Software Tool for Evaluation of Multi-Sensor Object Tracking in ADAS Systems

A.Medaglini, S. Bartolini

Department of Information Engineering and Mathematics, University of Siena, Via Roma 56, Siena, Italy; email: {alessio.medaglini, sandro.bartolini}@unisi.it

V. Di Massa

Thales Italy, Via Lucchese 33, Sesto Fiorentino, Italy; email: vincenzo.dimassa@thalesgroup.com

F. Dini

Magenta srl, Via B. Pasquini 6, Florence, Italy; email: fabrizio.dini@magentalab.it

Abstract

Nowadays, the innovations of AI and other automated decision-making software are spreading to many different areas. The automotive field in particular is rapidly shifting towards the concepts of Advanced Driver Assistance Systems (ADAS) and Obstacle Detection and Avoidance Systems (ODAS), which could bring huge benefits in the future. However, before being able to use these tools, many assurances are required regarding their functioning and safety. To this end, several control techniques exist to evaluate the performance of these software, but a reliable and repeatable method for evaluating complex scenarios and corner cases is still lacking. In this paper we propose a suite of tools for the generation and analysis of synthetic tests, aimed at evaluating and analyzing the functioning of autonomous driving systems in order to measure their effectiveness and to drive their development.

Keywords: synthetic test, autonomous driving, software tool.

1 Introduction

In the past ten years there was an exponential growth in the use of electronic components and software in automotive systems. Driven by the revolutions in AI and machine vision, the automotive field has been profoundly renewed by inserting an ever-increasing number of driving aid tools inside cars, from lane keeping to complete autonomous driving systems. In particular, thanks to the huge amount of data that can be collected, taking advantage of inertial platforms, sensors, and so on, automated decision-making systems are becoming increasingly popular. This huge amount of innovations is shifting the market towards Advanced Driver Assistance Systems (ADAS), with the aim of developing fully autonomous vehicles in the future. In such scenario, to verify the operation and evaluate the performance of this new kind of vehicles, a very broad and thorough analysis is required. In fact, as stated in [1], "Autonomous vehicles would have to be driven hundreds of millions of miles and sometimes hundreds of billions of miles to demonstrate their reliability in terms of fatalities and injuries". As one can imagine, it is not possible to verify such a requirement with tests done with real vehicles only, but it must also be accomplished by exploiting simulation tools to fully evaluate the reliability of these kind of vehicles. With regard to tests with real vehicles, there are also some drawbacks that make their use not recommended. The main of them are shown below:

- there is a too wide variety of scenarios to be explored and it would require an unimaginable amount of time, and resource.
- it could be difficult to exactly replicate a specific test scenario to verify if an improvement can help in facing it correctly.
- it is difficult to obtain quantitative measurement of performance using real collected data since as it does not gives us any reference about the desired behavior.
- danger for the personnel involved during the testing procedure is too high to take a chance, especially in edge cases [2].

Nowadays, for all these reasons, simulated tests seem to be an essential way for automotive industry to provide the safety requirements of autonomous vehicles, reducing the mentioned costs and giving a speed up to the testing procedure. Nevertheless, performing tests that produce quantitative, repeatable and comparable results remains challenging for autonomous vehicles, since the reliability of the results is strongly dependent on the accuracy of the simulated information used as input for the software. In particular, one of the crucial parts of the systems for autonomous driving is managing the problem of Object Tracking and Obstacle Detection (OTOD) [3], which is the focus of this paper. The situation awareness is indeed a crucial aspect to be able to develop properly working decision-making systems that are fully reliable in urban traffic scenarios. For this reason, we focus on the study of generating scenarios for such critical activities. In fact, our work aims at evaluating and measuring OTOD features and performance through a modular and efficient simulated approach.

In this area, there are some proposals that have been made to manage such issue, which can be split into two categories. The first is more oriented to the generation of test scenarios starting from real data, while the other is based on the use of mathematical models. The approaches in the first category gather data from real-world runs, to obtain a database of scenarios that can be used for building the simulated scenarios. On the other hand, approaches in the second category rely on abstract scenario generation, defined by mathematical or semantic languages which are then translated into test scenarios. Unfortunately, these methods can be expensive in some respects or requires a huge base of data to combine. Furthermore, the problem of evaluating the final results of the elaboration of these product scenarios is not always completely addressed, and many evaluation criteria are often based on different performance indicators whose interpretation and composition is not uniquely determined [4].

We propose a methodology, within the second category, that models the salient aspects of tracking and sensing objects, to effectively abstract the necessary facets to test OTOD behavior in ADAS systems. Our tool generates synthetic scenarios that replicate the sensors' perception of the objects around a vehicle, so that they are representative and effective without burdening the model with unnecessary information, promoting high modularity and flexibility. The main contribution of this paper can be summarized as follows:

- a method for the simulated evaluation of object detection and tracking submodules of an ADAS system.
- a synthetic scenario generation tool configurable through a simple scripting language that allows to describe scenarios quickly and concisely.
- a performance evaluation of the results obtained, through automatic calculation of aggregated Key Performance Indexes (KPIs), to produce comparable and easy-to-understand reports from test execution.

2 Related works

As already said in the previous section, it is possible to categorize the systems that deal with the evaluation of OTOD systems substantially in two macro categories, one more oriented to the generation of test scenarios starting from real data and another that instead bases the generation on mathematical models. An example of work related to the first category is proposed in [5]. In this paper they proposed an approach for scenario generation based on a scenario database. In particular, their schema builds upon the generated scenario database structure that clearly identifies the key components of a given autonomous paradigm. This abstraction enables the creation of parameterized test cases to test the autonomous functions under various adaptive conditions. Their scenario database consists of data collected from multiple sources, and store information about real world sensor data, parameters required to create the setup for testing and scenario definitions related to the functions supported. Similarly, de Gelder and Paardekooper [6] propose a method for evaluating the performance of the functions in an ADAS system based on real-life scenarios, taken from the Streetwise database, combined with Monte-Carlo simulations. Another possible approach, proposed in [7], is based on the use of advanced perception systems for obtaining reference data used for the automated generation of simulated driving scenarios. In this case the data provided by their referenced sensor system can be transferred into a simulation tool, to obtain virtual scenarios from real world scenarios. The second category instead relies on abstract scenario generation. For instance, in [8] the scenes and the related assertions are defined by a matrix-based semantic language and translated into test scenarios in simulation. They developed a semantic language for breaking down the factors that define a scenario, taking the input from the command line and parsing the formal grammar to generate tokens. Then, using a matrix based system they generalize the scenario characteristics. The numerical matrix is read as input where each row is a different assertion describing a single road piece or actor that can then be parsed to generate the scenario. In [9] they generate both static, i.e. scenarios where objects just follow a predefined trajectory, and hybrid scenarios where the vehicles need to to deviate due to the influence of other vehicles in order to avoid a crash. In order to do that they use a combinatorial interaction-testing algorithm together with a backtracking algorithm and a motion planner.

Our approach is mainly related to the category based on mathematical models, but we differ from the reported works for some main reasons:

- our scenario definition language is simple and easily human readable, it does not involve matrix definition or other complex mathematical formalization of the items present on the scene.
- in our case complex algorithms are not required to create scenario data starting from its representation, requiring a great computational power, but the output is automatically generated starting from the objects involved, and based on the sensors used, lightly and quickly.
- our tool also model how different sensors perceive the environment, replicating their transducer characteristics and specific format of data and meta-data, along with the different kind of noises that can affect them.

3 The ADAS system: a real-world use-case

Thales Italy (TH-ITA) industry has developed a multi-sensor ADAS system for city trams for the obstacle detection and collisions avoidance. The system assists and supports the driver in avoiding collisions by detecting and tracking obstacles in real-time, thereby compensating for driver errors. The aim of TH-ITA is to enhance the safety for trams and light rail vehicles, to the benefit of passengers, service operators and other traffic participants. Although technology cannot replace human drivers, it can complement human perception and decision making – often deciding between life and death. Indeed, the system will be able to significantly reduce the number of rear-end collisions involving tram vehicles and, as a result, help to avoid high follow-up costs.

The complexity of city traffic requires cognitive capabilities to improve vehicle reactivity and perception of near- and

long-range obstacles. The development of this technology and its impact in light rail transit will result in improved safety of daily operations. In fact, tramways are a challenging application for autonomous driving systems for many reasons. First of all, compared to mainline railways, trams rails are not always segregated from road traffic and pedestrians. For this reason, while in mainlines any detected obstacle on the track has to be considered a threat to safety [10], for a tram driving system it is not so straightforward to discriminate whether an object on the track rails can constitute a safety threat or not, depending on the specific situation. For example a car could drive on the rail, in front of the tram, preceding it while driving the same direction. In the same way it is common to find people crossing the rails or in close proximity of them, for instance in all those cases where the tram is approaching to a platform with people standing and walking in the surroundings. These tramway scenarios are normal and should not cause an alert. In contrast, when the ADAS system detects a car or pedestrian whose future trajectory can be predicted to intersect the tram's one, it shall generate an alert so that the driver can stop or slow down the tram to avoid collision. In fact, in a typical use case, there are many vehicles that can move around the tram, which is also moving. This situation generates a variety of possible scenarios, according to the different obstacles, weather, lighting and other spatial contour conditions around the tram. Moreover, along its path the tram meets some dangerous conditions like traffic lights, crosswalks, crossroads and cars parked on the street sides with people walking around them.

In addition, the same driving situation is certainly much more dangerous for a tram vehicle than an automobile. This is mainly caused by the high braking distance required by a tram, which is very different with respect to the one of a car, due to the great difference in weight and coefficient of friction with the respective transit surfaces between the two types of vehicles. For instance, as highlighted in Figure 1, to stop a tram traveling at $35km/h$ are needed about $20m$, which is the same distance required to stop a car running at more than $60km/h$. This produce a large increase in risk in the management of a tram with respect to a car, due to the absolutely relevant momentum of the former even at relatively low speeds, which can create safety hazards for other road users during its operating.

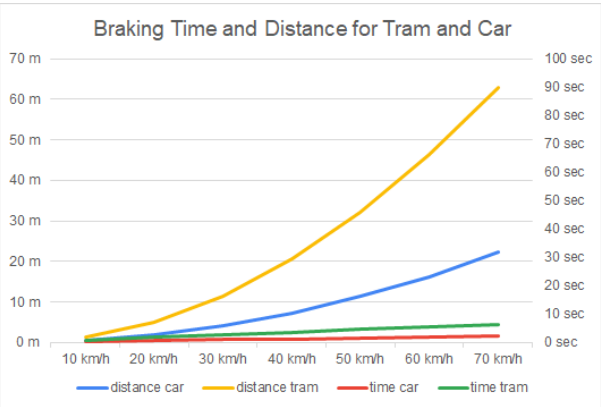


Figure 1: Braking time and distance for tram and car.

In Figure 2 some possible everyday scenarios are reported, with different objects moving around the tram. Each image within the figure shows different types of objects and road topologies that the tram encounters along its path. In fact, there are crossroads, crosswalks, platforms with people, cars moving parallel to the rail tracks or intersecting them. Depending on the evolution of their behavior all these objects can become obstacles for the tram. We consider an obstacle any possible object (including cars, bicycles, animals, human beings, things) which can collide with the tram because it stands between the rails or because it stands nearby and its shape and trajectory is suitable with a collision. The fact that both the tram and the surrounding objects can move poses critical issues from the point of view of the correct identification of the objects and the nature of their movement. In particular, there are situations in which the tram is stationary and has objects moving around it and other where the tram is moving and this affects the relative speed of the other objects (both moving and stationary). This latter case can produce critical effects for the object tracking algorithm. For instance when the tram curves, especially if the turn has a narrow radius, all the objects rapidly shift on the scene, and their speed and position change anomalously.

(a) Vehicles around the tram **(b) Motorbike crossing tracks**

(c) People at the tram stop **(d) Crossroads**

Figure 2: Typical urban scenarios taken from real tramway

The images in Figure 2 are taken from the camera mounted on the real tram. In fact, to be able to perform object detection and tracking algorithms, the vehicle must be equipped with a heterogeneous set of sensors. In particular, in the TH-ITA case, each tram has been equipped with two cameras, a radar sensor and a lidar one. The way these sensors perceive reality is critical to obtain good performance from ADAS systems [11].

The overall architecture of the TH-ITA ODAS system can be represented as depicted in Figure 3, where the entire data pipeline is reported. The ODAS system is composed of three main subsystems: a set of sensors that are installed on the tram vehicle, a data association and tracking (DAT) module, and a collision checker module (CCM). First of all, the sensors collect raw data from the real world, which are then analyzed by the respective pipelines to provide the system with bounding boxes representing detected objects. These data are then propagated to the DAT module, which is in charge of elaborating the raw data collected by the sensors

and associating them with the traces tracked by the system. This submodule traces the targets' evolution and produces estimates on the future positions of the objects, which are sent to the CCM of the system. This last module deals with the driving logic of the tram, deciding whether the future position of an object will be critical to the system, causing a collision, and if so providing an alert to the tram driver. As can be easily understood, the DAT module is the core of the ODAS system and, within it, we mainly focus on the detection and tracking aspects. In fact, the reliability of the system is based on the correctness of the object detection and tracking phase.

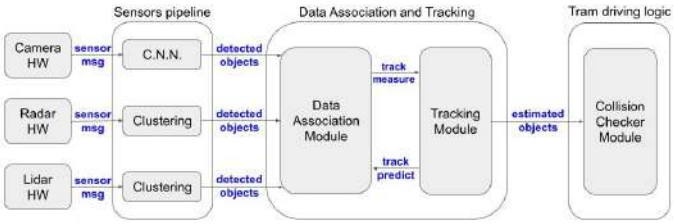


Figure 3: Architecture of an ADAS/ODAS system

4 Proposed approach

Developing a multi-sensor ADAS requires evaluating the behavior of association and tracking algorithms, which are fed with data from sensor fusion. To track real-world objects it is crucial to model how they are perceived by sensors. Therefore, we addressed both the simulation of reality, in a faithful but synthetic way, and also how this reality is perceived by the different sensors. Our proposal hence focuses on a method to generate scenarios that replicate as precisely as possible the trajectories of the objects and the sensor's perception of reality. In this way, it is possible to feed the data association and tracking algorithms with realistic and accurate data, for producing meaningful results that can then be analyzed. Indeed, the generation of test scenarios and the evaluation of the results obtained are two critical aspects that must be addressed jointly to be able of properly steering the development of an autonomous driving system. For this reason, we propose a software tool to guide the development of an autonomous driving system and to measure its effectiveness, starting from the description of a synthetic scenario to its implementation and evaluation. The first part of this proposal is dedicated to identifying a novel procedure to generate synthetic scenarios, while in the second part we deal with the evaluation of the obtained results using an automatic report generator based on reliable KPIs previously defined. To define the different scenarios and cases of study we follow an incremental approach. Initially, we classify all the possible behavior of a single object moving around the vehicle, distinguishing them according to the direction, trajectory, and position of the object. Then, we model the behavior of each possible sensor used for sensing the environment (camera, radar, and lidar in our case), specifying the typical characteristic of each one and modeling the noise that can affect them. In this way, by combining all the possible trajectories of objects and the way they are perceived by the different sensors, thousands of randomized variation of each specific scenario can be easily and effectively run. Lastly, the tool aggregates the results from multiple runs and can automatically produce reports summarizing the setup parameters, for experiment repeatability, and the achieved results through easily specifiable KPIs.

4.1 Objects movements

Regarding the first aspect, simulation of real objects, we identified some macro scenarios of typical configurations for the tramway system and the objects around, based on data collected by Thales Italy. A classification of these basic macro-scenarios is presented below:

- Static obstacle on the track rails: such obstacle could be a car blocked on the rails, a fallen bicycle or motorbike, a branch of a tree, or other unexpected things.
- Obstacle moving along the rail: a car or other vehicles moving along or beside the rail, from a side or the other one. This obstacle should have a size that is sufficient for impacting the tram or a trajectory too close and dangerous.
- Obstacle moving at a distance from the tram, but with trajectory and speed that are compatible with a future collision. This can happen when approaching a crossroad or a roundabout where vehicles cross the railway track.
- Obstacle moving in the nearby of the tram, but with trajectory and speed that are compatible with a future collision. This can happen for instance when the tram is stopped near a traffic light or a tramway station and other vehicles intersect the train tracks, passing close to the tram.

All these scenarios are very frequent during the travel of the tram since it is moving in a city area crowded with people and different typologies of vehicles in the surrounding of it. In particular, for the trajectories that move in front of the vehicle, it is important to make a distinction based on the distance from it, as it has an impact on the response time required to the vehicle. Furthermore, the direction in which the different trajectories are traveled is also relevant as it affects the vehicle's field of view. The above scenarios can be combined with each other to generate more complex situations, building new scenarios starting from multiple specific behaviors of different objects on the basis of the superposition principle or generating interference between them.

The taxonomy of basic behaviors we found for the objects, as reported above, is summarized in Fig. 4. This figure depicts the trajectories as plan views, with the trajectories as they would appear when viewed from above the tram. At the bottom of the figure the reference system is reported, which is centered in the front part of the tram with the x-axis directed along the direction of travel of the tram, the y-axis directed on the left, and the z-axis upwards. In particular, each of the lines represents one of the main categories of trajectories that we have identified as basic trajectories that an object can follow in the urban environment around a tram. In fact, from the analysis we carried out, each object around the tram can

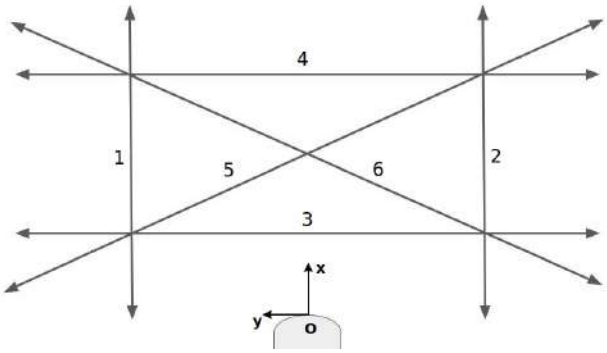


Figure 4: Basic trajectories of objects around the tram.

have different behaviors and therefore follow trajectories that can make it an obstacle for the train. These trajectories cover lateral movements with respect to the vehicle (num. 1,2), which does not intersect its trajectory, and movements that instead intersect its trajectory, both perpendicularly (num. 3,4) or with different angles (num. 5,6). Some examples of these trajectories can be seen in Fig. 2: the cars in Fig. 2a are moving following the trajectories of type 1 and 2; type 3 and 4 trajectories can happen in scenarios like the one reported in Fig. 2d; and in Fig. 2b an instance of the trajectory 5 is reported, but you can easily imagine similar scenarios for the type 6 trajectory. All the trajectories in Fig. 4 can be traveled in both directions, as underlined by the arrows drawn on both ends.

4.2 How to model objects

To model the behavior of objects on the scene, as explained above, it is necessary to estimate the temporal progression of their kinematic characteristics, just as sensors would capture them in the real case. Each sensor collects different information from the environment, and we identified that the movement of an object can effectively be described by a limited set of basic motion patterns like uniformly accelerated rectilinear motion or circular motion. In fact, each sensor allows to collect different information, but it is always possible to describe an object through its position, speed, and acceleration at different instants of time. We, therefore, decided to model the behavior of objects based on the description of these three kinematic characteristics, evolving them over time according to appropriate physical laws. In our model, each object is defined by specifying its initial position (assumed in the origin of the Cartesian system if not specified) and an initial speed and/or acceleration (assumed null if not specified). From that moment on the motion of the object is modeled using uniform or uniformly accelerated rectilinear motion laws using the set parameters. Providing different inputs for each of them at different instants of time it is therefore possible to accurately describe arbitrary motions of the different objects. Motion parameters (e.g. direction, acceleration) can also be changed, even randomly, at each time instant within the object's lifetime, so that even more complex motion laws can be easily modeled. Furthermore, such scenarios can be combined with each other to generate more articulated situations encompassing multiple objects and evaluating the consequent interaction effects in the object detection and tracking algorithm.

The object description is based on a formal language used to define each case. It is implemented in a human-readable form so that it is easy to write and immediately understood at first glance, as can be seen in the following snippet:

```
GLOBAL_END_TIME,40000
1,0,POS,2,−20,0
1,0,VEL,1,1,0
2,0,POS,2,20,0
2,0,VEL,1,−1,0
1,300,ACC,1,0.5,0
2,450,VEL,1,−2,0
```

The language exploits a comma-separated value format, where the first element in each line is the object id, the second one is the time instant (in millisecond) at which the following property is applied, the third is a kinematic keyword (*POS* for position, *VEL* for velocity and *ACC* for acceleration) representing the property we want to specify and, finally, there are the values of that property for each Cartesian coordinate *(x,y,z)* of the previously specified property. This triplet is expressed as meter (m) for position property, meter per second (m/s) for velocity, and meter per square second (m/s^2) where the property is equal to acceleration. We also create two other reserved words which are *GLOBAL_END_TIME* to define the last time instant in our scenario (on a reserved line) and *END* which is used instead of the kinematic property of an object to remove such object from the scenario at the specified time instant. Through this language, it is possible to define the position, speed, and acceleration along every Cartesian direction *(x, y, z)* for each involved object. These properties can be specified at each time instant within the lifetime of the object so that we can create even complex varying motion laws in a few steps. A specific part of our software is then in charge to verify that the described scenario is consistent, in order to avoid unexpected or not feasible behaviors. Using this procedure it is possible to describe every feasible scenario, and then obtain automatically an output file for each simulated sensor. The output file is created again in a CSV format, easily readable by both humans and machines, containing the kinematic properties of every object at each time instant, with a frequency corresponding to the specific simulated sensor's characteristic period. The output file can then be used to test each scenario within the specific application under development.

4.3 How to model sensors

During the scenario generation procedure we allow modeling of each sensor, accounting for its output data format, transducer characteristics, as well as the noise that can affect it. In particular, the output data format comprises the type and variety of positional data and meta-data (radar cross-section, probability of existence, object class, and so on) provided by the sensor. In fact, our system takes care of modeling each different kind of sensor in use, because sensors are one of the most relevant parts of ADAS systems. They are in charge of capturing information from the

environment, and therefore the final results strictly depend on their functioning. Moreover, since sensors also capture noise in their measurements, coming from poor external conditions or electromagnetic interference depending on the kind of sensors, it must be considered in the model. Consequently, our tool allows easy modeling of different possible measurement noises like additive zero-mean Gaussian noise, the type of noise that most commonly affects sensors, with a range of variances estimated from real data, and also some random zero/saturation/missing values can be simulated according to arbitrary statistical distributions. The features described above allow you to have high repeatability and controllability of the test scenarios, to make a huge number of experiments in many different conditions with the opportunity of easily steering the software development. Through this procedure, however, some approximations are introduced. These approximations mainly concern the dynamics of objects and the model used for the sensors, which however accurate cannot be the same as the real one. A fundamental aspect is in fact the trade-off between accuracy fo the scenario and computational power required to realize it, which must be optimized in order to make the designed scenario computable, as well as reliable, and significant.

4.4 Report generation

Another aspect covered by our tool is system evaluation. In fact, to evaluate the performance of the system it is of paramount importance to be able to compare the obtained results with a ground truth measurement of the specific scenario used. If the scenario comes from real-world data, a ground truth reference can be obtained only if we perform manual data inspection and labeling. This manual inspection and labeling procedure is prohibitively expensive to be performed extensively and accurately. Instead, using synthetic data, we describe the scenario we want to test and then simulate equivalent sensors to produce the information used to feed the data association and tracking components. Finally, we can compare the results with the known ground truth. Moreover, when we produce simulated sensor data we are able to inject known noise into the simulated data to model real-world noise. This way we can not only analyze and measure the performance of the implemented algorithms, but we can also infer properties about sensors' noise and test the resiliency of the system against them.

Once the scenarios have been tested, the final stage is to collect the produced results and aggregate them into meaningful KPIs. This phase is extremely important and comparative reports play an important role in each industry, for evaluating the performance level of a software application and for driving its evolution. Typically, those kinds of reports are manually composed from large amounts of information provided by various heterogeneous sources. Processing this information is tedious, time-consuming, and error-prone. For all these reasons, we have developed an automatic tool that aggregates the results from multiple runs of the tested software, executed on different scenarios or with different functioning parameters, and produce a comprehensive report which summarizes the environmental setup of the specific execution and the obtained results evaluated using some previously defined KPIs.

5 Performed experiments and evaluation

5.1 Methodology

Our synthetic model is based on the analysis of data collected on real trams. We carried out qualitative tests by analyzing videos of cars, pedestrians and other objects to empirically estimate their motions and the parameters to be used. Using our generation language it is possible to specify the initial conditions of each object and modify them during its movement. Object positions evolve according to discrete-time cinematic equations of uniform or constant accelerated rectilinear or circular motions, possibly with varying parameters over time. In time periods between two changes to the dynamic of a specific object its evolution is based on the physical laws of uniformly accelerated rectilinear motion. Obviously, if the acceleration is not present in the input parameters it is assumed equal to zero and the two equations reduce to a uniform rectilinear motion.

An important aspect to evaluate the behavior of the system is represented by the performance metrics to be used. In this work, we decide to focus on evaluating the performance of the multi-object tracker inside the ADAS system. In order to do so, we need to understand what qualities we expect from it. In an ideal world, such a tracker should, at all points in time, be able to identify the correct number of objects on the scene and estimate the position of each of them as accurately as possible. Additionally, we expect the tracker to be able to consistently track each object over time. This means that, if each object has been assigned to a unique trace ID, it remains constant throughout the entire sequence (even after a temporary occlusion). This leads to the following evaluation criteria for performance metrics:

1. They should allow to judge a tracker's precision in determining exact object locations.
2. They should reflect its ability to consistently track object configurations through time, that is, to correctly trace object trajectories, producing exactly one trajectory per object.
3. They should be clear, easily understandable, and behave according to human intuition.
4. They should be few in number and yet expressive, so that they can be used, for example, in evaluating complex systems.

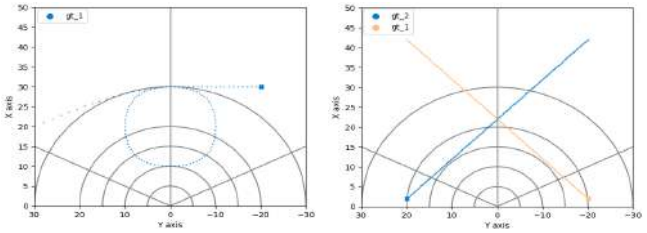
Based on the above criteria, we adopt the MOT Metrics [12] performance evaluation of the multi-object tracker inside the ADAS. The two most used metrics in this area are the Multiple Object Tracking Precision (MOTP) and the Multiple Object Tracking Accuracy (MOTA). In particular, we have chosen to calculate the MOTP also for each sensor used. This allows us to understand how each individual sensor contributes to the overall performance of the system, by varying the noise level modeled for each sensor. Then, we decided to add another metric to our KPIs: since we are mainly interested in evaluating the behavior of our tracker in the immediate surroundings of the tram, i.e. the areas

with the greatest risk of collision, we have decided to also use the Root Mean Square Error (RMSE) as a performance index. To this aim, we have divided the tram's field of view (FOV) into sectors, based on the distance and angle of objects relative to the tram, using a plan view of the scenario, and we compute the RMSE value for each of them. This is useful for understanding how the position of an object, its distance and angle with respect to the tram, affect the ability of the system to track it correctly. Indeed, performance evaluation has different levels of interest based on the location of sectors, with those close to the vehicle being the most critical for collision detection. For this reason, the FOV of the tram has been divided in such a way to focus the attention on the bands closest to it, with circular sections of increasing width as the distance increases. Furthermore, the FOV has been divided into four lateral slices according to the angle of the objects with respect to the tram, to separate the objects that are located in front of the tram from peripheral ones, which may be more difficult to follow. Lastly, it is worth noticing that thanks to the modularity and flexibility of our tool, it is easy to add and tune new KPIs to manage specific situations if needed.

5.2 Experiments and results discussion

The test framework we developed allows us, through scenario design and test's reports generation, to have deep insight into the system operations. There are a lot of parameters that an ADAS system depends on, and in this way we can evaluate the effects of each of them. Among the various analyzed trajectories and scenarios, we focus here on two of them, reported in Fig. 5:

1. In the first case an object is simulated moving in front of the tram with a curvilinear trajectory that goes around a roundabout (real case, for example in Batoni square, Florence).
2. The second case instead deals with a multi-object scenario where two objects move from each side of the tram with intersecting trajectories (any road intersection with vehicles coming from different directions).



(a) Roundabout motion **(b) Crossing trajectory**

Figure 5: Example of test scenarios

In the first test we performed, reported in Fig. 6, the sensors are modeled without noise, i.e. the ADAS system was fed with "exact" synthetic data, in order to evaluate their performance and behavior characteristics regardless of other influences. This is not realistic, but it is useful to tune the covariances of errors in the model we adopted so that it can cope with the dynamics of the objects. In fact, different sensors sense the environment differently, and this brings some difference in the measures they provide, even if they are referred to the same object. These differences in the measured position of the same object become noise for the system, and this needs to be accounted for in the measurement models. Thanks to our scenario generator we can simulate the behavior of the real sensors, and this allows us to visualize the differences between, for example, the lidar and the radar view of the same object. In Fig. 6, the image above shows the trajectory estimated with lidar measures only, instead in the image below the trajectory is estimated with radar measurements only. As the figure shows, the two trajectories are slightly different, because measurements from the two sensors are different in many ways: different timing, different estimated positions and different measurement models.

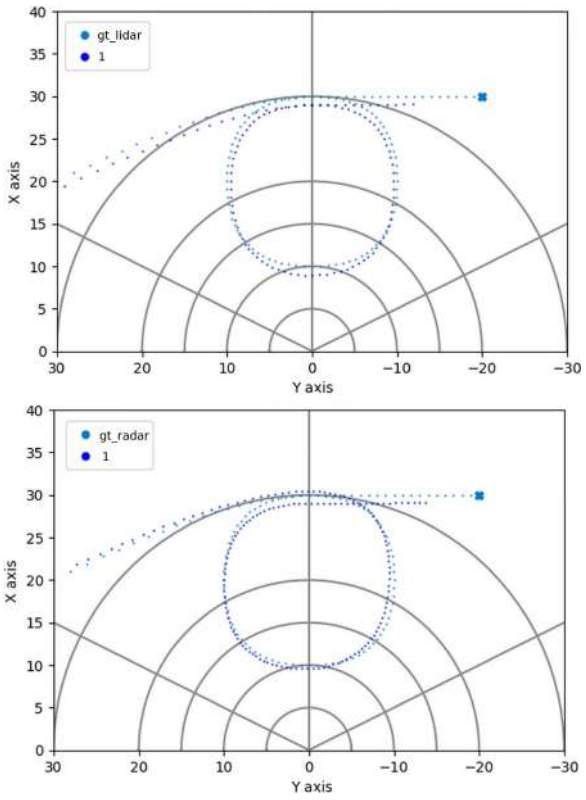


Figure 6: Single object track against GT, without noise. Above: lidar only. Below: radar only.

In this case, the track based on radar measures is more precise, as confirmed by the MOTP metrics value, which in the radar and lidar cases is equal to 84,3% and 90,8% respectively. This is due to the fact that the higher radar sensing ratio helps the system to correctly track the object. Therefore the different behavior in different conditions of the various sensors gives us the possibility of fusing their output, in order to enhance the overall performance of the system.

A second important feature we tested through our scenario generator regards the noise in the data, which we can control by varying the covariance of an additive zero-mean Gaussian noise that we inject into the input data. For different levels of noise estimated position of the targets appears farther from the ground truth and few measurements are left unassociated.

This can cause the tracker to lose its target and, if the unassociated measurements are enough, they could be used to instantiate a new tracker following the same target. By carrying out various tests we are able to investigate the sensitivity of MOTP values to noise variation. In Fig. 7 are reported the plots from two different simulations, with increasing noise covariance in sensor positions going from 0.5 to 0.9. A high level of noise in the measures degrades tracking performance, increasing the MOTP metric value from 53,7% to 69,3%. It is worth noting that these injected noise levels are larger than those usually recorded on sensor data, and we decided to apply them as a stress test for the system.

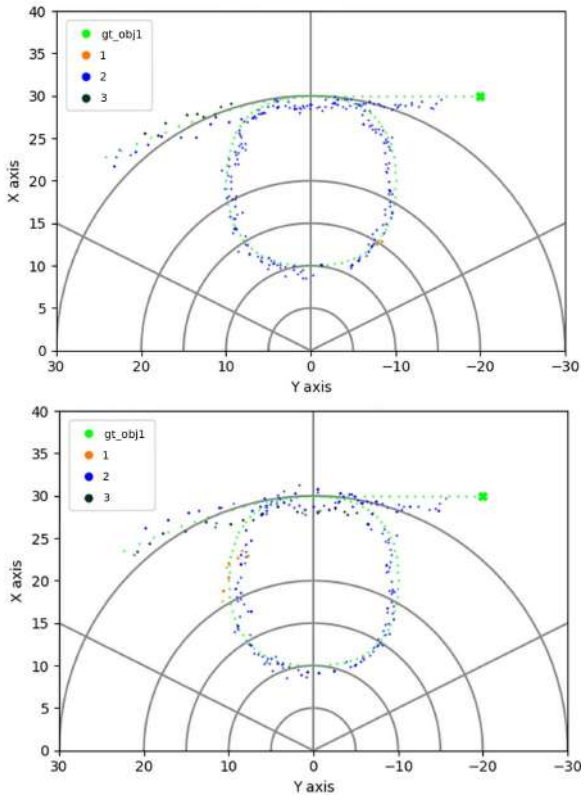


Figure 7: Effects of increasing noise in measurement data. Above: 0.5 covariance noise. Below: 0.9 covariance noise.

As a result, not only estimated position of the targets appears more noisy and far from the GT, but also a few measurements are left unassociated to the tracked object. This may have two effects: it may cause a tracker to lose its target and, if the unassociated measurements are enough, they could be used to instantiate new trackers which follow the same target. We have examples of this in both the plots, where orange and dark green dots are used for tracks different from the first one (blue). By carrying out this kind of experiments, we can investigate the sensitivity of the system to sensors' noise variation, measuring how much the accuracy of a single sensor affects the overall tracking results of the ADAS system.

Another key aspect addressed by our test framework is the sensitivity of the system to misdetections, i.e. cases when an object is present in the scene but sensors do not detect it. In fact, we define "misdetection" the event in which an object present in the scene does not cause a corresponding signal in a sensor's scan. When this happens the object is non-existent for that sensor. Actually, there are quite a few reasons why an object can be misdetected: the sensor may suffer from a temporary or local failure that prevent it from functioning correctly, there may be interference from external sources or, more likely, the object may be occluded by other objects or by the environment. Moreover, an object can also be mistakenly categorized as clutter and removed by decluttering filters. To test the system's sensitivity to objects' misdetections, our tool can simulate the random loss of measurements by defining the desired rate of misdetections for each sensor and evaluating the tracking robustness of the system. In this way, each time a measurement is supposed to be given as input by the system, there is a certain probability that the measurement will be simply dropped and not used by the system. In the plots in Fig. 8 we simulated an increasing misdetection probability of 20% and 40% on all sensors.

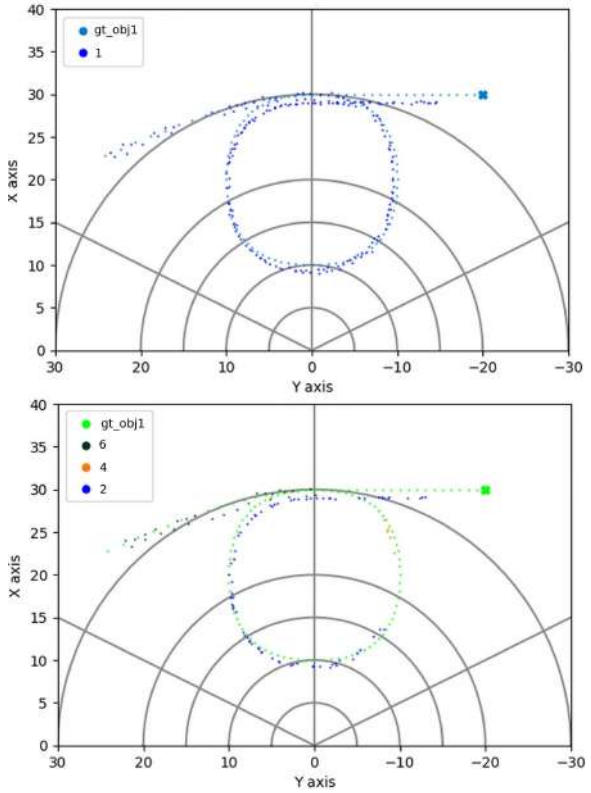


Figure 8: Effects of increasing probability of misdetection. Above: 20% misdetection probability. Below: 40% misdetection probability.

As shown from the figures, the system tolerates a high rate of misdetections, up to 20%, and this is probably due to the sequential approach we use to process the sensors' scans. In fact, although there is not a single sensor with a very high sampling rate, processing them when they arrive, one after the other, allows us to perform as if the system was fed by a single sensor having a sampling rate equal to the sum of each sensor's sampling rate. In the end, when the percentage of misdetections increases up to 40% the system cannot track the objects correctly anymore.

Single-target scenarios are useful to isolate specific issues

that could affect the core of the ADAS system, and can give important information about the performance of the system in terms of intra-track performance, such as those measured by the MOTP metric. They may also give some information about simple data association errors, as we have seen by simulating misdetection. However, they are not sufficient to investigate more complex scenarios where different association errors may occur, such as identity mismatch. In order to do this, we need to define multi-object scenarios where two or more objects may interfere with each other. A very simple example of this can be seen in the second scenario we want to discuss, reported in Fig. 5b. There, two objects start moving from each side of the tram, with intersecting trajectories. The objects' velocity is about 2.8 m/s, which is compatible for example with bikes or scooters.

In this multi-object scenario, repeating the experiments about misdetection probability, we can observe an interesting behavior. In Fig. 9 we can see that increasing the misdetection probability has no effect on the two predicted trajectories until 30% value is reached. Then, with this percentage of measurement loss, when the two objects intersect each other's trajectory, an identity mismatch occurs, as the plot clearly shows.

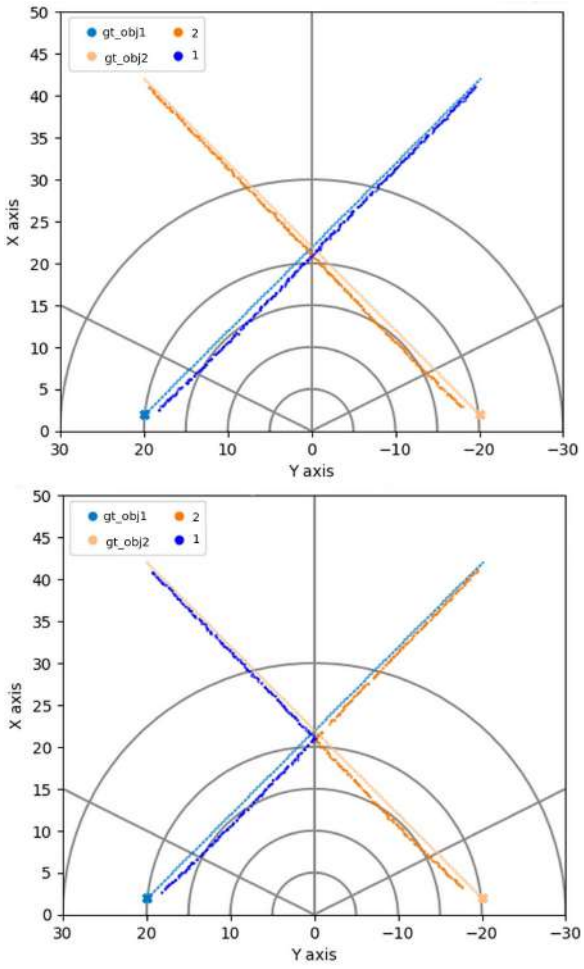


Figure 9: Effects of increasing probability of misdetection on multi-object scenario. Above: 20% misdetection probability. Below: 30% misdetection probability.

What happened here is that one of the two trackers suffered from a lack of measures close to the intersection, and at some point the association algorithm has associated it with measures originating from the other object. These measurements were stolen by their "legal owner", who was later forced to associate with the measurements originated from the other object. In the last part of the trajectories, although other misdetections surely occurred, the two objects are too far away to interfere with each other, and the tracking proceeds smoothly up to the end.

Lastly, when the misdetection probability reaches 40%, something different happens as reported in Fig. 10. Only the object moving from left to right suffered an association problem, while the other presented a continuous track. In this case, the association error is simply a misdetection error: one object suffered from a lack of measurements that elapsed long enough for the system to decide that the tracker had to be deleted, while the other continued to be fed with the correct measurements. This caused measurements from the first object to be left unassociated, and these were used to start another tracker.

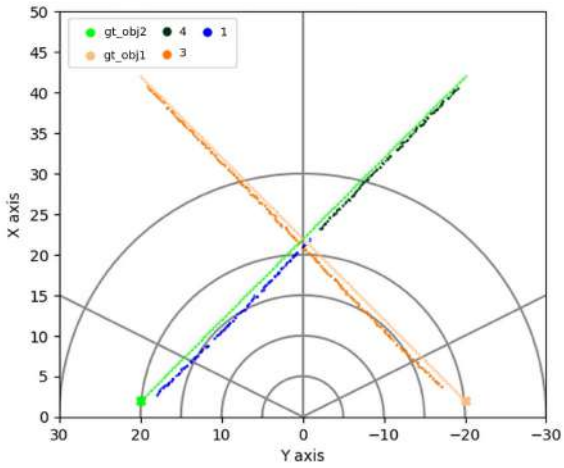


Figure 10: Objects on intersecting trajectories, with 40% misdetection probability.

The overall result is two-fold. On the one hand, the object has changed its "identity" (from 1 to 4, see the legend) and therefore we could not reconstruct the object's history, should we need to. On the other hand, the actual physical object has been undetected for a while, and this is surely a more severe problem for an ADAS system. In other words, this kind of error leads to identity changes between objects and apparently strange paths, posing challenges in the obstacle-evaluation algorithms of the system.

As a final note, it has to be pointed out that the previous effect does not only depend on the increase of the misdetection probability. The phenomena we have described above could also occur with a lower level of misdetections, since it all depends on the particular realization of the stochastic processes we are simulating in the test framework. For sure, higher levels of misdetections make that event more likely to occur, and this explains why we observed it in that particular experiment, but

there is a non-zero probability that such an event could occur in any of the synthetic test.

6 Conclusions and future work

Nowadays, the verification of the behavior of autonomous driving systems is critical for their deployment in everyday life and the importance of synthetic test environments is increasing. In this paper we tackled the problem of testing the behavior of ADAS systems throw a flexible tool for generating synthetic scenarios and evaluating KPIs. We have proposed a methodology for identifying test scenarios and presented a simulation framework for generating and running such scenarios and evaluating the system performance. Our scenario generation procedure uses a simulation model based on a simple scripting to describe the different scenarios, which are then run modeling sensors' behavior. Then, our tool can evaluate the overall system performance by aggregating execution results from multiple different runs. Our methodology brings complementary features compared to existing ones for the evaluation of object detection and tracking systems in ADAS systems, as it allows modular, effective, and repeatable simulation of representative scenarios with limited effort.

As we explained in a few examples in the previous section, our performance measurement tools allow us to make a very deep investigation about the problems that may relate to the Thales Italy ADAS system and, more in general, each autonomous driving system. Furthermore, after the data processing, through our tools it is possible to evaluate the results produced both qualitatively and quantitatively, using well-defined KPIs and graphical representations of the objects' trajectories. However, even if our framework works as expected, it needs to be tested more extensively, for assessing completely its functionality and optimize it. In this respect, thanks to a new feature we are developing, in the future also the tram with its motion will be represented in the scenarios. This will allow us to take into account also the possible trajectories of it in order to make our simulation tool more realistic and powerful. Moreover, we want to enrich the level of automation inside our scenario generation tool, exploiting a statistical approach to automatically derive the object motion parameters from real world use-cases, generating then many small variations of them to test the system behavior extensively. Lastly, we also want to compare our work with other strategies based on real-world data, trying to apply them jointly to exploit the strength points of each one.

References

[1] N. Kalra and S. M. Paddock, *Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?* Santa Monica, CA: RAND Corporation, 2016.

[2] C.-H. Yu, Y.-Z. Chen, and I.-C. Kuo, "The benefit of simulation test application on the development of autonomous driving system," in *2020 International Automatic Control Conference (CACS)*, 2020, pp. 1–5.

[3] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi, "Computing systems for autonomous driving: State of the art and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6469–6486, 2021.

[4] L. Li, W.-L. Huang, Y. Liu, N.-N. Zheng, and F.-Y. Wang, "Intelligence testing for autonomous vehicles: A new approach," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 158–166, 2016.

[5] A. Erdogan, E. Kaplan, A. Leitner, and M. Nager, "Parametrized end-to-end scenario generation architecture for autonomous vehicles," in *2018 6th International Conference on Control Engineering Information Technology (CEIT)*, 2018, pp. 1–6.

[6] E. de Gelder and J.-P. Paardekooper, "Assessment of automated driving systems using real-life scenarios," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 589–594.

[7] U. Lages, M. Spencer, and R. Katz, "Automatic scenario generation based on laserscanner reference data and advanced offline processing," in *2013 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, 2013, pp. 146–148.

[8] C. Medrano-Berumen and M. I. Akbaş, "Abstract simulation scenario generation for autonomous vehicle verification," in *2019 SoutheastCon*, 2019, pp. 1–6.

[9] E. Rocklage, H. Kraft, A. Karatas, and J. Seewig, "Automated scenario generation for regression testing of autonomous vehicles," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 476–483.

[10] P. Hyde, C. Ulianov, J. Liu, M. Banic, M. Simonovic, and D. Ristic-Durrant, "Use cases for obstacle detection and track intrusion detection systems in the context of new generation of railway traffic management systems," *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, vol. 236, no. 2, pp. 149–158, 2022.

[11] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[12] R. Stiefelhagen, K. Bernardin, R. Bowers, J. Garofolo, D. Mostefa, and P. Soundararajan, "The clear 2006 evaluation," vol. 4122, 04 2006, pp. 1–44.