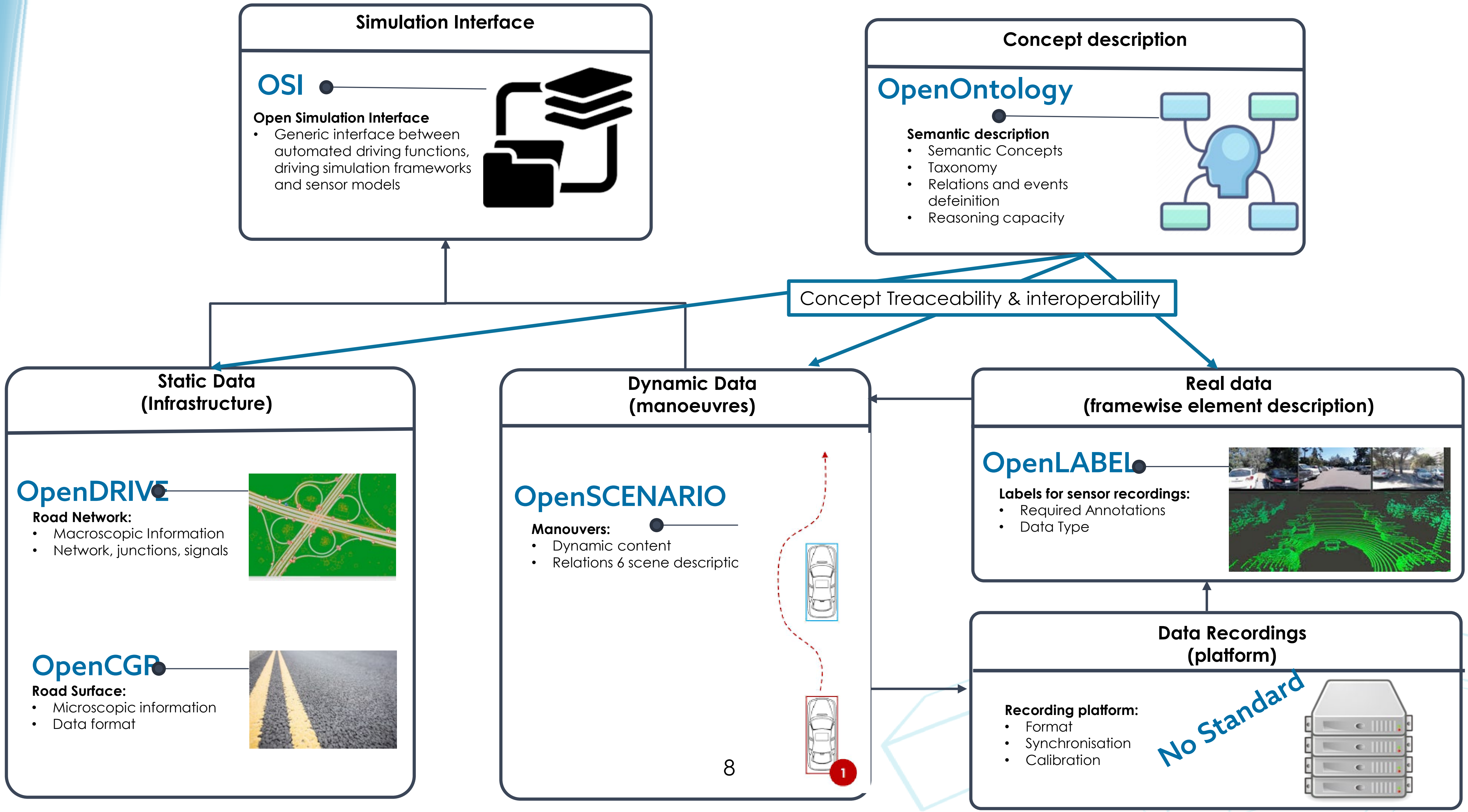
OpenLABEL

Labels for sensor recordings:
- Required Annotations
- Data Type

Data Recordings (platform)

Recording platform:
- Format
- Synchronisation
- Calibration

No Standard



Simulation Interface

OSI

Open Simulation Interface
- Generic interface between automated driving functions, driving simulation frameworks and sensor models

Concept description

OpenOntology

Semantic description
- Semantic Concepts
- Taxonomy
- Relations and events defeinition
- Reasoning capacity

Concept Treaceability & interoperability

Static Data (Infrastructure)

OpenDRIVE

Road Network:
- Macroscopic Information
- Network, junctions, signals

OpenCGR

Road Surface:
- Microscopic information
- Data format

Dynamic Data (manoeuvres)

OpenSCENARIO

Manouvers:
- Dynamic content
- Relations 6 scene descriptic

Real data (framewise element description)

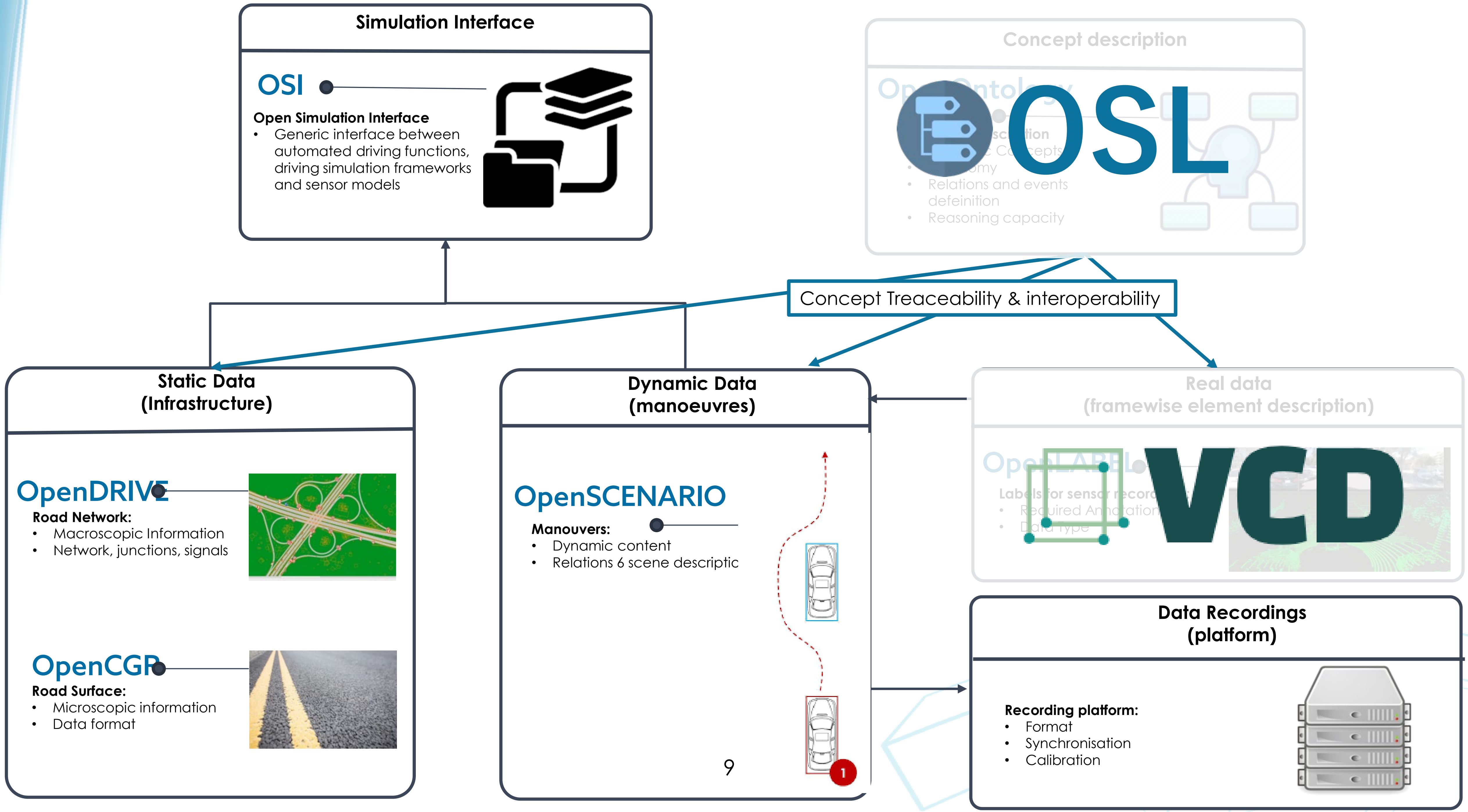
OpenLABEL

Labels for sensor recordings:
- Required Annotations
- Data Type

Data Recordings (platform)

Recording platform:
- Format
- Synchronisation
- Calibration

No Standard



Simulation Interface

OSI

Open Simulation Interface
- Generic interface between automated driving functions, driving simulation frameworks and sensor models

Concept description

OpenOntology

OSL

Description
- Basic Concepts
- Taxonomy
- Relations and events defeinition
- Reasoning capacity

Concept Treaceability & interoperability

Static Data (Infrastructure)

OpenDRIVE

Road Network:
- Macroscopic Information
- Network, junctions, signals

OpenCGR

Road Surface:
- Microscopic information
- Data format

Dynamic Data (manoeuvres)

OpenSCENARIO

Manouvers:
- Dynamic content
- Relations 6 scene descriptic

Real data (framewise element description)

OpenLABEL

VCD

Labels for sensor records
- Required Annotation
- Data Type

Data Recordings (platform)

Recording platform:
- Format
- Synchronisation
- Calibration

Labelling format

VCD - Video Content Description

vicomtech

VCD Video Content Description

- **Description:** syntax and tools for multi-purpose data labelling

- **Versions**
  - **VCD 1.0** – 2013
  - **VCD 2.0** – 2014
  - **VCD 3.0** – 2018
  - **VCD 4.0** – 2020

- **Used by**
  - **Internal developments** at Vicomtech
  - **European Projects** H2020
  - Industrial projects with Vicomtech's **costumers**

2013

- The idea was born.
- First steps on the creation of VCD were taken.

Viulib® turning vision into reality

2014

- Integrated into Viulib Library (module viulib_evaluation).
- Element-wise and Frame-wise models.
- XML and JSON serialization via ASL library.

2018

- Independent C++ library .
- Element-wise and Frame-wise modes .
- Multi-sensor support .
- JSON serialization via ASL library.
- Pixel-wise loss-less compression modes.
- Comparison routines.

2020

- Python library
- Element and Frame-wise mode simultaneously
- Multi-sensor and multi-interval
- Native Python JSON serialization

AUTOPILOT

CloudLSVA
Large Scale Video Analysis

Viulib®
turning vision into reality

EWISA

SmaCS

HEADSTART

VI-DAS

P-REACT

begirale

inLane
Lane Navigation Technology

graphicsmedia.net

VCD Video Content Description

VCD syntax

- Definition of **Elements**
  - (Actions, Events, Objects, Contexts, Relations)
- **Metadata** structures
- **Frame-wise** & Element-wise
- **Object data** primitives
  - name, type, uids
- Links to **ontologies**

Documentation

Schemas (JSON, Proto, TS)

VCD libraries

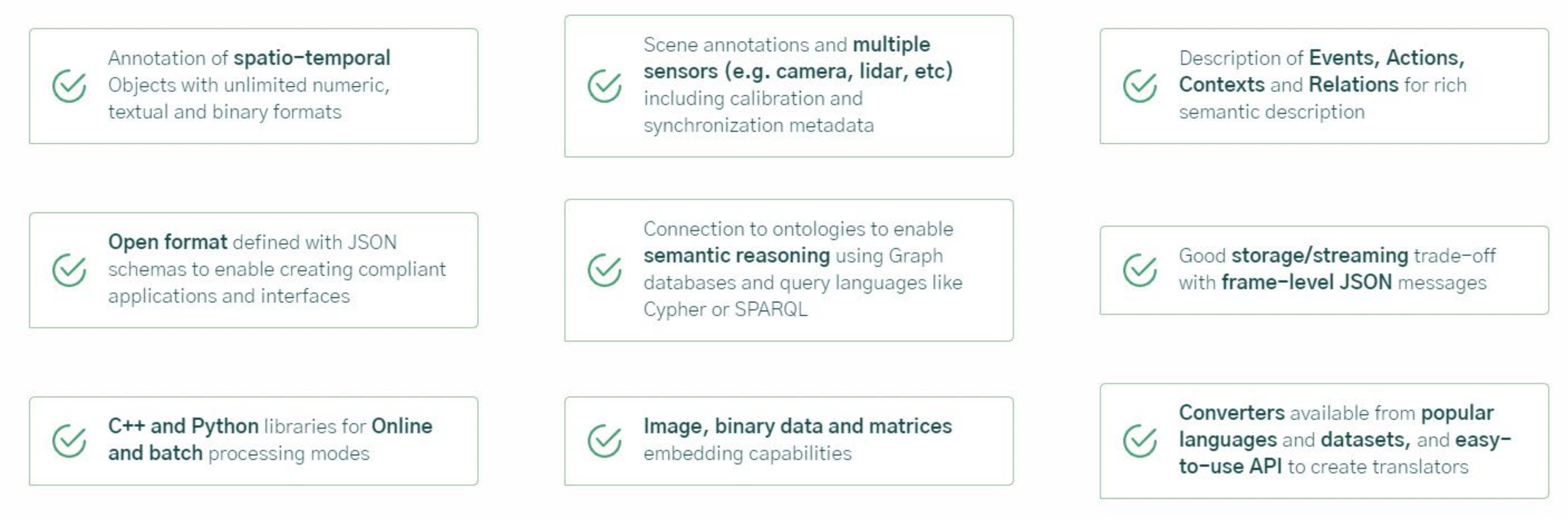
- Create and **manage** VCD content within apps
- **Search** and process metadata
- Load/Save **files**
- **Converters** from other formats

SW libraries

C++, Python

VCD Video Content Description

- **Features**
  - VCD's structure hosts **Objects, Actions, Events, Contexts, Relations** defined for **multiple-streams**, at multiple **frame-intervals**, at frame-level or static-level, and **serializable** for storage and communications



✓ Annotation of **spatio-temporal** Objects with unlimited numeric, textual and binary formats

✓ Scene annotations and **multiple sensors (e.g. camera, lidar, etc)** including calibration and synchronization metadata

✓ Description of **Events, Actions, Contexts** and **Relations** for rich semantic description

✓ **Open format** defined with JSON schemas to enable creating compliant applications and interfaces

✓ Connection to ontologies to enable **semantic reasoning** using Graph databases and query languages like Cypher or SPARQL

✓ Good **storage/streaming** trade-off with **frame-level JSON** messages

✓ **C++ and Python** libraries for **Online and batch** processing modes

✓ **Image, binary data and matrices** embedding capabilities

✓ **Converters** available from **popular languages** and **datasets,** and **easy-to-use API** to create translators

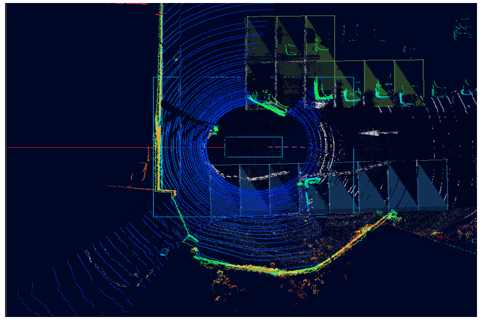
vicomtech

VCD Video Content Description

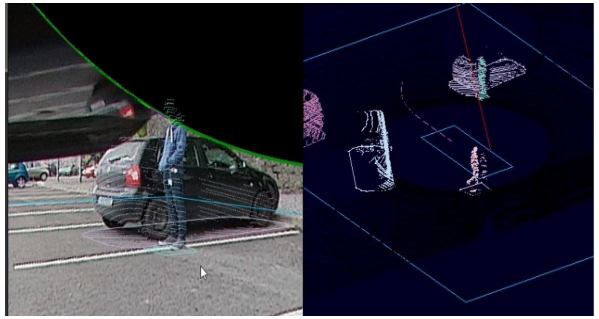
- **Metadata** which describe the content of a **scene** in an structured manner

- In many cases, **metadata needs to be attached to data series:** videos, lidar, static images, etc.

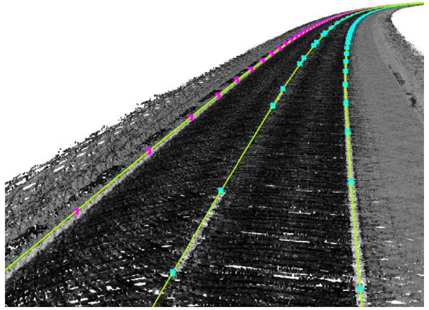
- **Annotations** need to cover:
  - Object descriptions
  - Spatio-temporal entities
  - Synchronization and timestamps
  - Sensor calibration
  - Numerical ranges
  - Actions and events
  - Time intervals
  - Relations between elements
  - Semantic concepts



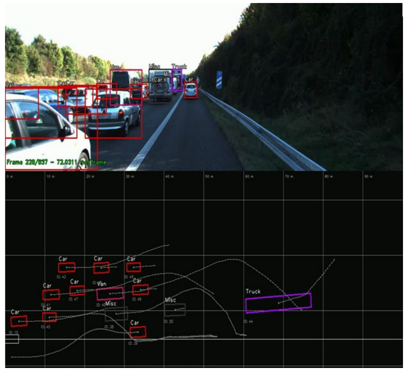
3D Parking slots



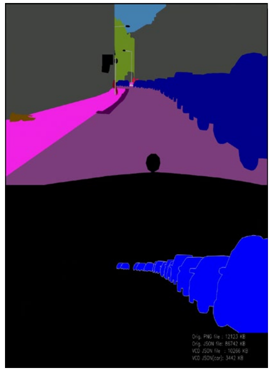
3D objects



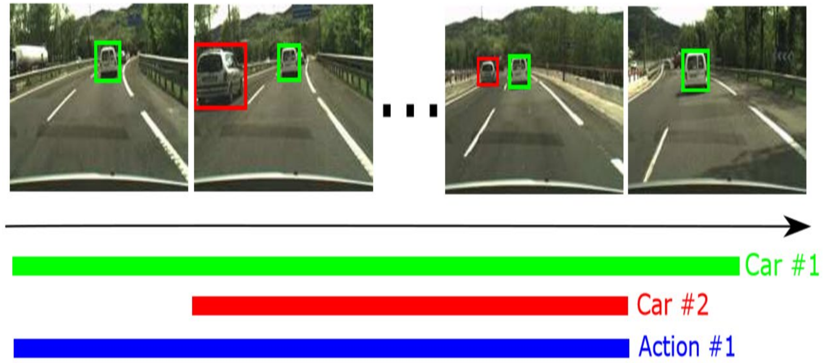
3D lane markings



2D-3D objects



2D segmentation



Maneouvres (actions)

graphicsmedia.net

- **VCD structure**
  - **Elements**
    - containers of description of the scene
  - **Streams**
    - description of data sequences/measurements/observations of the scene
  - **Frames**
    - time-wise samples containing elements and streams information

VCD Video Content Description

vicomtech

- **Elements** = {**Objects**, **Actions**, **Events**, **Contexts**, **Relations**}

Object

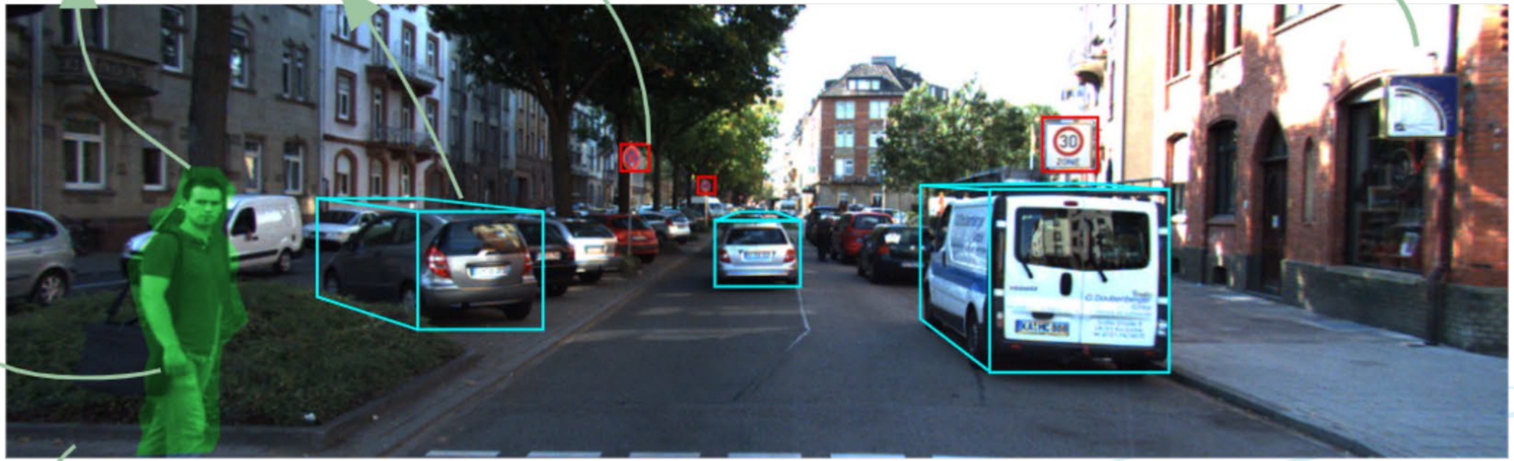
Person, signal, car or any object with spatial description (e.g. bounding box) and sensor ID from which it is seen

Context

This is an urban scene, it is **sunny** it is a sequence from an onboard camera

Action

The period of time where an action happens:
looking at ego-vehicle
crossing



Event

The **moment** in which the **person** starts **crossing** the road

Relation

The object person is the actor of the action, and the event triggers the action.
A **person crosses** the road when is **sunny**

graphicsmedia.net

VCD Video Content Description

- **Elements** = {**Objects**, **Actions**, **Events**, **Contexts**, **Relations**}

Objects, and **Contexts** can be static (no time-information)

e.g.: (pseudo-code expressions)

Object "Mike" is a "Person"
Mike's age is 38
Mike's address is "Calle Mayor 12, Madrid"

Static attributes

Object "TrafficSign1" is a "TrafficSign"
TrafficSign1's position is (43.302276, -2.002997)
TrafficSign1's class is "Stop"
TrafficSign1's visibility is "Poor"
TrafficSign1's position's labeled by "Annotator1"
TrafficSign1's position's interpolated is "True"

Nested attributes

Context "Weather" is "Sunny"
Context "Road" is "Urban"
…

VCD Video Content Description

- **Elements** = {**Objects**, **Actions**, **Events**, **Contexts**, **Relations**}

Objects, **Actions**, **Events**, **Contexts** can be dynamic (defined with time information)

e.g.:
Mike’s position is (2.6, 5.3, 0.0) at frame = 14
Mike’s position’s reference is “ISO8855, rear-axle”
Mike’s speed is (2.5) at frame = 14
…
Mike’s position is (2.5, 5.2, 0.0) at frame = 15
…

Dynamic attributes

Action “MikeCrossing” is a “Crossing”
Crossing1 happens during frames (15, 270)
Action “MikeLooking” is a “Looking”
MikeLooking happens during frames [(50, 200), (210, 250)]

Frame intervals

Event “MikeStartsCrossing” is a “StartCrossing”
MikeStartsCrossing happens at frame (14)
…

Relation “r1” means “Mike” “isActorOf” “MikeCrossing”
Relation “r2” means “Mike” “isNear” “TrafficSign1”
…

Semantic relations

VCD Video Content Description

- **Streams**
  - Metadata: sensor type, properties, calibration, etc.
  - E.g. “front-camera”, “rear-camera”, “top-lidar”, “left-lidar”, etc.
  - Elements can be defined at Stream level
    - E.g. Other vehicle’s is at bbox (125, 54, 66, 50) **for Stream “front-camera” at frame = 0**
    - E.g. Other vehicle’s is at cuboid (4.60, 12.01, …) **for Stream “top-lidar” at frame = 0**
    - …

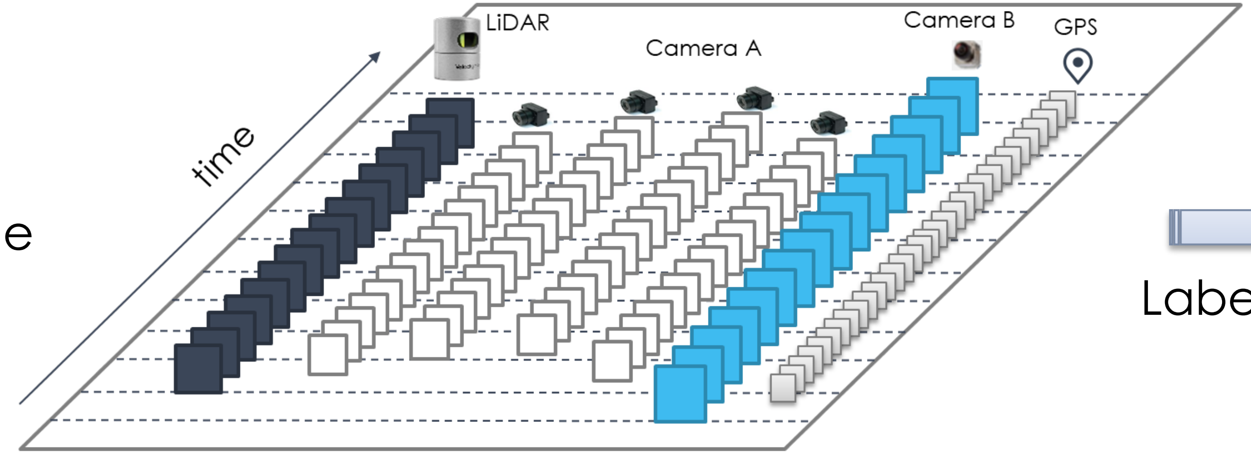
- **Frames**
  - VCD Master frame sequence [0, N]
  - Frames { “0”: {…}, “1”: {…}, … }
  - Stream properties
    - “front-camera”’s **frame idx = 2** at (master) frame = 0
    - “rear-camera”’s **frame idx = 2** at (master) frame = 1
    - “front-camera”’s **frame idx = 3** at (master) frame = 0
    - …
    - “front-camera”’s **timestamp = “2007-11-03T13:18:05.000” at (master) frame = 52**

Inter-stream frame-sync

Inter-stream timestamping

VCD Video Content Description

Batch / Offline

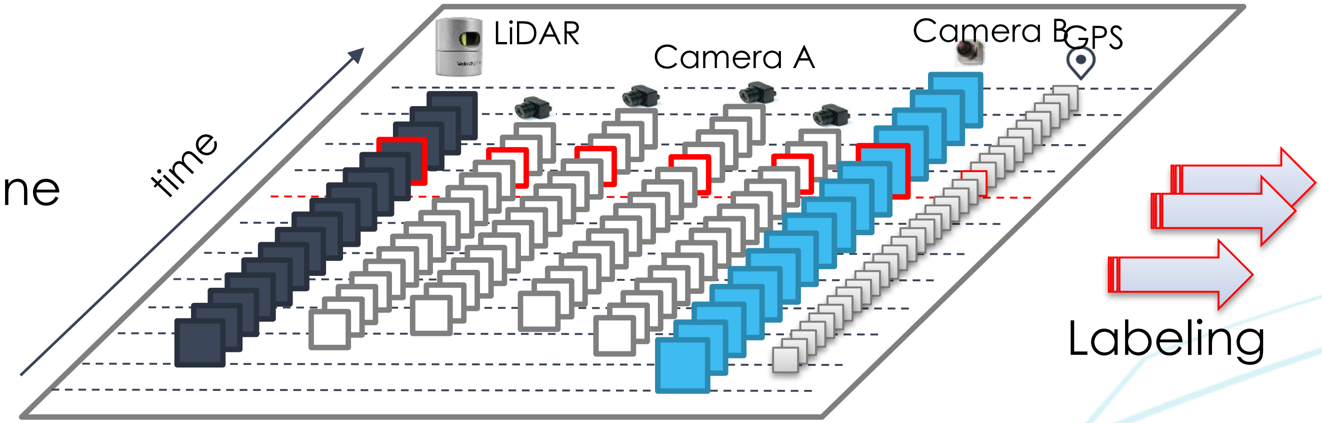


Recording - Data

Labeling

VCD file
-To be stored for archive/processing

Frames / Online



Life/Stream data

Labeling

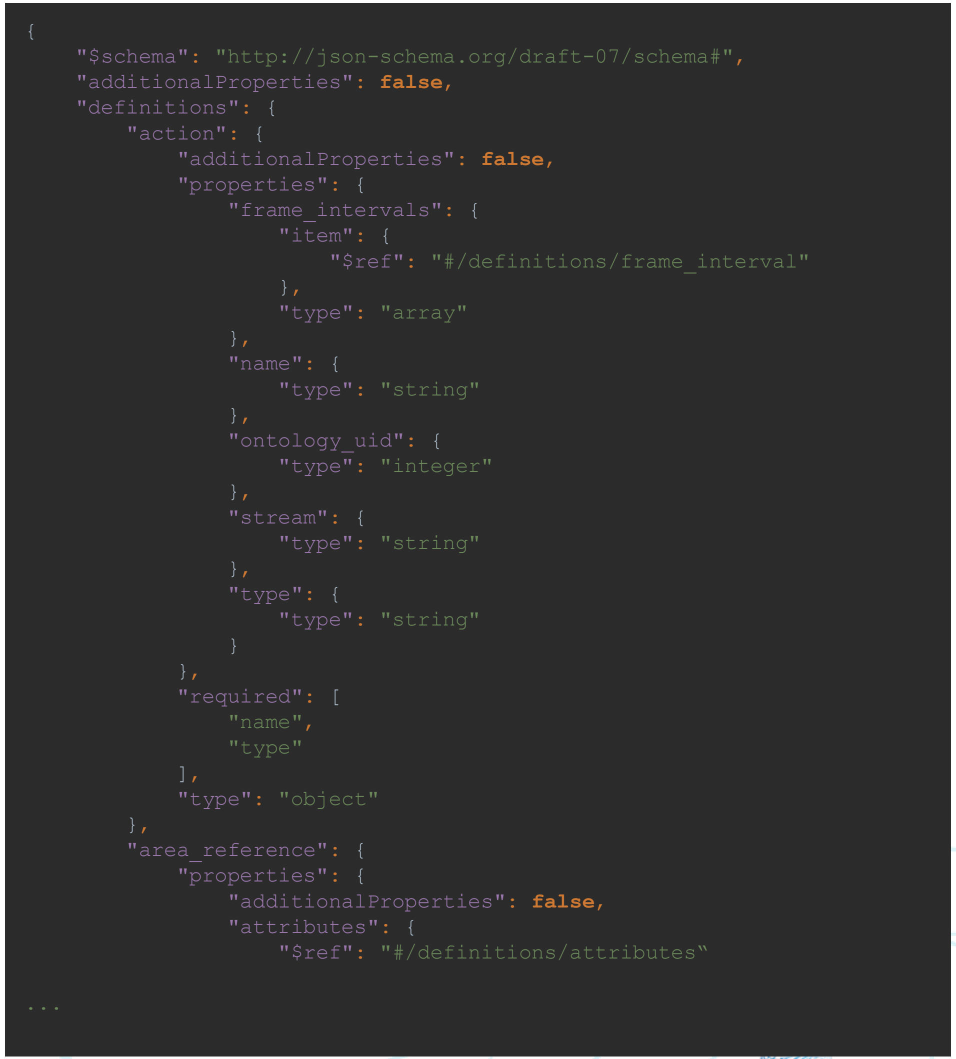
VCD messages
-To be sent via comms

VCD Video Content Description

- **VCD defined with structured schemas**
  - JSON schema
  - From JSON Schema to other formats
    - Google Protobuf
    - TypeScript

  - Formal description of structure
  - Validates content
  - Version control (current v4.0.0)

vcd_schema_json-v4.0.0.json



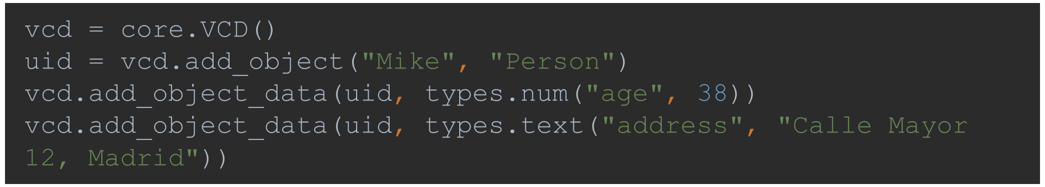
```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "additionalProperties": false,
    "definitions": {
        "action": {
            "additionalProperties": false,
            "properties": {
                "frame_intervals": {
                    "item": {
                        "$ref": "#/definitions/frame_interval"
                    },
                    "type": "array"
                },
                "name": {
                    "type": "string"
                },
                "ontology_uid": {
                    "type": "integer"
                },
                "stream": {
                    "type": "string"
                },
                "type": {
                    "type": "string"
                }
            },
            "required": [
                "name",
                "type"
            ],
            "type": "object"
        },
        "area_reference": {
            "properties": {
                "additionalProperties": false,
                "attributes": {
                    "$ref": "#/definitions/attributes‟

...
```

VCD Video Content Description

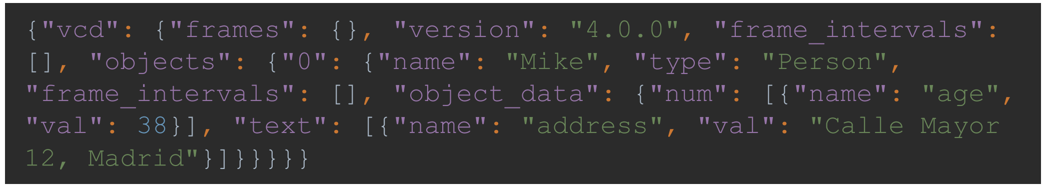
Serializable JSON/Proto

Object “Mike” is a “Person”
Mike’s age is 38
Mike’s address is “Calle Mayor 12, Madrid”



```
vcd = core.VCD()
uid = vcd.add_object("Mike", "Person")
vcd.add_object_data(uid, types.num("age", 38))
vcd.add_object_data(uid, types.text("address", "Calle Mayor
12, Madrid"))
```

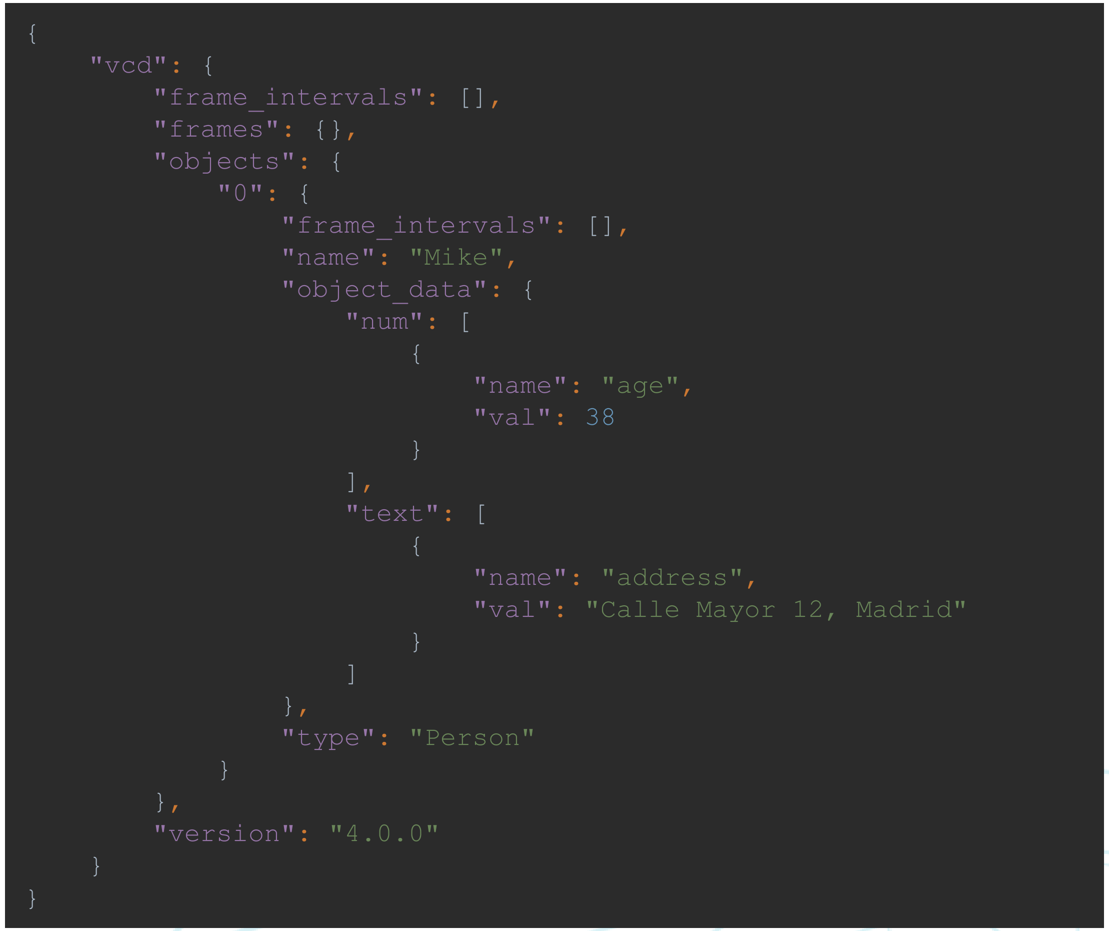
Python VCD library



```
{"vcd": {"frames": {}, "version": "4.0.0", "frame_intervals":
[], "objects": {"0": {"name": "Mike", "type": "Person",
"frame_intervals": [], "object_data": {"num": [{"name": "age",
"val": 38}], "text": [{"name": "address", "val": "Calle Mayor
12, Madrid"}]}}}}}
```

263 bytes

Pretty

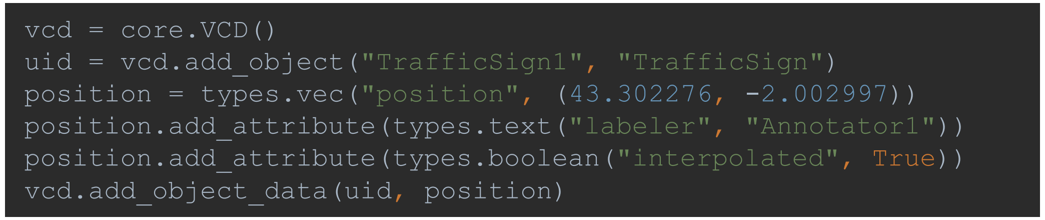


```
{
    "vcd": {
        "frame_intervals": [],
        "frames": {},
        "objects": {
            "0": {
                "frame_intervals": [],
                "name": "Mike",
                "object_data": {
                    "num": [
                        {
                            "name": "age",
                            "val": 38
                        }
                    ],
                    "text": [
                        {
                            "name": "address",
                            "val": "Calle Mayor 12, Madrid"
                        }
                    ]
                },
                "type": "Person"
            }
        },
        "version": "4.0.0"
    }
}
```

VCD Video Content Description

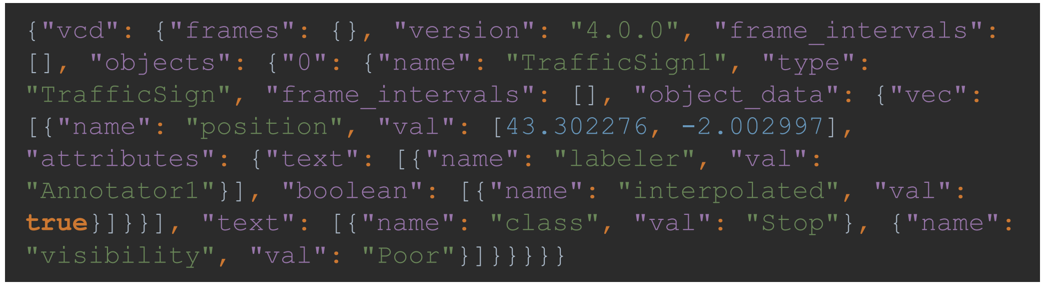
Serializable JSON/Proto

Object “TrafficSign1” is a “TrafficSign”
TrafficSign1’s position is (43.302276, -2.002997)
TrafficSign1’s class is “Stop”
TrafficSign1’s visibility is “Poor”
TrafficSign1’s position’s labeled by “Annotator1”
TrafficSign1’s position’s interpolated is “True”



```python
vcd = core.VCD()
uid = vcd.add_object("TrafficSign1", "TrafficSign")
position = types.vec("position", (43.302276, -2.002997))
position.add_attribute(types.text("labeler", "Annotator1"))
position.add_attribute(types.boolean("interpolated", True))
vcd.add_object_data(uid, position)
```

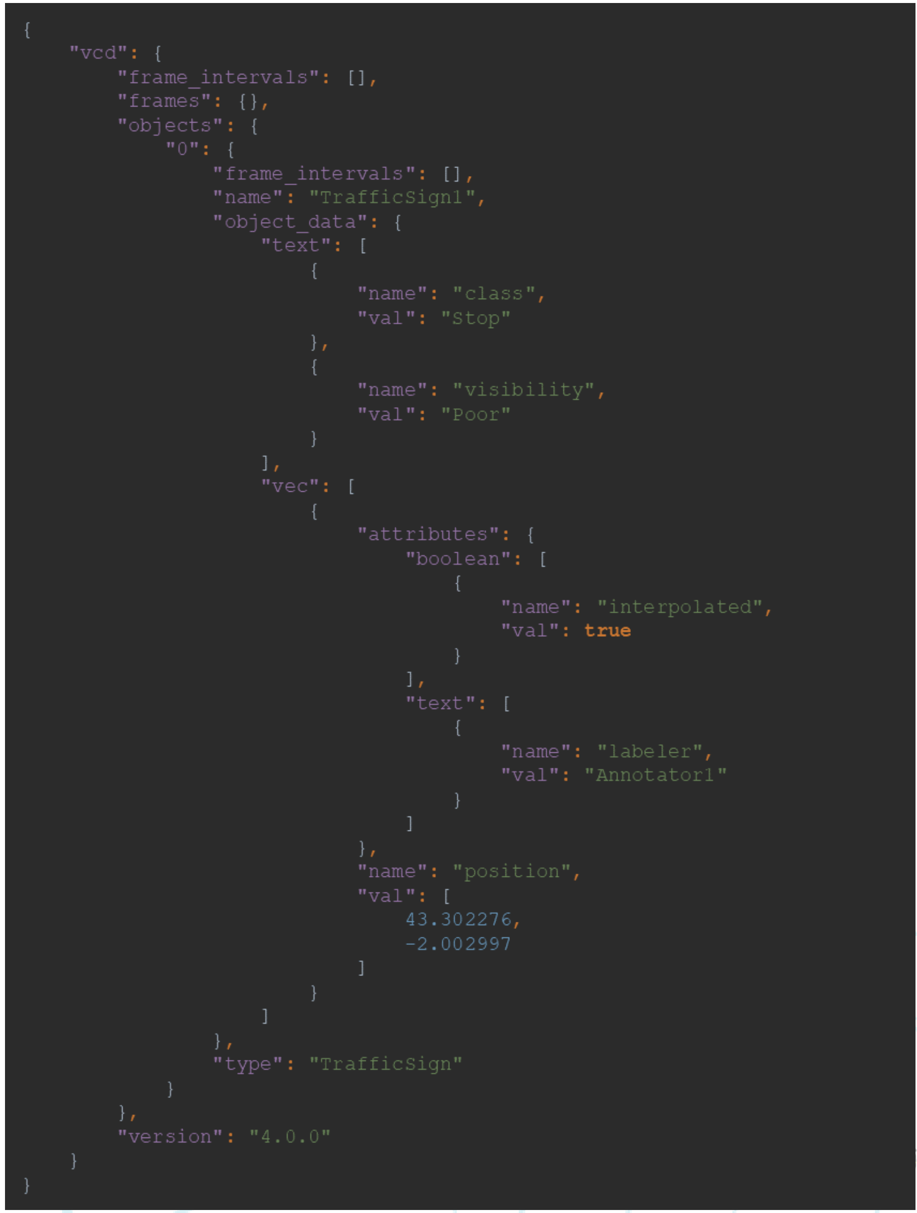
Python VCD library



```json
{"vcd": {"frames": {}, "version": "4.0.0", "frame_intervals":
[], "objects": {"0": {"name": "TrafficSign1", "type":
"TrafficSign", "frame_intervals": [], "object_data": {"vec":
[{"name": "position", "val": [43.302276, -2.002997],
"attributes": {"text": [{"name": "labeler", "val":
"Annotator1"}], "boolean": [{"name": "interpolated", "val":
true}]}}], "text": [{"name": "class", "val": "Stop"}, {"name":
"visibility", "val": "Poor"}]}}}}}
```

438 bytes

Pretty

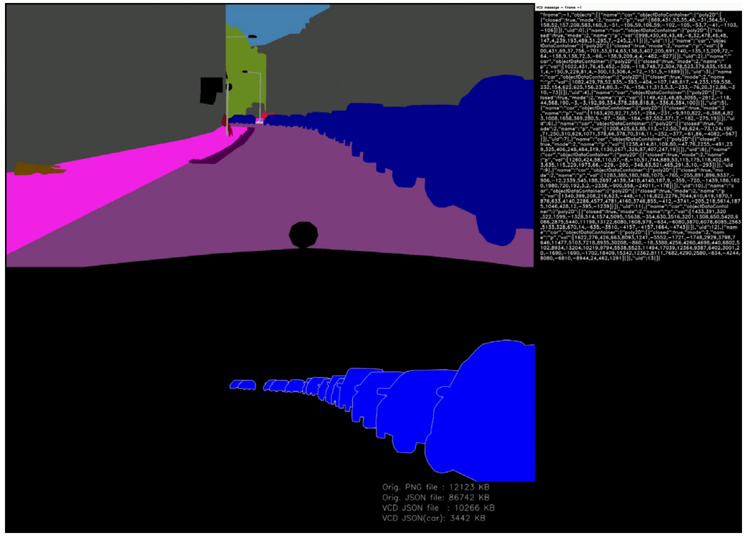


```json
{
    "vcd": {
        "frame_intervals": [],
        "frames": {},
        "objects": {
            "0": {
                "frame_intervals": [],
                "name": "TrafficSign1",
                "object_data": {
                    "text": [
                        {
                            "name": "class",
                            "val": "Stop"
                        },
                        {
                            "name": "visibility",
                            "val": "Poor"
                        }
                    ],
                    "vec": [
                        {
                            "attributes": {
                                "boolean": [
                                    {
                                        "name": "interpolated",
                                        "val": true
                                    }
                                ],
                                "text": [
                                    {
                                        "name": "labeler",
                                        "val": "Annotator1"
                                    }
                                ]
                            },
                            "name": "position",
                            "val": [
                                43.302276,
                                -2.002997
                            ]
                        }
                    ]
                },
                "type": "TrafficSign"
            }
        },
        "version": "4.0.0"
    }
}
```

VCD Video Content Description

Pixel-wise / Semantic segmentation

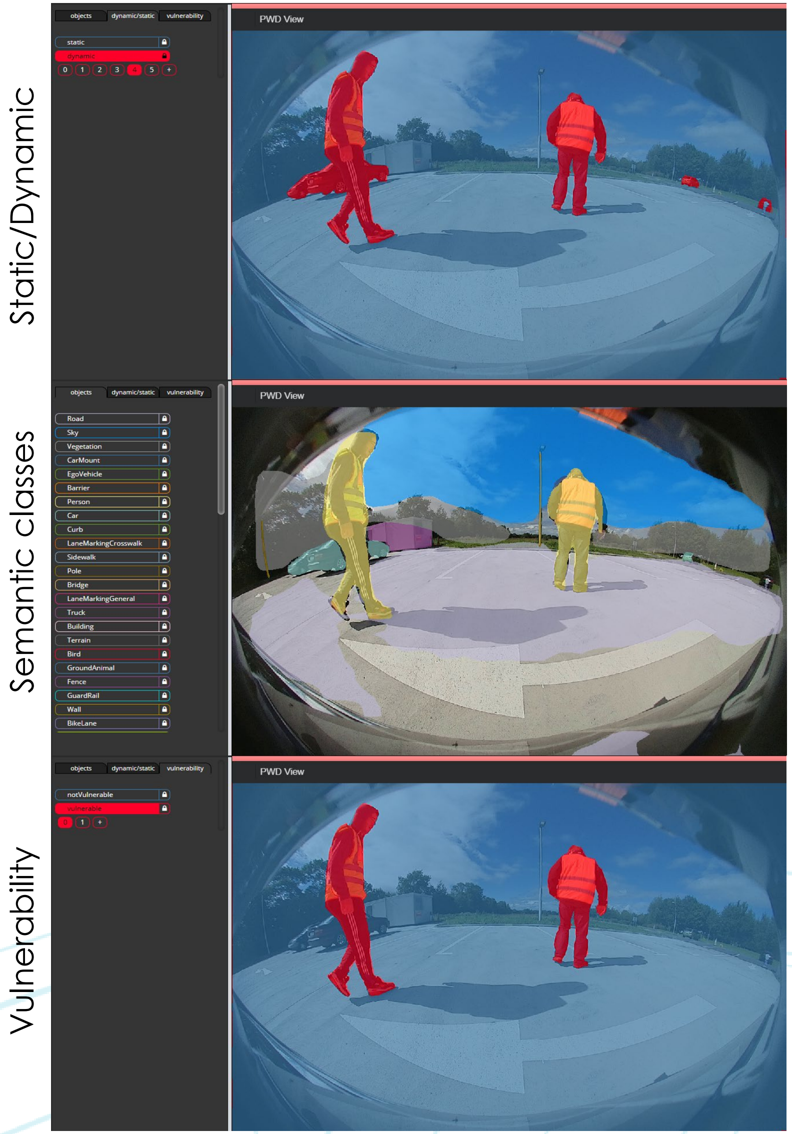
- **Huge datasets** (Mapillary, KITTI) for machine learning training
  - Data: Original HR images
  - Labels: PNG images, each pixel with a code color representing a class, and an instance
- VCD allows representing **these PNG labels** as polygons or
  - Accessing specific classes, no need to have image readers each time
  - Compression: polygonization techniques can be applied
- VCD can describe **class**, **instances**, and **holes**
- **VCD Python library has lossless polygon compression**



VCD message - Frame -1

"frame":-1,"objects":[{"name":"car","objectDataContainer":{"poly2D":[
{"closed":true,"mode":2,"name":"p","val":[869,431,53,35,48,-31,364,51,
158,52,157,208,583,160,3,-51,-106,59,106,59,-102,-105,-53,7,-41,-1103,
-106]}]},"uid":0},{"name":"car","objectDataContainer":{"poly2D":[{"clo
sed":true,"mode":2,"name":"p","val":[998,430,49,43,48,-8,32,478,45,48,
147,4,239,193,489,51,295,7,-245,2,11]}]},"uid":1},{"name":"car","objec
tDataContainer":{"poly2D":[{"closed":true,"mode":2,"name":"p","val":[9
00,431,69,37,756,-701,33,614,63,138,3,407,205,691,140,-135,13,209,72,-
64,-138,9,138,72,3,-66,-138,9,209,4,4,-482,-827]}]},"uid":2},{"name":"
car","objectDataContainer":{"poly2D":[{"closed":true,"mode":2,"name":"
p","val":[1022,431,76,45,452,-309,-118,748,72,304,78,523,379,835,153,8
1,4,-150,9,229,81,4,-300,13,306,4,-72,-151,5,-1889]}]},"uid":3},{"name
":"car","objectDataContainer":{"poly2D":[{"closed":true,"mode":2,"name
":"p","val":[1082,429,78,52,935,-393,-404,-107,148,617,-4,233,159,538,
232,154,622,625,156,234,80,3,-76,-156,11,313,5,3,-233,-76,20,312,86,-3
10,-73]}]},"uid":4},{"name":"car","objectDataContainer":{"poly2D":[{"c
losed":true,"mode":2,"name":"p","val":[1149,423,48,65,3055,-2912,-118,
44,568,190,-3,-3,192,99,334,378,288,818,8,-336,6,384,100]}]},"uid":5},
{"name":"car","objectDataContainer":{"poly2D":[{"closed":true,"mode":2
,"name":"p","val":[1163,420,92,71,551,-284,-231,-9,910,822,-6,368,4,82
3,1008,1658,369,280,5,-87,-368,-184,-87,552,371,7,-182,-275,19]}]},"ui
d":6},{"name":"car","objectDataContainer":{"poly2D":[{"closed":true,"m
ode":2,"name":"p","val":[1208,425,63,85,113,-12,50,749,624,-73,124,190
,71,250,310,629,1071,378,66,378,70,316,11,-252,-377,-61,86,-4082,-567]
}]},"uid":7},{"name":"car","objectDataContainer":{"poly2D":[{"closed":
true,"mode":2,"name":"p","val":[1238,414,81,109,80,-47,76,2255,-491,23
9,325,406,248,484,319,1130,2671,326,87,407,247,19]}]},"uid":8},{"name"
:"car","objectDataContainer":{"poly2D":[{"closed":true,"mode":2,"name"
:"p","val":[1260,424,58,110,57,-8,-10,51,744,689,53,115,175,118,402,46
3,635,115,229,1973,66,-229,-290,-348,63,521,465,291,5,10,-293]}]},"uid
":9},{"name":"car","objectDataContainer":{"poly2D":[{"closed":true,"mo
de":2,"name":"p","val":[1283,385,180,168,1075,-765,-255,891,896,9337,-
906,-12,2339,545,188,2697,4139,3418,4140,187,9,-359,-720,-1439,186,162
0,1980,720,192,3,2,-2338,-900,558,-24011,-178]}]},"uid":10},{"name":"c
ar","objectDataContainer":{"poly2D":[{"closed":true,"mode":2,"name":"p
","val":[1340,399,208,219,623,-448,-1,116,822,2276,7044,610,619,1870,1
876,633,4140,2286,4577,4781,4160,3746,855,-412,-3741,-205,218,5614,187
5,1046,428,12,-395,-1239]}]},"uid":11},{"name":"car","objectDataContai
ner":{"poly2D":[{"closed":true,"mode":2,"name":"p","val":[1433,391,320
,322,1599,-1328,514,1574,5095,15638,-354,630,3516,3201,1308,650,5420,6
066,2875,5440,11196,13122,6080,1608,979,-634,-6080,3870,6078,6085,2563
,5133,328,670,14,-635,-3510,-4157,-4157,1664,-4743]}]},"uid":12},{"nam
e":"car","objectDataContainer":{"poly2D":[{"closed":true,"mode":2,"nam
e":"p","val":[1622,276,426,663,8093,1241,-5552,-1721,-1748,2929,3798,7
646,11477,5103,7218,8935,30208,-860,-18,3388,4256,4260,4698,440,6802,5
102,8934,13204,10219,9794,5538,5523,11494,17039,12364,9387,6402,3001,2
0,-1690,-1690,-1702,18409,15342,12362,8111,7682,4290,2580,-834,-4244,-
8080,-6810,-8944,24,462,1291]}]},"uid":13}]}

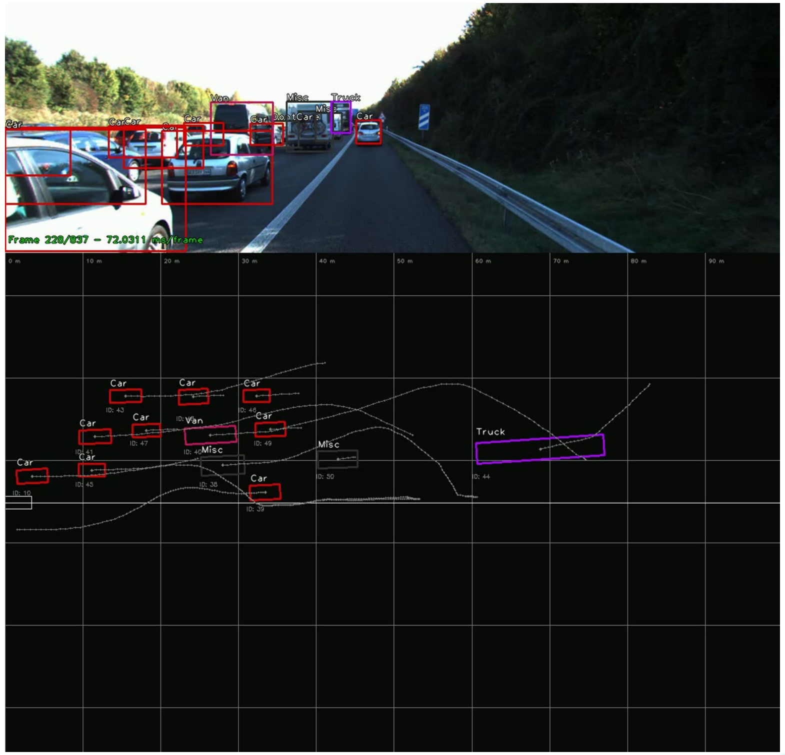
```
Orig. PNG file : 12123 KB
Orig. JSON file: 86742 KB
VCD JSON file  : 10266 KB
VCD JSON(car): 3442 KB
```

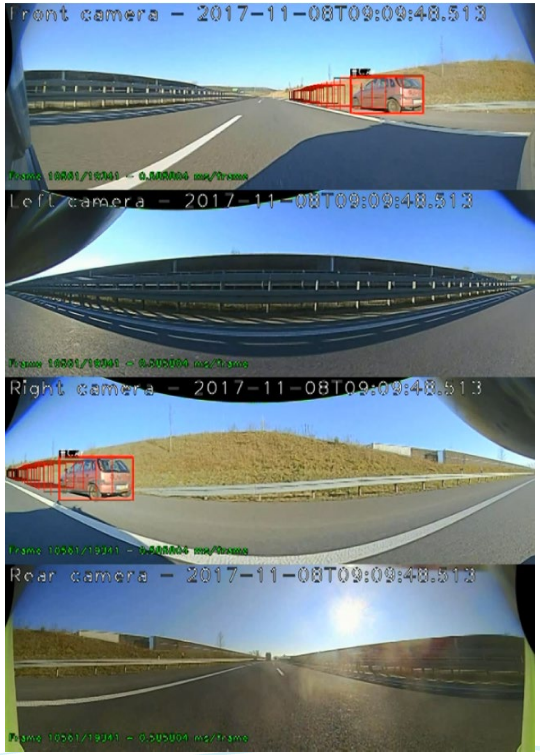


VCD Video Content Description

Multi-sensor annotation

- **Same real object**
- Described as a single
  - Inside, each representation of the Object for each view/stream
- Useful if data is annotated in several steps
  - Video first
  - Then LIDAR
  - Or different annotators/detectors
- **Annotations can be added** seamlessly to the VCD
- **Timestamping** and synchronization is added to guarantee alignment with data




Front camera – 2017-11-08T09:09:48.513
Left camera – 2017-11-08T09:09:48.513
Right camera – 2017-11-08T09:09:48.513
Rear camera – 2017-11-08T09:09:48.513

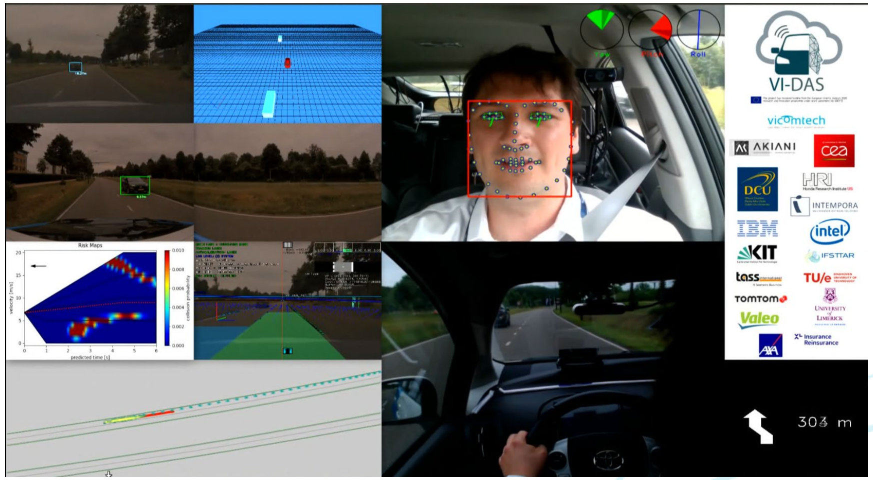
VCD Video Content Description

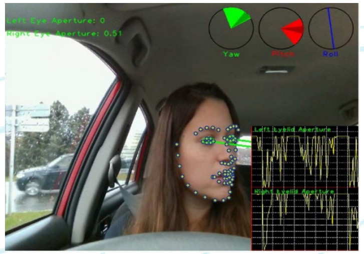
Useful Object primitives

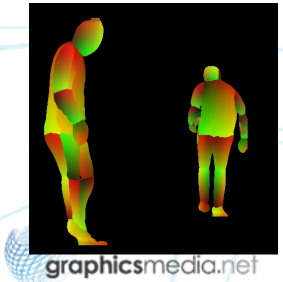
- **ObjectData**
  - bbox
  - poly2d
  - poly3d
  - image
  - binary
  - mat
  - num
  - vec
  - text
  - boolean
  - mesh
  - line_reference
  - area_reference
- Nested ObjectData (**attributes**) to describe any complex structure

Additional example use cases

- Lanes as 3D polylines with text attributes
- Parking slots as meshes of points, lines and areas, with attributes
- Lateral position within ego-lane as vec
- Eye blinks, head pose and gaze vectors of driver
- Skeleton and human-pose
- …





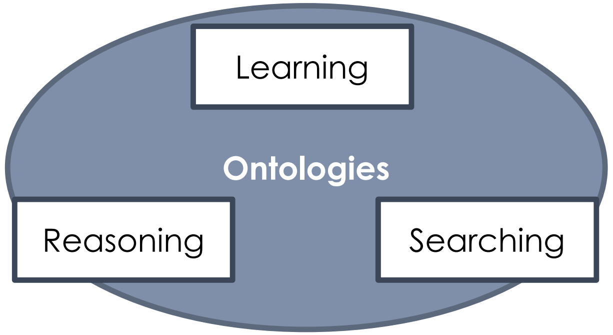


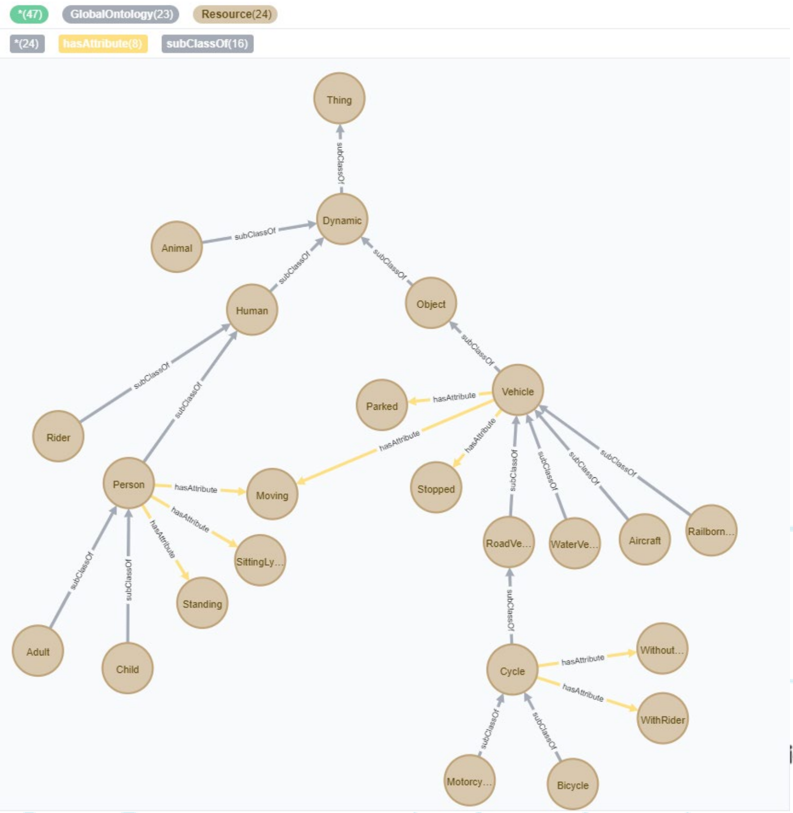
OSL Ontology-based Semantic Labelling

Semantic labelling - Ontologies

- Semantics provide meaning to data
  - Learning
  - Reasoning
  - Searching

- Ontologies
  - Classes
  - Properties
  - Relations


Learning
Ontologies
Reasoning
Searching



Ontology
.owl

- Ontologies to host knowledge, establish rules, enable translation, advanced querying

- Labelled data using classes, properties and relations from Ontologies
  - Enable dataset fusion, translation
  - Guarantees compatibility with future extensions, adding detail
  - Advanced querying

OSL Ontology-based Semantic Labelling

Semantic labelling - Labels

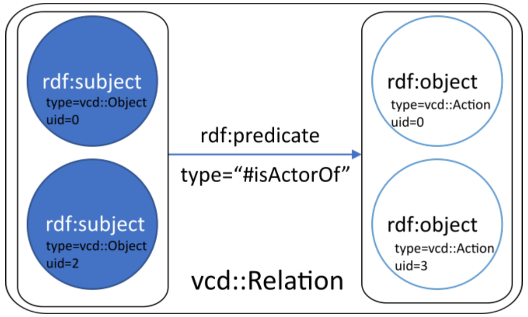
- Element’s types can refer to a Class in an ontology

  e.g. **Object** “Mike” is a “#Person” as defined in https://vcd.vicomtech.org/ontologies/automotive

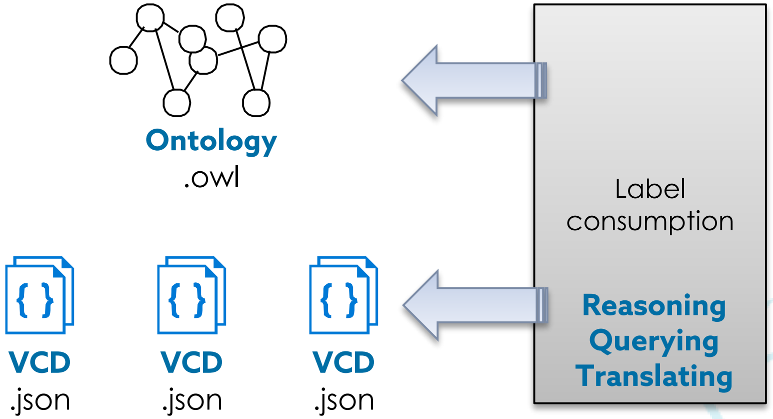
- **Relations** can define relationships between elements, defined as ontology classes

  Relations are defined as **RDF triplets**

  e.g. **Relation** “r1” means “Mike” “#isActorOf” “MikeCrossing”

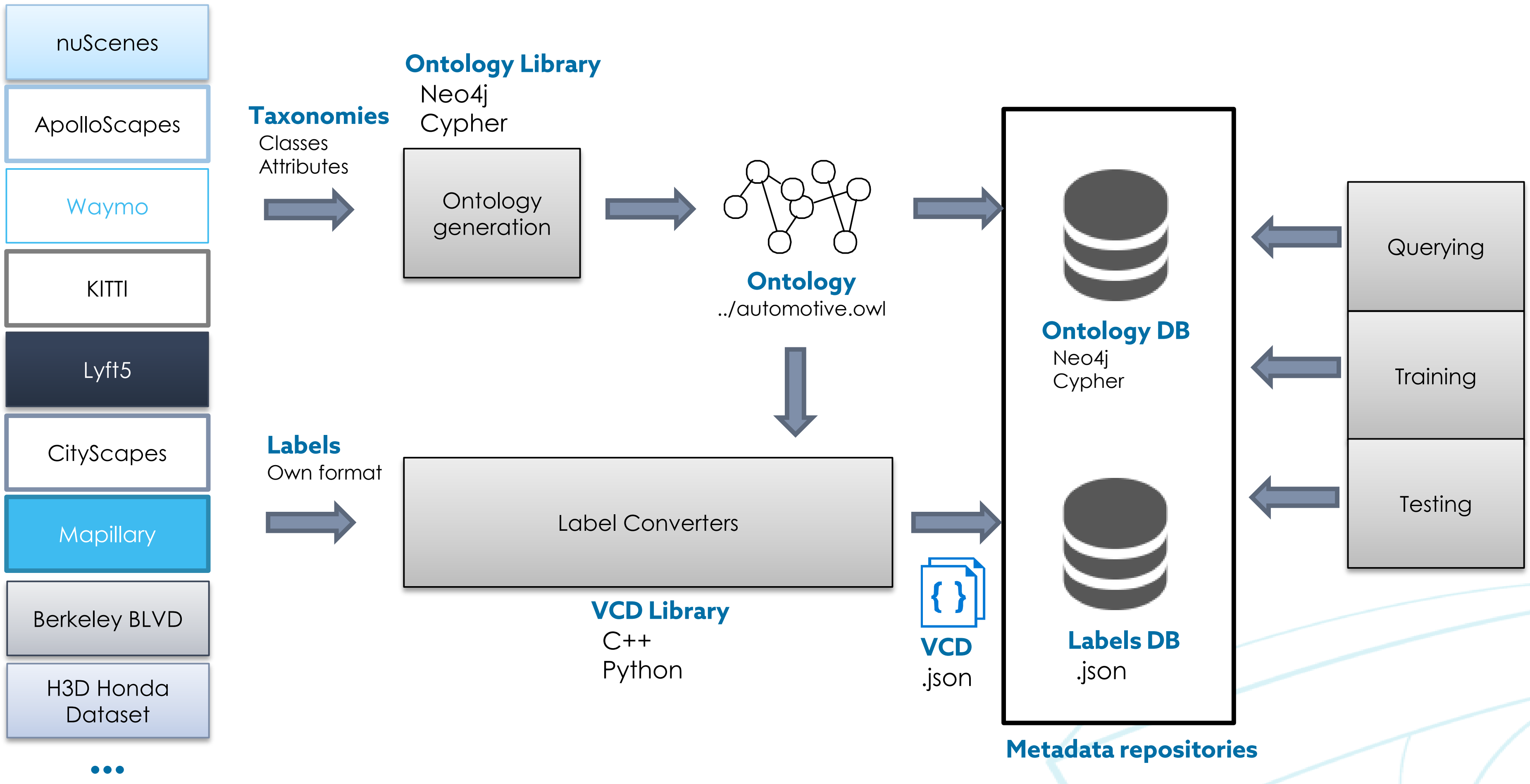


- Ontology-related keywords
  - **Class hierarchies (“isA”)**: Pedestrian “isA” Person, SUV “isA” Vehicle
  - **Similarity/Translation links (“sameAs”)**: Pedestrian “sameAs” Fußgänger
  - **Possible attributes (“hasAttribute”)**: Pedestrian “hasAttribute” moving
  - …



OSL Ontology-based Semantic Labelling

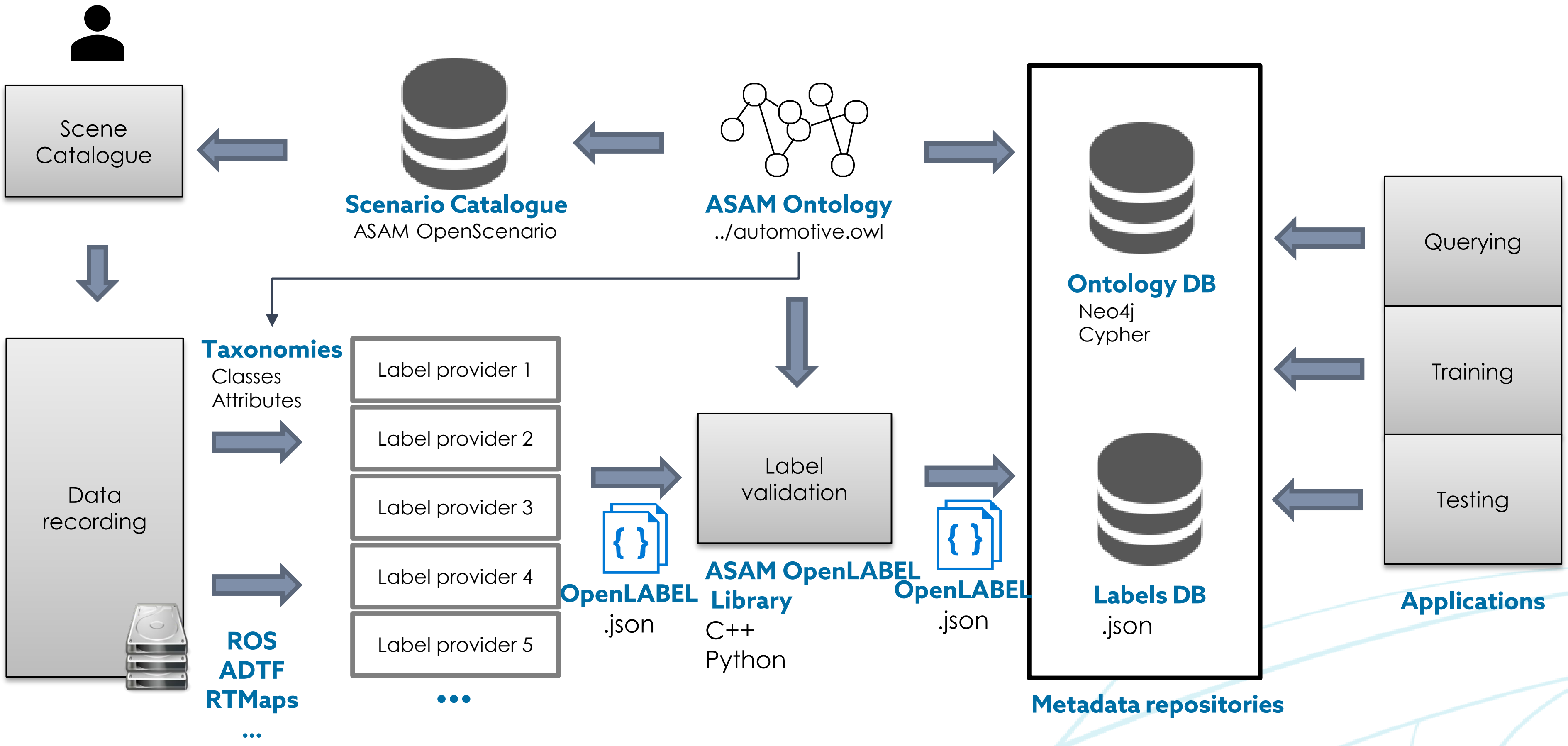
Semantic labelling – Creating the standard



OSL Ontology-based Semantic Labelling

vicomtech

Semantic labelling – Using the standard



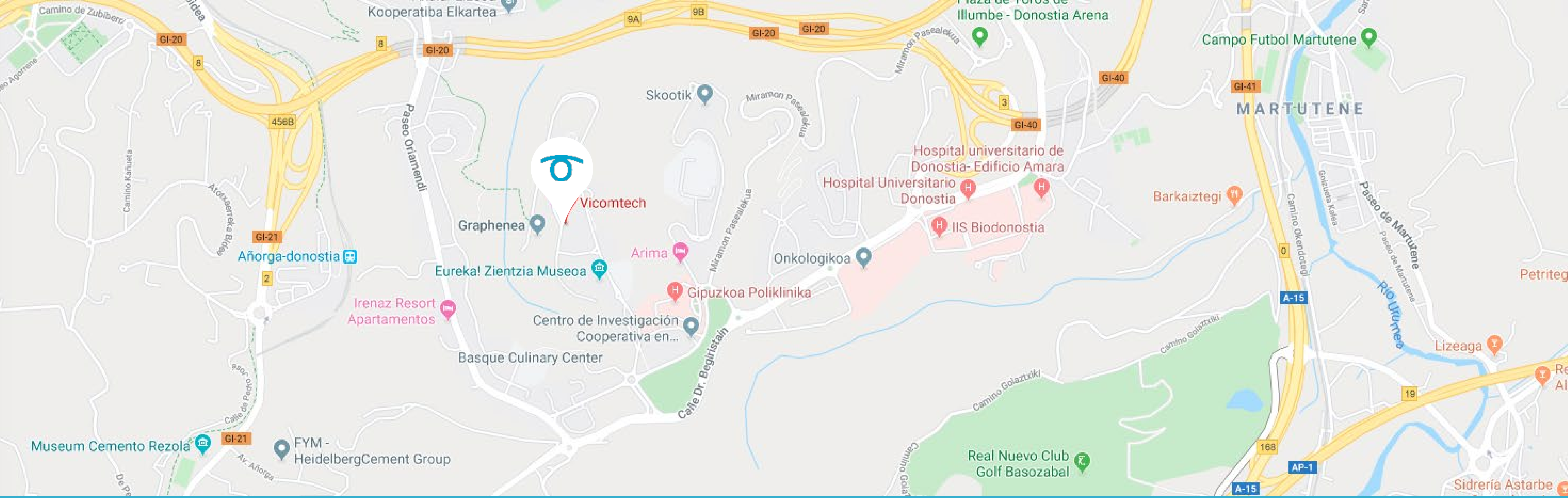
graphicsmedia.net

Next steps

- **Provide documentation about** VCD 4.0.0 schema and VCD 4.0.0 Python library
- **Consolidate OSL Ontology** from major datasets
- **Open github** repositories and publish current developments

- Study **OpenLABEL requirements** and participate in **definition of standard**
- Analyze co-existence of other related languages and standard initiatives (**OSI, OpenScenario, OpenDrive**)

- **Create tools and applications**
- **Open to develop reference implementation of standards**



Eskerrik asko
Gracias
Thank You

Address:
Paseo Mikeletegui 57
San Sebastián, Spain

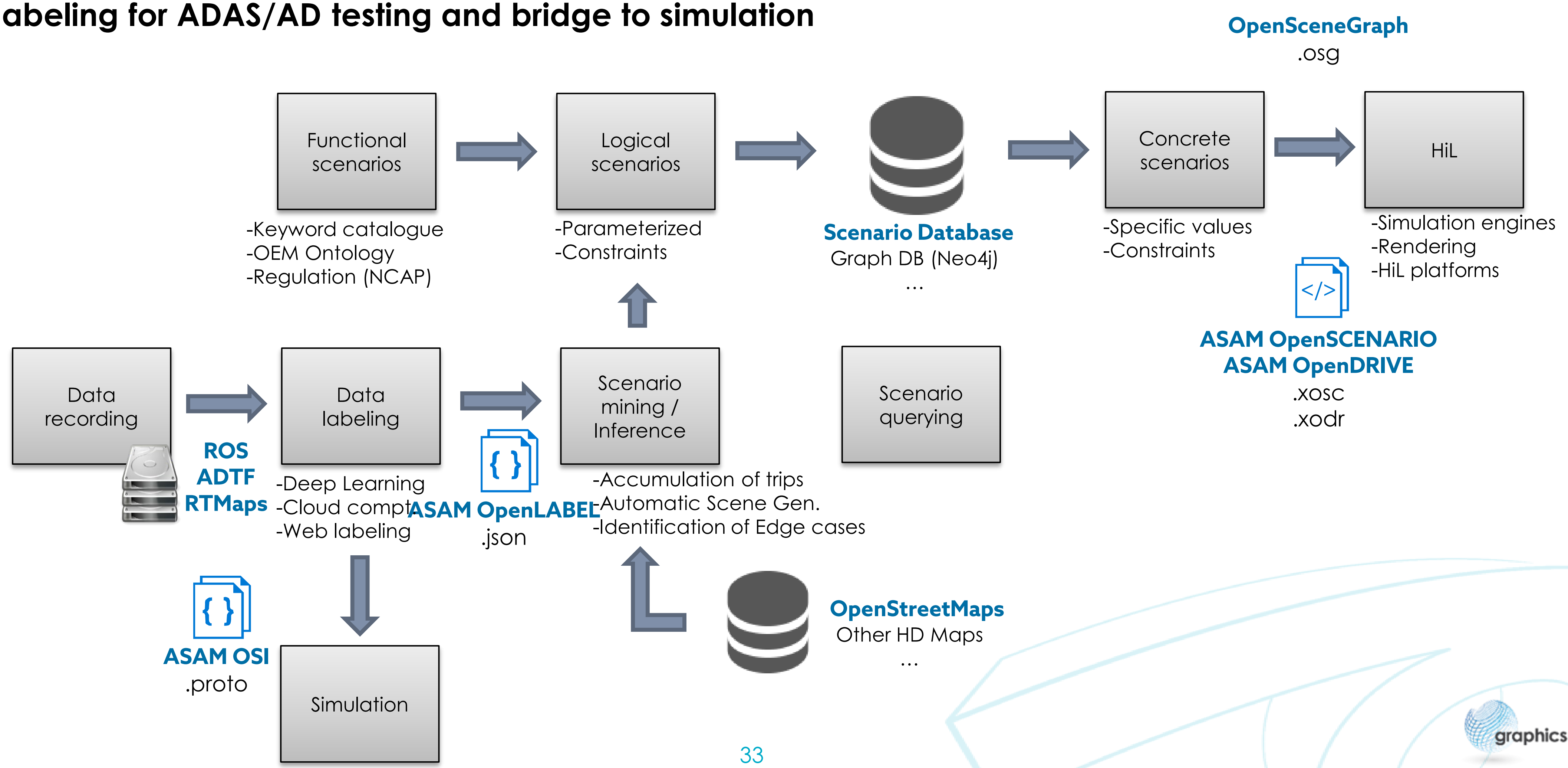
+34 943 30 92 30

ootaegui@vicomtech.org

OSL Ontology-based Semantic Labelling

vicomtech

Data labeling for ADAS/AD testing and bridge to simulation



OpenSceneGraph
.osg

Functional scenarios
-Keyword catalogue
-OEM Ontology
-Regulation (NCAP)

Logical scenarios
-Parameterized
-Constraints

Scenario Database
Graph DB (Neo4j)
…

Concrete scenarios
-Specific values
-Constraints

HiL
-Simulation engines
-Rendering
-HiL platforms

ASAM OpenSCENARIO
ASAM OpenDRIVE
.xosc
.xodr

Data recording

ROS
ADTF
RTMaps

Data labeling
-Deep Learning
-Cloud compt
-Web labeling

ASAM OpenLABEL
.json

Scenario mining / Inference
-Accumulation of trips
-Automatic Scene Gen.
-Identification of Edge cases

Scenario querying

ASAM OSI
.proto

Simulation

OpenStreetMaps
Other HD Maps
…

graphicsmedia.net