Developement of an autonomous driving environment model visualization based on object list level

Tobias Wagner Christoph Zach Max Haindl Philipp Korn Stehpan ...? Dennis Roessler Max Pfaller Domi ...?

***Abstract*— This electronic document is a live template. The various components of your paper [title, text, heads, etc.] are already defined on the style sheet, as illustrated by the portions given in this document.**

I. INTRODUCTION

II. RELATED WORKS

IEEE Fabio Reway Test Method for Measuring the Simulation-to-Reality Gap of Camera-based Object Detection Algorithms for Autonomous Driving

III. MATERIALS AND METHODS

Kleines Intro was jetzt kommt

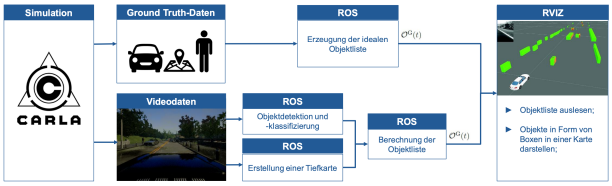

Fig. 1. Ueberblick

A. Creating simulation scenario

- Welcher Simulator wurde verwendet
- Welches Szenario (NCAP)
- Szenario beschreiben

B. Creating objects list of ground-truth data (TP1

- Erstellung Objektliste
- Objektliste anhand Attribut-Vektor beschreiben
- Ros-System beschreiben
- Feature Vektor Ermittlung

C. Evaluation of video data (TP2)

- Detektion Objekte (Yolo)
- Tracking Objekte (Tracker)
- Gleichung zur Berechnung von zB Geschwindigkeit, Beschleunigung
- Ermittlung Classification / prop mov / prop exis

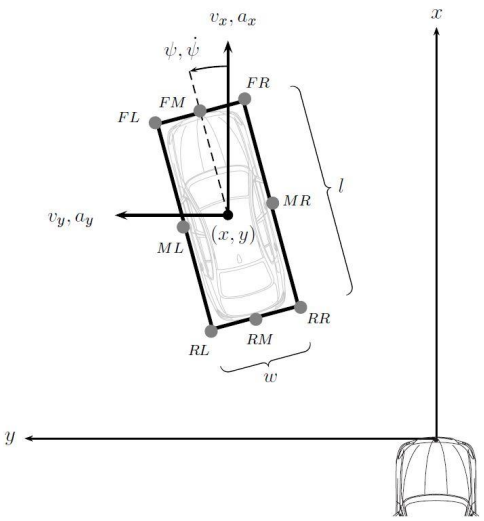

Fig. 2. Fahrzeugkoordinatensystem

*This work was not supported by any organization

D. Visualization of object lists

The published topics of Ground-Truth data and Camera-Calculation data will be subscribed. Each topic contains the ego vehicle data and the specific generated object list. In order to evaluate the object lists, the objects will be analyzed per frame. In RVIZ, the objects are represented by primitive figures with the help of marker messages. Figure 3 shows the used topics and nodes. Rectangles represent topics and ellipses the called nodes. Moreover, tf is a package and controls the coordinate relationship of the ego vehicle.

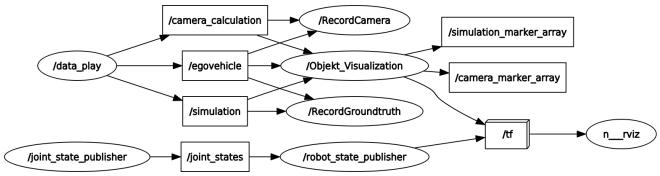

Fig. 3. Nodes / Topics in ROS

Marker messages are described with specific properties such as position, scale, type, color, orientation. Each object class will assigned selected shapes and colors so that they can be differentiate in RVIZ. The display variants for the possible object classes are shown in table I.

TABLE I
CLASSIFICATION ASSIGNMENT

Classification	Shape	Color[RGB]
car	cube	[1, 0, 0]
truck	cube	[0, 1, 0]
pedestrian	cylinder	[0, 0, 1]
motorcyle	cube	[1, 0, 1]
car	cube	[1, 0, 0]
bicycle	cylinder	[1, 1, 0]
stacionary	sphere	[0, 1, 1]
other	sphere	[1, 1, 1]

In addition, the yaw angle of the objects has to be transformed into a quaternion for the visualization in RVIZ. The markers for the calculated camera data are assigned an RGB alpha value of 0.5, so that the difference between the camera data and the GT data is visually recognizable.

The highest detection probability of an object indicates the classification, so that the properties value of each shape can assigned to the marker message. Furthermore, each detection position must be mirrored on the Y-axis, because the vehicle coordinate system does not match to the RVIZ coordinate system. Finally, the generated markers are combined into a marker array and will be published.

The ego vehicle is described as a robot model by a URDF model. Furthermore, the model can be moved and rotated in the RVIZ coordinate system by tf messages.

The pubished topics of Ground-Truth data, Camera-Calculation data and ego vehicle data are also saved in a Rosbag File. Each bagfile contains the published ego data and the corresponding object lists.In the following, these files are used for postprocessing.

1) Evaluation: To determine whether a given camera object is evaluated as True Positive (TP), False Positive (FP), False Negative (FN) or a mismatch (mm), the **Intersection over Union** (IoU) value is used, like shown in [**?**]. Like shown before, in general there is a list of m camera objects (B_{pr}) and a list of n ground truth objects (B_{gt}) for each frame. To evaluate a frame, for each combination of GT object and camera object the IoU value is calculated. All those values build a matrix like shown in Table II.

TABLE II
IOU-MATRIX FOR A SINGLE FRAME

$IoU(B_{gt,1},B_{pr,1})$	$IoU(B_{gt,2},B_{pr,1})$	...	$IoU(B_{gt,n},B_{pr,1})$
$IoU(B_{gt,1},B_{pr,2})$	$IoU(B_{gt,2},B_{pr,2})$	...	$IoU(B_{gt,n},B_{pr,2})$
...	...	...	...
$IoU(B_{gt,1},B_{pr,m})$	$IoU(B_{gt,2},B_{pr,m})$	...	$IoU(B_{gt,n},B_{pr,m})$

A given camera object $B_{pr,i}$ is ...

... **FP**, if there is no value

$$IoU(B_{gt,k},B_{pr,i}) > t \quad \text{with} \quad k \in \{1,...,n\} \tag{1}$$

in the according row of the matrix which is greater than the given threshold.

... **FP**, if there is one or more IoU values in the according row greater than the threshold, but for every $B_{gt,k}$, for which equation 1 is true, there is another $B_{pr,j}$ ($j \neq i$) which matches with $B_{gt,k}$ and $IoU(B_{gt,k},B_{pr,j}) > IoU(B_{gt,k},B_{pr,i})$

... a mismatch (**mm**) if it is no FP case, but none of the found possible matching $B_{gt,k}$ has the same class as $B_{pr,i}$.

... **TP**, also called a match, if none of the other mentioned cases are detected. That means, that there is at least one $B_{gt,k}$ which fulfills equation 1 and has the same classification as $B_{pr,i}$ and there is no other $B_{pr,j}$ which matches better with the found $B_{gt,k}$

Going through the rows of the matrix, for each $B_{pr,i}$ in the given frame it can be decided, whether the case is TP, FP or mm.

On the other way round, examining the Ground Truth objects $B_{gt,k}$, that means the columns of the calculated matrix, all FN cases can be detected. It is an **FN**, if there is no $B_{pr,i}$ for which

$$IoU(B_{gt,k},B_{pr,i}) > t \quad \text{with} \quad i \in \{1,...,m\} \tag{2}$$

Going through the columns of the matrix, this decision can be made for every $B_{gt,k}$.
With this steps, a given frame with m camera objects and n ground truth objects can be investigated.

In this project these functions, one for investigating the camera objects and one for detecting all FN cases, were realized in Python. The calculation of an IoU value is processed with functions of the package *shapely* [**?**]. First, the given objects, which are defined through their properties *x*, *y*, *length*, *height*, *yaw* and *classification* like presented in [**?**] are transformed into bounding boxes with a shapely function. With two of these bounding boxes shapely can calculate the intersection area and the union area, and so the IoU value can be processed.

2) Graphical User Interface: To investigate the quality of the processed camera object data, a graphical user interface (GUI) was created. It was designed with Pythons binding package for Qt (PyQt5) [**?**] and defined as a plugin for *rqt*, a *ROS* framework for GUI development [**?**]. With this plugin, the user can import two bag files, one ground truth data bag file and one camera data bag file. By using the functions mentioned before, the GUI can show several data graphs to the user, like raw data plots, comparing plots with object data of both files or evaluation data. Along with the data the mean value and the standard deviation for each data set is portrayed in the GUI.
Apart from data plots the interface can also show quality parameters for the whole camera data bag file in an extra widget. For each operation, where IoU calculation is needed, the user can set the threshold value for the evaluation.

3) Results: The performance of processed camera data can be evaluated with metrics presented in [**?**], which are realized like introduced in part III-D.1. For the given scenario the reached performance is shown in Table III.

TABLE III
PERFORMANCE RESULTS

threshold	$t = 0.5$	$t = 0.6$	$t = 0.7$
Precision	...	...	...
Recall	...	...	...
FPPI	...	...	...
MOTA	...	...	...
MOTP	...	...	...

4) Conclusions:

IV. RESULTS

Ergebnisse des Projekts:

- Funktioniert die Auswertung
- Wie gut sind die Kamerawerte im Vergleich zu Groundtruth Werte
- sind die Werte represenativ

V. CONCLUSIONS

APPENDIX

ACKNOWLEDGMENT