




Specification Based Testing of Object Detection for Automated Driving Systems via BBSL

Kento Tanaka¹^a, Toshiaki Aoki¹^b, Tatsuji Kawai¹^c, Takashi Tomita¹^d, Daisuke Kawakami²
and Nobuo Chida²

¹*Japan Advanced Institute of Science and Technology, 1-1, Asahi-dai, Nomi, Ishikawa, 923-1292, Japan*

²*Advanced Technology R&D Center, Mitsubishi Electric Corporation,
8-1-1, Tsukaguchi-Honmachi, Amagasaki, Hyogo, 661-8661, Japan*

Keywords: Automated Driving, Machine Learning, Deep Learning, Object Detection, Testing, Formal Specification, Image Processing.


Abstract: Automated driving systems(ADS) are major trend and the safety of such critical system has become one of the most important research topics. However, ADS are complex systems that involve various elements. Moreover, it is difficult to ensure safety using conventional testing methods due to the diversity of driving environments. Deep Neural Network(DNN) is effective for object detection processing that takes diverse driving environments as input. A method such as Intersection over Union (IoU) that defines a threshold value for the discrepancy between the bounding box of the inference result and the bounding box of the ground-truth-label can be used to test the DNN. However, there is a problem that these tests are difficult to sufficiently test to what extent they meet the specifications of ADS. Therefore, we propose a method for converting formal specifications of ADS written in Bounding Box Specification Language (BBSL) into tests for object detection. BBSL is a language that can mathematically describe the specification of OEDR (Object and Event Detection and Response), one of the tasks of ADS. Using these specifications, we define specification based testing of object detection for ADS. Then, we evaluate that this test is more safety-conscious for ADS than tests using IoU.


1 INTRODUCTION


Automated driving systems (ADS) are actively developed by several manufacturers and their failure can cost human life (Devi et al., 2020). Therefore, ensuring their safety has become one of the most important research topics. However, ADS are complex systems including various elements, such as machine learning, route search algorithm and sensing technology. Furthermore, the driving environment surrounding those systems is diverse. Therefore, it becomes difficult to design specifications and test them as in general software development methods. To solve this problem, government agencies in various countries are researching frameworks for designing and testing ADS by defining and designing multiple scenarios of driving environments, systematizing use cases, setting


safety standards, and establishing evaluation frameworks. For example, the National Highway Traffic Safety Administration (NHTSA) in the United States first determines the level of automation of the automated driving system to be developed, and then develops the Operational Design Domain(ODD) and the Object and Event and Response (OEDR) are clarified. ODD is the specific conditions under which ADS or its functions are designed to operate, such as road types, speed limits, lighting conditions, weather conditions, and other operational constraints. The various driving environments are then classified into somewhat abstract scenarios, such as "merging into another lane at low speed," and OEDRs are designed based on these scenarios to monitor the driving environment and respond appropriately to these objects and events (Thorn et al., 2018). These levels of automation, ODDs, and OEDRs are defined in the SAE J3016 standard (Committee, 2021).

Among the components of ADS with the above characteristics, our research focuses on the object de-

^a <https://orcid.org/0000-0002-3532-6954>

^b <https://orcid.org/0000-0002-1209-6375>

^c <https://orcid.org/0000-0003-1247-5663>

^d <https://orcid.org/0000-0003-1249-7862>

tection process. In object detection for ADS, deep neural networks (DNN) are used to cope with large amounts of input. As shown in Figure 1, this DNN typically takes an image as input and a labeled rectangle called bounding box as the output of the inference result. The general approach to testing such a DNN is to match the inferred label by the DNN with the ground-truth label, given an image. However, it is difficult for DNN to detect the position of a particular object in perfect agreement with its ground-truth label. Therefore, in practice, it is tested using a threshold called Intersection over Union (IoU) (Everingham and Winn, 2012). IoU is a number that quantifies the degree of overlap between two boxes. In the case of object detection, IoU evaluates the overlap of the ground-truth label and inferred label. For example, Figure 2 shows two images with a ground-truth label (red bounding box) and an inferred label (green bounding box). In this case, the IoU is about 1/3 in both images. However, given a safety requirement to stop when there is a vehicle in the direction of travel, the DNN that infers the green bounding box in the left image violates the safety requirement, while the DNN that infers the green bounding box in the left image satisfies it. The above shows that the test of object detection using IoU is at variance with the specifications for ADS. Therefore, it is necessary to study specification-based testing methods for ADS.



Figure 1: DNN inputs and outputs.



Figure 2: Problems when thresholds are defined in the IoU.

In order to achieve specification based testing of the object detection process, a rigorously defined specification of how the system should operate in a given driving environment is required. Among the tasks of automated driving systems, there is a language called BBSL for writing specifications for OEDR (Tanaka et al., 2022). BBSL is a language that can describe specifications mathematically by representing objects such as other vehicles and pedestrians as bounding boxes and using positional relationships between bounding boxes. The specification based testing proposed in this paper is a method for specification based testing of the object detection process, which has a particular impact on safety among the components of automated driving systems. Us-

ing a simple example specification, we evaluate that the test is specification based and includes important safety contexts that cannot be considered in conventional IoU based testing methods.

2 RELATED WORK

A common ADS is composed of four functional modules, namely, the sensing module, the perception module, the planning module and the control module in Figure 3. The purpose of our study is to judge whether a bug exists in the perception module of the perception of these.

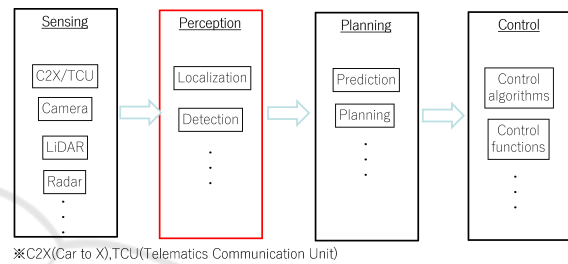


Figure 3: Typical architecture of an ADS.

The general approach of testing a DNN in the perception module is to match the inferred label by the DNN with the ground-truth label, given an image. Usually, these ground-truth labels are obtained by manual labeling (Sun et al., 2019) (Kondermann et al., 2016). Then, IoU, also known as Jaccard index, is used as a threshold to determine whether the positional descriptions of the ground-truth-label and inferred-label match or not. IoU computes the discrepancy between the bounding box of the inference result and the bounding box of the ground-truth-label, but in our study, we use the specification of ADS and compute it based on whether the specification is satisfied.

In perception testing, due to the huge input space of the DNN models, it is a great challenge to specify the oracles for all the input images. One solution to this problem is metamorphic testing. Metamorphic testing was introduced to tackle the problem when the test oracle is absent in traditional software testing (Chen et al., 2020). This test describes the system functionality in terms of generic relations (metamorphic relations) between inputs rather than as mappings between input and output. In ADS, Various metamorphic relations have been proposed, over images and frames in a scenario. For example, in object detection, there is a metamorphic relation that objects detected in the original image should also be detected in the synthetic images (Shao, 2021), and

for LiDAR based object detection, there is a metamorphic relation over the image that the noise points outside the Region of Interest (ROI) should not affect the detection of objects within the ROI (Zhou and Sun, 2019). Also, two metamorphic relations over frames in a scenario are proposed to respectively for identifying temporal and stereo inconsistencies that exist in different frames of a scenario (Ramanagopal et al., 2018). The temporal metamorphic relation says that an object detected in a previous frame should also be detected in a later frame, and the stereo metamorphic relation is defined in a similar way, for regulating the spatial consistency of the objects in different frames of a scenario.

Recently, temporal logics based formal specifications have been adopted in the monitoring of the perception module of ADS. In general, temporal logics are a family of formalism used to express temporal properties of systems. For example, a new form called Timed Quality Temporal Logic (TQTL), which can be used to express temporal properties that should be held by the perception module during object detection (Dokhanchi et al., 2018). Conceptually, the properties expressed by TQTL are similar to the ones in the metamorphic relations as mentioned above (Ramanagopal et al., 2018). however, by adopting such a formal specification to express these properties, one can synthesize a monitor that automatically checks the satisfiability of the system execution. TQTL is later extended to Spatio-Temporal Quality Logic (STQL) (Balakrishnan et al., 2021), which has enriched syntax to express more refined properties over the bounding boxes used in object detection. In our study, we test the perception module of ADS using a formal specification called BBSL. BBSL does not currently use temporal logics, so temporal properties cannot be expressed. However, BBSL can express properties related to position on the bounding box in more detail than STQL.

3 BOUNDING BOX SPECIFICATION LANGUAGE

The specification-based test proposed in this paper uses specifications written in a formal specification language called BBSL (Tanaka et al., 2022). BBSL is a language that describes objects in an image as bounding boxes and the positional relationships between the bounding boxes at a level of abstraction that can be defined manually. By defining a bounding box as a two-dimensional interval in interval analysis (Moore et al., 2009), BBSL strictly describes positional relationships on an image as relationships on a

bounding box (or set of bounding boxes). In addition, special relations and functions are defined for positional relations that cannot be described by interval analysis. In this section, we show the types and relations of BBSL and outline the specifications of ADS written in BBSL.

First, we explain the basic types used in BBSL, *interval* and *bb*. The *interval type* represents an interval like decelerating distance in Figure 4. This interval has the same definition as that defined in interval analysis. This is shown in Definition 1.



Figure 4: Visual representation of the driving environment and abstract specifications written in BBSL.

Definition 1. Let \underline{a}, \bar{a} be real numbers. Then an interval a is defined as follows:

$$a = [\underline{a}, \bar{a}] = \{x \in \mathbb{R} : \underline{a} \leq x \leq \bar{a}\}$$

Objects in the image, such as vehicles, are represented by bounding boxes defined by a two-dimensional interval. The *bb* type represents a bounding box, which has the same definition in interval analysis. This is shown in Definition 2.

Definition 2. Let $a_i, i = 1, \dots, n$ be intervals. Then a multidimensional interval a is defined as follows:

$$a = (a_1, \dots, a_n)$$

In particular, when $n = 2$, the multidimensional interval a is called the bounding box.

The magnitude relationship of intervals has the same definition as the interval analysis. However, since interval analysis is defined to calculate real values with rounding errors and measurement errors, many relationships cannot be applied to the description of positional relationships. Therefore, BBSL provides unique operations and relationships that are useful for describing image specifications.

First, we introduce the basic relations used to describe the positional relationship between intervals. These definitions are the same as those used in interval analysis and are shown in Definition 3, Definition 4.

Definition 3. Let a, b be intervals. Then the binary relation $<$ on two intervals is defined as follows:

$$a < b \Leftrightarrow \bar{a} < \underline{b}$$

Definition 4. Let a, b be intervals. Then the equivalence relation $=$ on two intervals is defined as follows:

$$a = b \Leftrightarrow \underline{a} = \underline{b} \text{ and } \bar{a} = \bar{b}$$

With these relationships, it is possible to represent the positional relationship between the vehicle's y-coordinate interval and the stoppingDistance defined as the distance to be maintained, as shown in Figure 5.



Figure 5: Example of simple positional relationships.

The definition of the inclusion relationship between intervals is shown in Definition 5.

Definition 5. Let a, b be intervals. Then the inclusion relation \subseteq on two intervals is defined as follows:

$$a \subseteq b \Leftrightarrow \underline{b} \leq \underline{a} \text{ and } \bar{a} \leq \bar{b}$$

For the positional relationship between these intervals, \approx is introduced as a shorthand notation to express the relationship where two intervals overlap, which frequently occurs in the specification of automated driving systems. This is shown in Definition 6.

Definition 6. Let a, b be intervals. Then the overlap relation \approx on two intervals is defined as follows:

$$a \approx b \Leftrightarrow \underline{b} \leq \bar{a} \text{ and } \underline{a} \leq \bar{b}$$

In addition, a function called *PROJ* function is provided to map the object to the x- and y-axis side intervals. This is shown in Definition 7.

Definition 7. Let a be bounding box (a_x, a_y) . Then the projection function $PROJ_i$ from bounding box to interval is defined as follows:

$$PROJ_i(a) = a_i, PROJ_{\bar{i}}(a) = [\bar{a}, \bar{a}], PROJ_{\underline{i}}(a) = [\underline{a}, \underline{a}], i \in \{x, y\}$$

BBSL can describe various positional relationships strictly using the *PROJ* function and interval relationships described above for the bounding boxes representing objects. For example, Figure 6 shows some examples of a positional relationship for two bounding boxes a and b described in BBSL.

By using such types and functions, BBSL can describe OEDR specifications for ADS strictly from an image perspective. Listing 1 shows an example of a specification described in BBSL. This specification defines the cases in which ADS should or should not

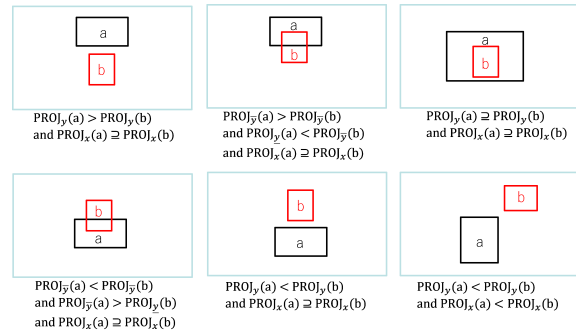


Figure 6: Examples of various positional relationships that can be distinguished using the *PROJ* function.

stop, depending on the position of a single vehicle. The specification described in BBSL is divided into three blocks, as shown in this specification. The first block, *external function block* (lines 1-8 of Listing 1), defines functions to receive values needed in advance to write this specification. For example, the specification provides a function to check for the presence of a vehicle, a function to return the bounding box surrounding the vehicle, and a function to return an interval representing the stopping distance.

The second block, the *precondition block* (lines 10-12 in Listing 1), is used to describe the conditions for applying the specification. For example, it states that the specification is written on the assumption that there will always be a vehicle of some kind.

The third block, *case block* (lines 14-19 and 21-25 in Listing 1), is the main part of this specification. It describes a case for each reaction of the ADS and the conditions under which the system should react. This means that in the list 1, the specification strictly describes the need to stop if a vehicle is before or overlaps the stoppingDistance from the viewpoint of the forward camera image, as in Figure 5.

Listing 1: Specifications of ADS response to the distance to the vehicle.

```

1 exfunction
2   //Judge the existence of the vehicle.
   True if it exists.
3   vehicleExists():bool
4   //Calculate the bounding box that
   surrounds the vehicle.
5   vehicle():bb
6   //Calculate the interval that represents
   the range to be stopped.
7   stoppingDistance():interval
8 endexfunction
9
10 precondition
11   [vehicleExists() = true]
12 endprecondition
13
14 case stop
15   let vehicle : bb = vehicle(),
16   stoppingDistance : interval =
   stoppingDistance() in
17   PROJ_y(vehicle) ≈ stoppingDistance
    
```

```

18     or PROJy(vehicle) < stoppingDistance
19endcase
20
21case NOT stop
22   let vehicle : bb = vehicle(),
23   stoppingDistance : interval =
     stoppingDistance() in
24     PROJy(vehicle) > stoppingDistance
25endcase

```

4 SPECIFICATION BASED TESTING

As described in the previous section, specifications for ADS written using BBSL are unambiguous and rigorously described. Therefore, specifications written in BBSL can be used to rigorously define specification-based tests for object detection processing. In this section, we make some preparations and define specification-based tests.

The specification written in BBSL represents images with multiple labeled boundingboxes, as shown in Figure 7. First, the set of images with such labeled boundingboxes is shown in Definition 11.



Figure 7: Examples of Image with multiple boundingboxes labeled.

Definition 8. Let I be a set of images, BB be a set of boundingboxes, and L be a set of labels such as vehicle and pedestrian. The set of images I_{BB} with multiple boundingboxes with labels is defined as follows:

$$I_{BB} = I \times 2^{BB \times L}$$

Next, a specification written in BBSL for use in testing is defined in Definition 9.

Definition 9. A specification written in BBSL is defined as a pair $S = (C, f)$.

- C is the set of cases. For example, in Listing 1, $C = \{stop, NOT\ stop\}$.
- Let f be defined by the function $f : I_{BB} \rightarrow 2^C$ where $I_{BB} = \{i_{BB} \in I_{BB} \mid i_{BB} \text{ satisfies precondition condition.}\}$.

Hereafter, I_{BB} of f in the specification $S = (C, f)$ is denoted as $dom(f)$.

For the purpose of preparing a test covering all images or defining a unique test, the three types of

properties on the specification written in BBSL are defined in Definition 10.

Definition 10. For a specification $S = (C, f)$ written in BBSL, the properties of the three types of specifications are defined as follows:

S is an exhaustive specification

$$\Leftrightarrow \forall i \in dom(f). (f(i) \neq \emptyset)$$

S is an exclusionary specification

$$\Leftrightarrow \forall i \in dom(f), \exists c \in C.$$

$$(f(i) = \{c\} \text{ or } f(i) = \emptyset)$$

S is a non-redundant specification

$$\Leftrightarrow \forall c \in C, \exists i \in dom(f). (c \in f(i))$$

Thus, the specification of Listing 1 is an exhaustive, exclusionary, and non-redundant specification. In this paper, unless otherwise specified, specification S written in BBSL is an exhaustive, exclusionary, and non-redundant specification.

Next, the definitions of the basic elements necessary to define a test are given in Definition 11.

Definition 11. The test data Td is a subset of the set of images I . The object detection system to be tested is exactly the DNN shown in Figure 1, which takes an image as input and returns an image with inferred labels as output. This is defined by the function $SUT : I \rightarrow I_{BB}$. In addition, assume that the ground-truth label is given by the function $GT : I \rightarrow I_{BB}$.

Using the above definitions, a test case is defined in Definition 12.

Definition 12. Given an exhaustive, exclusionary, and non-redundant specification $S = (C, f)$, test data Td and ground-truth data GT , test case $CASE$ is defined as follows:

$$CASE = \{(td, f(GT(td))) \mid td \in Td\}$$

In Definition 12., $f(GT(td))$ plays the role of a specification-based pseudo-oracle.

Finally, the decision conditions for the specification-based test are defined in Definition 13.

Definition 13. Given an exhaustive, exclusionary, and non-redundant specification $S = (C, f)$, SUT and a test case $CASE$, the test decision condition $P : CASE \rightarrow \{T, F\}$ is defined for any case $(td, c) \in CASE$ is defined as follows:

$$P(td, c) = \begin{cases} T & f(SUT(td)) = c \\ F & \text{otherwise} \end{cases}$$

The above definitions enabled specification-based testing using specifications written in BBSL.

5 EXPERIMENT

In this section, we actually prepare an object detection system for ADS, a grand tuluth dataset with two-dimensional bounding box, and a specification written in BBSL, and compare the proposed test method with the IoU method.

5.1 Preparations

First, we used the KITTI dataset (Geiger et al., 2013) for Td_1 and Td_2 ($Td_1 \cap Td_2 = \emptyset$) as the two types of test data set and GT as the grand truth label. These are 349 and 1300 images from the forward camera of ADS, respectively, as shown in Table 1. Each image contains one or more vehicles, and the number of vehicles in the dataset is 2736 and 5644, respectively. The specifications of these grand tuluth labels are given by enclosing each vehicle in a two-dimensional bounding box.

Table 1: Test Data Details.

Name	Number of images	Number of vehicles
Td_1	349	2736
Td_2	1300	5644

Next, two object detection systems to be tested are prepared as SUT_1 and SUT_2 , respectively. Both object detection algorithms used in these systems are based on Yolov3 (Redmon and Farhadi, 2018), and the DNN network used is darknet53. In our study, we prepared two types of object detection systems by using yolov3-kitti.weights(<https://drive.google.com/file/d/1BRJDDCMRXdQdQs6-x-3PmlzcEuT9wxJV/view>) for SUT_1 and yolov3.weights(<https://github.com/patrick013/Object-Detection---Yolov3/blob/master/model/yolov3.weights>) for SUT_2 from publicly available weighting files, without training them independently.

In addition, we prepared four simple specifications written in BBSL as S_1 , S_2 , S_3 and S_4 on which to base our tests. S_1 is Listing 1 already described above as an example, which defines the cases in which ADS should or should not stop depending on the distance of the vehicle in front. S_2 is shown in Listing 2. This specification defines the cases in which ADS should and should not stop depending on whether the target vehicle encroaches into the linear distance of the own vehicle or not.

Listing 2: Specifications of ADS response to the position of x-axis the vehicle.

```

1exfunction
2  //Judge the existence of the vehicle.
   True if it exists.
3  VehicleExists():bool

```

```

4  //Calculate the bounding box that
   surrounds the vehicle.
5  Vehicle():bb
6  //Calculate the interval that represents
   the range to be stopped.
7  directionAreaDistance():interval
8endexfunction
9
10precondition
11  [VehicleExists() = true]
12endprecondition
13
14case stop
15  let Vehicle : bb = lVehicle(),
16  directionAreaDistance : interval =
   directionAreaDistance() in
17  PROJx(Vehicle) ≈
   directionAreaDistance
18endcase
19
20case NOT stop
21  let Vehicle : bb = Vehicle(),
22  directionAreaDistance : interval =
   directionAreaDistance() in
23  not (PROJx(Vehicle) ≈
   directionAreaDistance)
24endcase

```

S_3 is shown in Listing 3. This specification combines S_1 and S_2 , and specifies that it is a case of Stop if the distance between vehicles is close and the vehicle has entered the travel direction, and a case of Not Stop otherwise.

Listing 3: Specifications with two cases combining S_1 and S_2 .

```

1exfunction
2  vehicleExists():bool
3  vehicle():bb
4  directionAreaDistance():interval
5  stoppingDistance():interval
6endexfunction
7
8precondition
9  [vehicleExists() = true]
10endprecondition
11
12case stop
13  let vehicle : bb = vehicle(),
14  directionAreaDistance : interval =
   directionAreaDistance(),
15  stoppingDistance : interval =
   stoppingDistance() in
16  PROJx(vehicle) ≈
   directionAreaDistance
17  and PROJy(vehicle) ≈
   stoppingDistance
18endcase
19
20case NOT stop
21  let vehicle : bb = vehicle(),
22  directionAreaDistance : interval =
   directionAreaDistance() in
23  not (PROJx(vehicle) ≈
   directionAreaDistance) or
24  not (PROJy(vehicle) ≈
   stoppingDistance)
25endcase

```

Finally, S_4 is shown in Listing 4. Like S_3 , this specification is a combination of S_1 and S_2 . The condition of x_yStop is the same as that of S_3 , but the

condition of Not Stop is divided into `ySAFE_xwarning`, `xSAFE_ywarning` and `NOT warning` in more detail depending on the relationship between the `y` and `x` coordinates on the image.

Listing 4: Specifications with four cases combining S_1 and S_2 .

```

1 exfunction
2   vehicleExists():bool
3   vehicle():bb
4   directionAreaDistance():interval
5   stoppingDistance():interval
6 endexfunction
7
8 precondition
9   [vehicleExists() = true]
10 endprecondition
11
12 case x_ystop
13   let vehicle : bb = vehicle(),
14     directionAreaDistance : interval =
15     directionAreaDistance(),
16     stoppingDistance : interval =
17     stoppingDistance() in
18     PROJx(vehicle) ≈
19     directionAreaDistance
20     and PROJy(vehicle) ≈
21     stoppingDistance
22 endcase
23
24 case ySAFE_xwarning
25   let directionAreaDistance : interval =
26     directionAreaDistance(),
27     stoppingDistance : interval =
28     stoppingDistance() in
29     PROJx(vehicle) ≈
30     directionAreaDistance
31     and not (PROJy(vehicle) ≈
32     stoppingDistance)
33 endcase
34
35 case xSAFE_ywarning
36   let directionAreaDistance : interval =
37     directionAreaDistance(),
38     stoppingDistance : interval =
39     stoppingDistance() in
40     not (PROJx(vehicle) ≈
41     directionAreaDistance)
42     and PROJy(vehicle) ≈
43     stoppingDistance
44 endcase
45
46 case NOT warning
47   let vehicle : bb = vehicle(),
48     directionAreaDistance : interval =
49     directionAreaDistance() in
50     not (PROJx(vehicle) ≈
51     directionAreaDistance) and
52     not (PROJy(vehicle) ≈
53     stoppingDistance)
54 endcase

```

Each of these specifications is interpreted by implementing the conditions in Python using BBSL semantics. Since all of the specifications described here describe conditions for a single vehicle object, the test is interpreted for only one vehicle for each image with multiple vehicles in it. Therefore, the number of test cases is 2736,5644, which is the number of vehicle objects in Td_1 and Td_2 , respectively.

The implementation of each exfunction was given as a constant. The values are those of the coordinates with the upper left corner as 0 in all images (size is 1242×375), and the values are shown in Table 2. In Table 2, $sD()$ in the *Given Exfunction column* stands for *stoppingDistance()* and $dA()$ for *directionAreaDistance*.

Table 2: Details of all tests.

Test ID	S	Given Exfunctions	Td	SUT
ID1	S_1	$sD()$ = [275, 375]	Td_1	SUT_1
ID2	S_1	$sD()$ = [275, 375]	Td_1	SUT_2
ID3	S_1	$sD()$ = [250, 375]	Td_1	SUT_1
ID4	S_1	$sD()$ = [300, 375]	Td_1	SUT_1
ID5	S_2	$dA()$ = [420, 821]	Td_1	SUT_1
ID6	S_3	$sD()$ $dA()$ = [275, 375] = [420, 821]	Td_1	SUT_1
ID7	S_4	$sD()$ $dA()$ = [275, 375] = [420, 821]	Td_1	SUT_1
ID8	S_3	$sD()$ $dA()$ = [275, 375] = [420, 821]	Td_1	SUT_2
ID9	S_4	$sD()$ $dA()$ = [275, 375] = [420, 821]	Td_1	SUT_2
ID10	S_1	$sD()$ = [275, 375]	Td_2	SUT_1
ID11	S_2	$dA()$ = [420, 821]	Td_2	SUT_1

5.2 Evaluation

The judgment results of the proposed test and the judgment results with $IoU_{0.6}$ and $IoU_{0.8}$ on 11 different tests are shown in Table 3. It can be seen that, unlike the IoU calculated only from the test data and SUT, the proposed method changes its judgment depending on the given specification. In addition, even though the test data sets Td_1 and Td_2 were not prepared artificially, it is clear that there is a large discrepancy between the IoU test and the proposed test.

Furthermore, for test ID1, we measured the judgment result in the case of $IoU_{0.6}$ and the judgment

Table 3: Tests results.

Test ID	$IoU_{0.6}$ ($T/T+F$)	$IoU_{0.8}$ ($T/T+F$)	suggestion test ($T/T+F$)
ID1	$2180/(2180+556) = 79.7\%$	$1432/(1432+1304) = 52.3\%$	$2527/(2527+209) = 92.4\%$
ID2	$1221/(1221+1515) = 44.6\%$	$791/(791+1945) = 28.9\%$	$1929/(1929+807) = 70.5\%$
ID3	$2180/(2180+556) = 79.7\%$	$1432/(1432+1304) = 52.3\%$	$2500/(2500+236) = 91.4\%$
ID4	$2180/(2180+556) = 79.7\%$	$1432/(1432+1304) = 52.3\%$	$2524/(2524+212) = 92.3\%$
ID5	$2180/(2180+556) = 79.7\%$	$1432/(1432+1304) = 52.3\%$	$2591/(2591+145) = 94.7\%$
ID6	$2180/(2180+556) = 79.7\%$	$1432/(1432+1304) = 52.3\%$	$2604/(2604+132) = 95.2\%$
ID7	$2180/(2180+556) = 79.7\%$	$1432/(1432+1304) = 52.3\%$	$2502/(2502+234) = 91.4\%$
ID8	$1221/(1221+1515) = 44.6\%$	$791/(791+1945) = 28.9\%$	$2065/(2065+671) = 75.5\%$
ID9	$1221/(1221+1515) = 44.6\%$	$791/(791+1945) = 28.9\%$	$1867/(1867+869) = 68.2\%$
ID10	$4842/(4842+802) = 85.8\%$	$3182/(3182+2462) = 56.4\%$	$5299/(5299+345) = 93.9\%$
ID11	$4842/(4842+802) = 85.8\%$	$3182/(3182+2462) = 56.4\%$	$5314/(5314+330) = 94.2\%$

result in the case of the proposed test for each case of expected value as shown in Figure 8. The aggregate results are shown in Table 4. The numbers to the right of ① to ⑧ shown in Table 4 are the number of applicable test cases, corresponding to ① to ⑧ in Figure 8, respectively. In Table 4 and Figure 8, BBSL refers to the proposed test, and expectation is c in this test case (td, c). As can be seen from the results, although the number of test cases is biased, there are test cases that correspond to all of them. In particular, the existence of test cases corresponding to ③, ④, ⑤, and ⑥ indicates that there are cases in which the proposed method makes decisions that differ from those of IoU. In addition, the existence of test cases corresponding to ⑥ and ⑧ indicates that the proposed test detects malfunctions such as not being able to stop when ADS should be stop based on the specification. The above indicates that the proposed method detects the test cases where the IoU is inadequate if these data are valid.

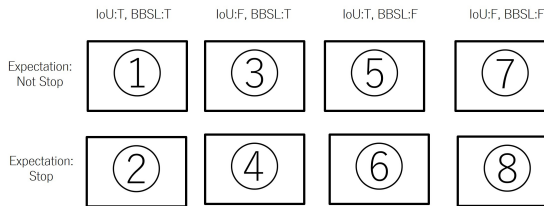


Figure 8: How to classify test cases in Test ID1.

6 DISCUSSION

As shown in Table 3, the proposed test can be performed on various test datasets as long as they are ground truth data with a two-dimensional bounding box. Since many test data for object detection systems have two-dimensional bounding boxes as ground truth data, the proposed test can be performed on many existing data sets. In addition, the

proposed method determines the tolerance for misalignment using an algorithm that is clearly different from IoU in that it determines whether a test case is acceptable or not based on the specifications written in the prepared BBSL.

Furthermore, we focus on each of the data in Table 4 and discuss its effectiveness based on each example. First, we discussed test case to ① on Figure 9, in other words, a case where the expected value is Not Stop in the specification, and where both the proposed test and $IoU_{0.6}$ are judged as T. The white vehicle in Figure 9 is the relevant test case, and the bounding box of the inferred result is indicated. Since this white vehicle has a sufficient distance from its own vehicle, the expected value is "Not Stop" based on the specification of Listing 1. The bounding box in the inferred result is determined to be T with $IoU_{0.6}$ because it is detected with almost no deviation. In such a case, there is clearly no defect in the specification of ADS, and the proposed test is judged to be T, which is reasonable.



Figure 9: Example of a test case corresponding to ① on Figure 8.

Next, we will discuss the test case classified as ②, in other words, an environment where the expected value is Stop in the specification, and Figure 10 is an example where the proposed test and $IoU_{0.6}$ are both determined to be T. The red vehicle in Figure 10 is the relevant test case, and the bounding box of the inferred result is shown. Since this vehicle is very close to its own vehicle, the expected value is Stop based on the specification of Listing 1. The bounding box in the inferred result has some visual deviation, but

Table 4: Test case classification results for test ID1.

expectation	$IoU_{0.6}:T, BBSL:T$	$IoU_{0.6}:F, BBSL:T$	$IoU_{0.6}:T, BBSL:F$	$IoU_{0.6}:F, BBSL:F$
Not stop	①1744	③345	⑤4	⑦74
stop	②404	④34	⑥28	⑧103

because it is a large object in the image, IoU is calculated to be high, and it is judged to be T with $IoU_{0.6}$. Since the direction of the misalignment is in the lateral direction and the distance recognition is correct, there is no defect in the specification of ADS, and the proposed test is also judged to be T, which is reasonable.



Figure 10: Example of a test case corresponding to ② on Figure 8.

Next, we will discuss the test case classified as ③, in other words, in other words, an environment with an expected value of Not stop in the specification, which is judged as T in the proposed test and F in the $IoU_{0.6}$ test. The vehicle surrounded by a red bounding box of grand-truth label hidden by a white vehicle and a building in the upper image of Figure 11 is the corresponding test case, and the bounding box of the inferred result is shown in the lower image. Since this vehicle is sufficiently far from own vehicle, the expected value is Not stop based on the specification of Listing 1. Since the object detection system under test cannot recognize the vehicle in the back and only detects the white car in the front, the IoU is very low and is determined to be F with $IoU_{0.6}$. However, if the white vehicle in the front can be correctly recognized and judged to have a sufficient distance, there is no defect in the specification base of ADS, and the proposed test is judged to be T, which is reasonable.

Next, we will discuss the test case classified as ④, in other words, an environment with an expected value of Stop in the specification, which is judged as T in the proposed test and F in the $IoU_{0.6}$ test. The black vehicle surrounded by a red bounding box in the image in Figure 12 is the relevant test case, and the inferred result is indicated by the purple bounding box. Since this vehicle is very close to the own vehicle, the expected value is Stop based on the specification of Listing 1. Since only the first part of the vehicle is shown in the image, the object detection system recognizes the vehicle as smaller than the actual size of the vehicle indicated by the red bounding box, and the IoU is very low and is determined to be F

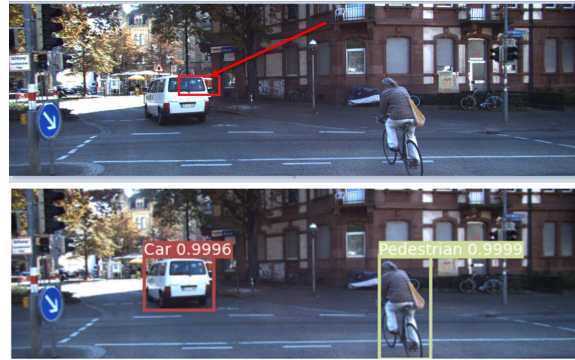


Figure 11: Example of a test case corresponding to ③ on Figure 8.

at $IoU_{0.6}$. However, most of the discrepancy is in the height of the vehicle, and there is almost no discrepancy in the recognition of the distance from own vehicle. Therefore, there is no defect in the specification base of ADS, and the proposed test is also determined to be T, which is reasonable.

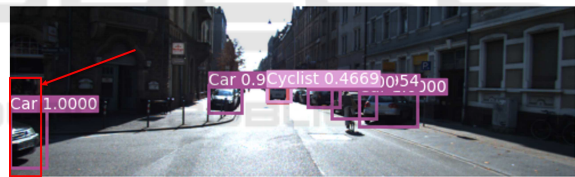


Figure 12: Example of a test case corresponding to ④ on Figure 8.

Next, we will discuss the test case classified as ⑤, in other words, in other words, an environment where the expected value is Not stop in the specification and the proposed test determines F and $IoU_{0.6}$ to be T. The red vehicle in the image in Figure 13 is the corresponding test case, and the inferred result is indicated by the purple bounding box. The red vehicle in the image in Figure 13 is the test case, and the inferred result is indicated by the purple bounding box. Since this vehicle is far away from the own vehicle, the expected value is "Not stop" based on the specification of Listing 1. And since this inferred result shows a slight downward of the misalignment but almost no upward or lateral of misalignment, the IoU is not low and is determined to be T at $IoU_{0.6}$. However, because of the downward of the misalignment, there is a possibility that the vehicle may stop in a situation where it is not necessary to stop due to the misalignment with respect to the recognition of the distance between the

vehicles. Therefore, this is a fault based on the specification that reduces the reliability of ADS, and the proposed test also judges it as F, which is reasonable.

Unlike the case of judging by IoU, the proposed test can reflect the direction of misalignment in the judgment criteria by using the specification written in BBSL. Therefore, as in the red vehicle in Figure 10, even if the degree of misalignment of the inferred result is large, it can be judged to be True on the specification basis. Conversely, even when the degree of misalignment of the inferred result is small, as in the case of the black vehicle in Figure 13, it can be determined to be false on the specification basis. This is an important feature of the proposed test.



Figure 13: Example of a test case corresponding to ⑤ on Figure 8.

Next, we will discuss the test case classified as ⑥, in other words, an environment where the expected value is Stop in the specification, and the proposed test determines F and $IoU_{0.6}$ to be T. The vehicle surrounded by the red bounding box on the upper image in Figure 14 is the corresponding test case, and the bounding box of the detection result is shown on the lower image. Since this vehicle is close to the own vehicle, the expected value is Stop based on the specification of Listing 1. The bounding box in the inferred result is determined to be T with $IoU_{0.6}$ because it is detected with almost no deviation. However, since the subtle misalignment of the distance between vehicles crosses the border of the stop condition in the specification, it is a defect based on the specification that reduces the safety of the automatic driving system, and is judged as F in the proposed test, which is reasonable. Thus, the proposed test strictly determines T or F for test cases around the boundary of case condition described in the specification. Since the boundary of case condition on the specification is a part that should be functionally tested especially carefully, it is an important feature of the proposed test that this part is rigorously tested.

Next, we will discuss the test case classified as ⑦, in other words, an environment where the expected value is Not Stop in the specification, and where the proposed test and $IoU_{0.6}$ are both judged as F. The red vehicle partially hidden by a building in the image in Figure 15 is a relevant test case. This vehicle has a



Figure 14: Example of a test case corresponding to ⑥ on Figure 8.

sufficient distance from the vehicle, so the expected value is Not stop based on the specification of Listing 1. Since the object detection system under test cannot recognize this red vehicle at all and there is no overlap with the inferred result of the white vehicle before it, IoU is 0 and $IoU_{0.6}$ is determined to be F. This is because Listing 1 is sufficiently far from the vehicle. This is an example where the inference result does not satisfy the conditions of the precondition block of Listing 1, while the ground truth data satisfy Not stop.



Figure 15: Example of a test case corresponding to ⑦ on Figure 8.

Finally, we will discuss the test case classified as ⑧, in other words, Figure 16, which is an environment where the expected value is Stop in the specification, and where both the proposed test and $IoU_{0.6}$ determine the value to be F. The vehicle on the far left in Figure 16 is the relevant test case, and since this vehicle is very close to its own vehicle, it is an expected value Stop based on the specification of Listing 1. And since the object detection system under test does not recognize this vehicle at all, IoU is 0 and $IoU_{0.6}$ is determined to be F. This is an example of a situation in which Listing 1 specification says Stop for ground truth data, but the car does not exist as a inferred result, and the precondition block condition of Listing 1 specification is no longer satisfied.

The examples shown in Figures 15 and 16 are both examples of cases where the inferred results fall outside scope of the specification. In the first example, the white vehicle before the vehicle is correctly recognized and judged to have a sufficient distance,

so there is no defect, and it is reasonable to judge the car to be T in the proposed test. However, in the second example, the system is not able to detect a vehicle at a position where it should stop due to a short distance between own vehicle and the vehicle, and this is a defect that reduces the safety of ADS. Thus, there are cases in which the inferred result is outside the scope of the specification even if the image is subject to the specification and the expected value can be defined, and it was not possible to clarify how to give aHEREHEREHEREaccurate judgment to these cases. Therefore, the proposed test gives priority to safety and defines both cases to be judged as F.



Figure 16: Example of a test case corresponding to ⑧ on Figure 8.

Based on the above discussion, the proposed test returns a valid decision result as a test of safety and reliability based on the specification. This is clearly different from the test method used to evaluate the performance of object detection systems such as IoU. Furthermore, it is an important test when incorporating an object detection system into a large piece of software that requires high reliability and high safety, such as an ADS.

7 CONCLUSIONS

By using the proposed test method, the object detection system of an ADS can be tested based on the specifications. Since the test is based on the degree to which the object detection system under test meets the specification when it is incorporated into an ADS with the relevant specification, the test is able to detect cases of impair safety or reliability defects that are not detected by conventional testing methods. For these reasons, our test is an important and innovative test for incorporating object detection systems into complex and safety critical software such as ADS.

Finally, we show three future works. The first is to formally verify specifications written in BBSL on theorem proving. Since BBSL has not yet been formalized in a theorem proving system, and no parser has been prepared, this study was programmed in python so that the implementation would be equivalent to the specification used in the experiments. This work is important for testing in larger, more realistic environ-

ments and will contribute to the development of real-time monitoring tools for object detection systems. The second is to extend specification-based testing with more complex ADS specifications described in BBSL. The tests experimented with in our study used only a simple specification for the relationship between a single object in the image and the own vehicle. However, the description capability of BBSL discussed in this paper is only part of the picture, and in practice it can describe the positional relationships of multiple objects and objects of complex shapes. We think that testing extensions to handle these specifications will contribute to the development of even more secure ADS. The third is to propose and evaluate coverage that correlates to the quality of the specification-based tests proposed in our study. It is not known how many and what kind of test cases are needed to sufficiently test the specification-based test proposed in our study. To increase the utility of this test, we believe it is necessary to propose validity index for test, for example, coverage on the position on the image and coverage on the conditions of the specification written in BBSL.

REFERENCES

- Balakrishnan, A., Deshmukh, J., Hoxha, B., Yamaguchi, T., and Fainekos, G. (2021). Percemon: Online monitoring for perception systems. *CoRR*, abs/2108.08289.
- Chen, T. Y., Cheung, S. C., and Yiu, S. (2020). Metamorphic testing: A new approach for generating next test cases. *CoRR*, abs/2002.12543.
- Committee, O.-R. A. D. O. (2021). *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*.
- Devi, S., Malarvezhi, P., Dayana, R., and Vadivukkarasi, K. (2020). A comprehensive survey on autonomous driving cars: A perspective view. *Wirel. Pers. Commun.*, 114(3):2121–2133.
- Dokhanchi, A., Amor, H. B., Deshmukh, J. V., and Fainekos, G. (2018). Evaluating perception systems for autonomous vehicles using quality temporal logic. In Colombo, C. and Leucker, M., editors, *Runtime Verification*, pages 409–416, Cham. Springer International Publishing.
- Everingham, M. and Winn, J. (2012). The pascal visual object classes challenge 2012 (voc2012) development kit. *Pattern Anal. Stat. Model. Comput. Learn., Tech. Rep.*, 2007:1–45.
- Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.
- Kondermann, D., Nair, R., Honauer, K., Krispin, K., Andrusis, J., Brock, A., Güssefeld, B., Rahimimoghaddam, M., Hofmann, S., Brenner, C., and Jähne, B.

- (2016). The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 19–28.
- Moore, R. E., Kearfott, R. B., and Cloud, M. J. (2009). *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics.
- Ramanagopal, M. S., Anderson, C., Vasudevan, R., and Johnson-Roberson, M. (2018). Failing to learn: Autonomously identifying perception failures for self-driving cars. *IEEE Robotics and Automation Letters*, 3(4):3860–3867.
- Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement.
- Shao, J. (2021). Testing object detection for autonomous driving systems via 3d reconstruction. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 117–119.
- Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. (2019). Scalability in perception for autonomous driving: Waymo open dataset. *CoRR*, abs/1912.04838.
- Tanaka, K., Aoki, T., Kawai, T., Tomita, T., Kawakami, D., and Chida, N. (2022). A formal specification language based on positional relationship between objects in automated driving systems. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 950–955.
- Thorn, E., Kimmel, S. C., and Chaka, M. (2018). A framework for automated driving system testable cases and scenarios.
- Zhou, Z. Q. and Sun, L. (2019). Metamorphic testing of driverless cars. *Commun. ACM*, 62(3):61–67.