K-DAsFault: Kaistian-Designed Automatically Testing Self-Driving Cars with Search-Based Procedural Content Generation

JAEMIN CHO[1], GEON KIM[2], JAEUK KIM[3], JIIN KIM[4], AND SEUNGHEE HAN[5]

[1]*20160633, School of Computing, Korea Advanced Institute of Science and Technology(KAIST), Daejeon 34141, Korea*
[2]*20170057, School of Computing, Korea Advanced Institute of Science and Technology(KAIST), Daejeon 34141, Korea*
[3]*20170150 , School of Electrical Engineering, Korea Advanced Institute of Science and Technology(KAIST), Daejeon 34141, Korea*
[4]*20170168 , School of Electrical Engineering, Korea Advanced Institute of Science and Technology(KAIST), Daejeon 34141, Korea*
[5]*20170719, School of Electrical Engineering, Korea Advanced Institute of Science and Technology(KAIST), Daejeon 34141, Korea*
[*]*Team Number: 21*
[*]*Project corresponds to Option 2 (Replicating and/or Improving Literature)*

Autonomous driving industry has become one of the most valuable industries. In this area, finding possible accident of autonomous vehicle has become important issue. However, testing in real traffic is not only dangerous but also expensive. So, the virtual test method is emerging as a good alternative. AsFault is the system which combined Procedural Content Generation (PCG) and Genetic Algorithm (GA) to test lane-keeping functionality of self-driving car. In our project, we replicated the AsFault by implementing our own PCG and GA to automatically generate and evolve various road maps. Then, we evaluated our system using MATLAB Simulink autonomous driving toolbox. In addition, we proposed K-DAsFault to introduce novel methods on search operator and similarity test. Our evaluation showed K-DAsFault can produce effective road map which not only make more Out of Bound Episodes (OBE) but also cause more vision error of lane-keeping functionality of Simulink simulation than randomly generated road map.
Our team's work (including codes, presentation slides, and video) is available at link below:

https://github.com/shhan1755/CS454-Final-Project

1. INTRODUCTION

Nowadays, with advances in AI technique, the autonomous driving industry has become one of the most valuable industries. A number of auto and software companies, including Tesla, Waymo and Google, are making significant investments in autonomous vehicle research. The most important part of autonomous vehicle is safety. So, to make completely safe autonomous driving system, finding possible accident of autonomous vehicle has become important issue. The most intuitive way to find fault of self-driving cars is to test them in real traffic. However, real traffic testing method has some drawbacks. At first, real traffic testing is dangerous. Incomplete self-driving system can cause some terrible accidents. And, it is expensive and time consuming. To find defects in self-driving cars, hundreds or thousands of tests are needed. Driving on the actual road requires a great deal of time to carry out the process. Therefore, virtual test, test using computer simulation, can be a good alternative. Nevertheless, finding the way to create a lot of suitable test scenarios is difficult problem. To solve this drawback of virtual test, Alessio, Marc, and Gordon proposed AsFault.

AsFault is automated system that tests lane-keeping functionality of self-driving car. They focused on testing lane-keeping system and merged Procedural Content Generation (PCG) and Genetic Algorithm (GA) so that they proposed new system to automatically create random road map and evolve road map to expose problems in self-driving software. In our project, our main objective is to replicate AsFault to make the system which test the lane-keeping functionality of autonomous vehicles.

Thus, we implemented testing road map generator using PCG and search-based road evolving system using GA. Through the implemented system, we wanted to automatically generate virtual road maps which effectively expose faults in lane-keeping system of autonomous vehicle. Then, to evaluate our system, we applied MATLAB Simulink autonomous driving tool box and checked out of bound episodes (OBE) in each simulation. Moreover, in this paper, we proposed K-DAsFault which is improved version of AsFault to introduce novel methods on search operator and similarity test.

2. SYSTEM IMPLEMENTATION

In this part, we described implementation of our system. Main objective of our project is to generate effective test data to improve the lane keeping system of self-driving car. In other words, what we want is to generate the pragmatically synthesized road network which induce failure of lane keeping system in self-driving cars. To achieve our purpose, we introduced two main functionalities to our system, Procedural content generation (PCG) and Genetic algorithm (GA). In this part, we explained PCG first, then described GA.

A. Procedural Content Generation

Procedural content generation (PCG) is a technique mainly used in the game industry. It means creation of game content automatically using algorithms. In our project, we decided to utilize the procedural content generation for spawning initial road network populations. In section A, we explained the way how we create the initial random road network using PCG. Then, in section B, we explained our validation method to check whether generated road and road network is valid or not.

A.1. Road network Generation

In this project, we generate random road network using PCG technique. Road network generation process is an incremental process from road segments to road networks. In this section, we firstly explained definition of a road. Then, we described detailed road network generation process step by step.

Road Definition

The smallest unit of road network is road segment. If multiple road segments are gathered, they become a road, then if multiple roads are accumulated, a single road network is formed. Figure 1 show examples of road segment. In our project, we represented each road segment as five different poly-lines. Back line is start line of road segment which is expressed as magenta line. Front line is end line of segment expressed as orange line. Left edge, center line, and right edge are internal lines which connect the back line and front line of road segment, they are expressed as red, blue, and green line, respectively. In this project, we defined two types of road segment. Straight road segment is segment to generate short straight road. It has three properties, width, length and the number of points in internal line (figure.1.a). Arc road segment is segment to generate short curved road. Properties of arc road segment are radius of curvature, rotation angle, the number of points in internal line (figure.1.b).

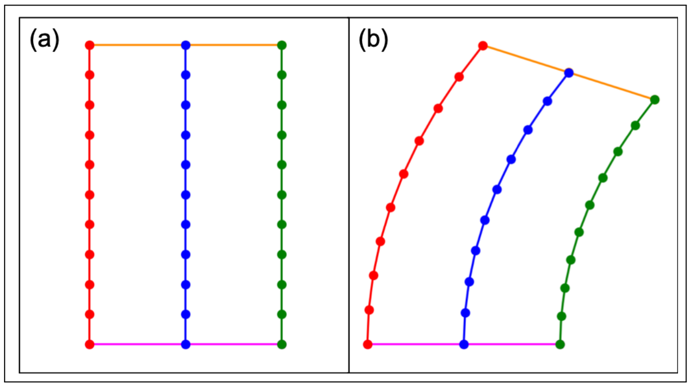

Figure. 1. Road segment example.
(a) and (b) are example of road segment which are defined by multiple poly-lines. Magenta and orange line is back line and front line of road segment. And red, blue and green lines are left edge, center line and right edge respectively. (a) Example of straight road segment. (b) Example of arc road segment.

As mentioned already, a single road is collection of road segment and a road network is collection. Figure 2 shows examples of a single road and road network. Left figure shows randomly generated single road using our PCG code (figure.2.a). Right figure shows randomly generated road network which has 3 different roads on same map (figure.2.b).

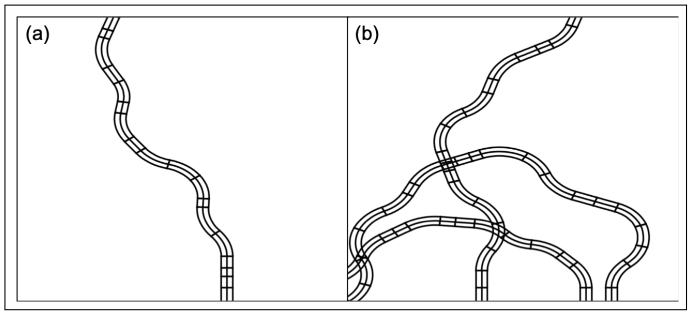

Figure. 2. Road and road network examples.
(a) Example of a random single road. (b) Example of random road network that consists of 3 different roads.

Road Segmentation Generation

In our project, we have two road segment types, straight road segment and arc road segment. Each road segment type is generated grammatically in a different way. At first, straight road segment is created using CreateStraightRoadSegment function which has three inputs, back line, length, orient. Back line is start line of new road segment defined as numpy array. Length means length of straight road segment. Orient is direction to grow road segment. Generation process is shown on left side of figure 3. First step is to construct start line by applying back line (figure.3.a). Next is to get orthogonal vector to back line. Direction of vector is determined by orient. Then, construct end line of segment by adding orthogonal vector to start line (figure.3.b) Third step is to calculate internal points of left edge, center line, and right edge. Each internal point is far away at the same interval (figure.3.c). Finally, connect internal points to form left edge, center line, and right edge (figure.3.d).

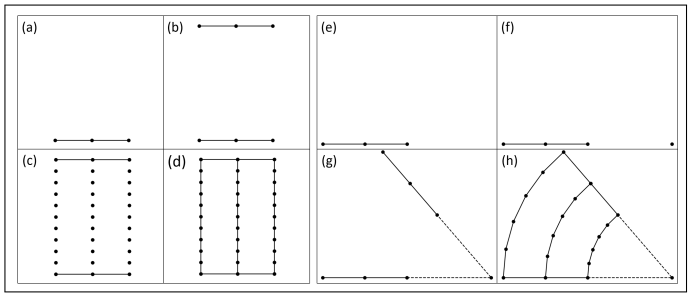

Figure. 3. Road segment generation.
Straight road segment (a) Construct back line (start line of segment) using input back line. (b) Construct front line (end line of segment) which is input length away from back line. (c) Calculate the internal points by dividing the input length by a specific interval. (d) Connect each internal point to form left edge, center line, right edge.
Arc road segment (e) Construct back line (start line of segment) using input back line. (f) Calculate the center of rotation, (g) Construct front line (end line of segment) by applying rotational matrix to back line. (h) Construct left edge, center line, and right edge whose internal points has same intervals.

Arc road segment is created using CreateArcRoadSegment function which also has three inputs, back line, radius, and angle. Back line is same as input of straight road segment. Radius is radius of curvature and angle is rotation angle. The positive angle value causes the road segment to rotate counterclockwise and negative value cause to rotate clockwise. Generation process is shown on right side of figure 3. First step is to construct the start line of road segment by using back line (figure.3.e). Next step is to obtain rotational matrix to rotate back line. First of all, get center of rotation by adding radius to center of back line. Then, obtain a rotation matrix that rotates back line by an angle (figure.3.f). Then, construct front line by multiply rotational matrix to back line (figure.3.g). In the end, construct left edge, center line, and right edge through new rotational matrix whose rotation angle is divided by specific interval (figure.3.h).

Single Road Generation

A single road is collection of multiple road segments. So, overall single road generation process is repetition of road segment creation with some validation checks. This functionality is run by CreateRandomRoad whose process is shown on figure 4. a, b, c. First step is to generate a short straight road segment on either the x-axis or y-axis. Position to be generated is randomly chosen (figure.4.a). Afterward, grow the road by creating random road segment Repeatedly (figure.4.b). Back line of newly created road segment will be front line of previous segment to generate gapless road. In this process, we check validity of growing road. Detailed validation method is explained in Section B. When the growing road reached one boundary of the map, single road generation process is done (figure.4.c).

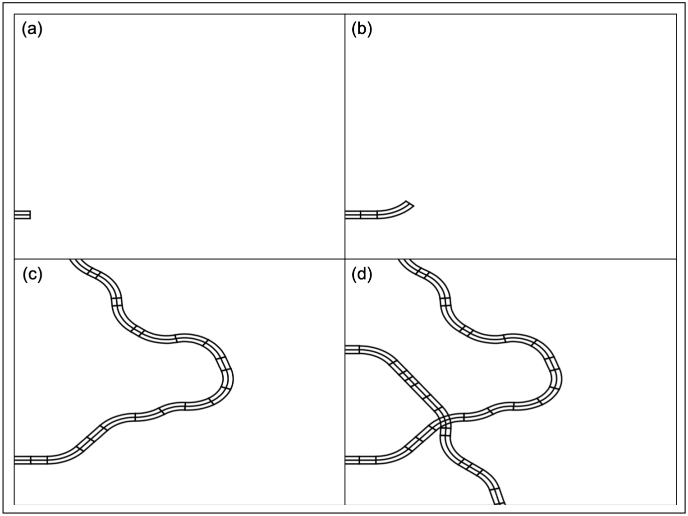

Figure. 4. Road and Road network generation.
(a) Construct a short straight road segment on either the x-axis or the y-axis. (b) Grow the generating road by creating random road segments. (c) Road generation process is completed when the road reached one boundary of the map. (d) Example of generated road network.

Road network Generation

A road network is collection of multiple roads. Therefore, to generate random road network, create some random roads and just add them. Similar to single road generation process, road network process has validation check step. The validation method was described in Section B. Example of random road network is shown on figure.4.d.

A.2. Map Validation Method

The procedural content generation should produce only roads and road networks on which autonomous vehicles can drive. Therefore, validation process is required after generating road and road network. Validation is performed on single road and road network respectively.

Single Road Validation

In order for a single road to be valid, two conditions must be satisfied. First, the two endpoints of the road must cross the map boundary. This condition is always satisfied because the segment is made repeatedly until it reaches the boundary of map. Second, self-intersection should not exist. Self-intersection is a point where two segments constituting a road cross each other. Figure 5 shows an example of road that includes self-intersection. In single road validation, it finds and removes self-intersection in a road.

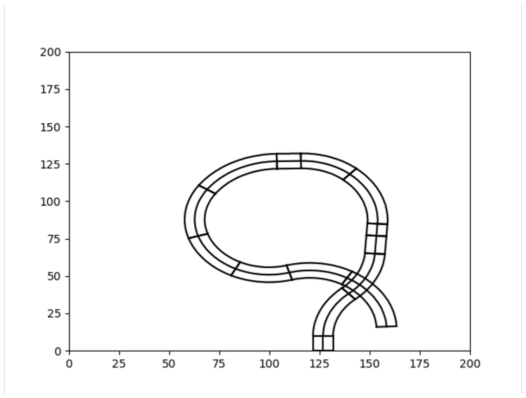

Figure. 5. Self-intersection.
Figure shows an example of self-intersection.

To understand how to check self-intersection, it is necessary to know how to find the cross-point of line segments. The cross-point of two line segments can be found by using the coordinates of both endpoints of the two line segments. Figure 6 shows two line segments with its two endpoints $f, p + r$ and $q, q + s$, respectively. p, q, r, s, t, u have the following relationship, equation 1 and equation 2.

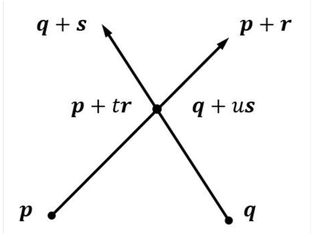

Figure. 6. Cross point of two vector.
If there are t and u between 0 to 1, the two vectors have cross point. The cross point can be represented as p + tr or q + us.

$$(p + tr) \times s = (q + us) \times s \tag{1}$$

$$(p + tr) \times r = (q + us) \times r \tag{2}$$

Using equation 1 and equation 2, t and u can be found.

$$t = \frac{(q-p)\times s}{(r\times s)} \quad (3)$$

$$u = \frac{(p-q)\times r}{(s\times r)} \quad (4)$$

If t and u exist between 0 and 1, the two line segments have an cross-point. The coordinate of the cross-point can be expressed as $p + tr$ or $q + us$. A road can be checked whether it has self-intersection by using the method of finding the cross-point of two line segments. Each road segment should not have intersections with other road segments in the same road. To check whether segments s and t have an intersection, the cross-points of the left and right edges of s and the left and right edges of t must be checked. Figure 7 shows 4 types of cross-points.

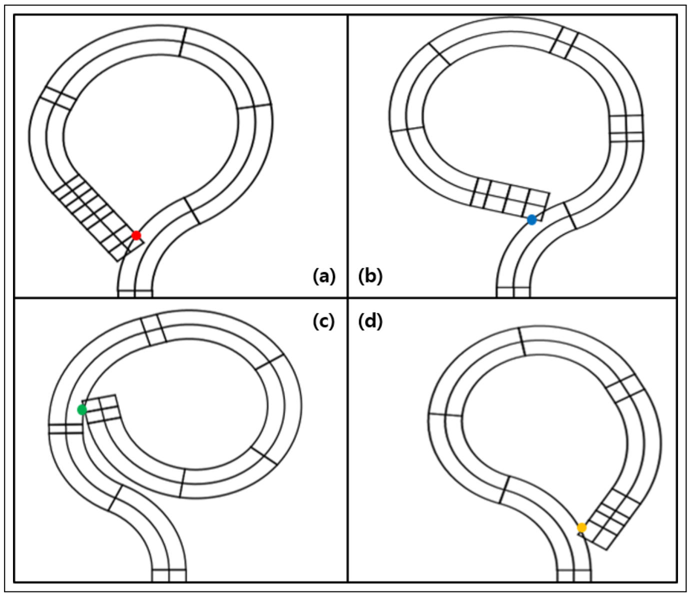

Figure. 7. Types of cross-point.
(a) Left-left cross-point. (b) Left-right cross-point. (c) Right-left cross-point. (d) Right-right cross-point.

There are left-left, left-right, right-left, and right-right pairs. For a total of 4 pairs of lines, the existence of cross-point must be checked. If any of the 4 pairs have cross-point, it is considered as an invalid road.

Road Network Validation

There are two conditions for validating the road network. First, all roads must have only valid intersections. Second, all roads must have at least one intersection. Invalid intersection that can occur in a road network is partial overlap. Partial overlap means that the roads do not completely intersect, but partially overlap.

As shown in figure 8, a valid intersection makes three cross-points on the left, center, and right lines, each. Therefore, in a valid road network, three lines of all roads have the same number of intersections, and the number is always multiple of 3. Figure 9 shows road networks with partial overlap.

In road a in figure.9.a, only the right line has two cross-points and the other two roads have no cross-point. Therefore, this road network has partial overlap and is invalid. In the case of road c in figure.9.b, the left line has 2 cross-points, the center line has 4 cross-points, and the right line has 6 cross-points. Since the number of intersections of each line is different, this road network is also an invalid road network.

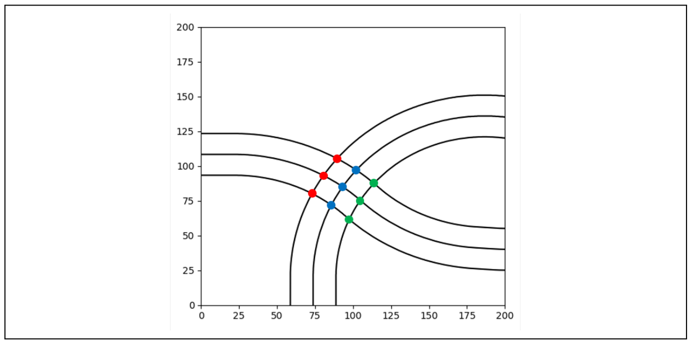

Figure. 8. Example of valid intersection.
Figure shows an example of valid intersection.

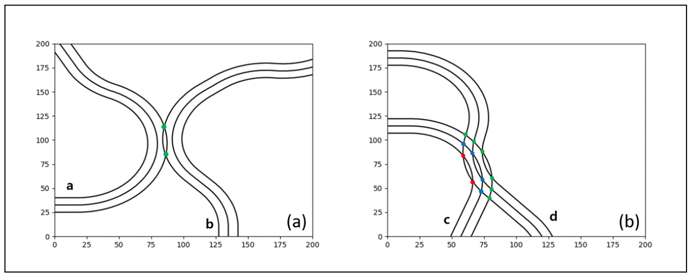

Figure. 9. Example of partial overlap.
(a) Road a has 2 cross-points on right-line, but others has no cross-point. (b) Road c has 2, 4 and 6 cross-points on left, center and right line each. As the numbers of cross points on each lines are different, they are invalid road networks.

After checking existence of partial overlap, it should be checked whether all roads have at least one intersection. If a road does not intersect with another road, the vehicle cannot move on that road from other road. Therefore, a road without an intersection becomes meaningless in a road network. If a road does not have partial overlap and the number of intersections of each line of the road is all 0, then this road does not have an intersection, and a road network with such a road becomes an invalid road network. Figure 10 is an example of invalid road network and valid road network.

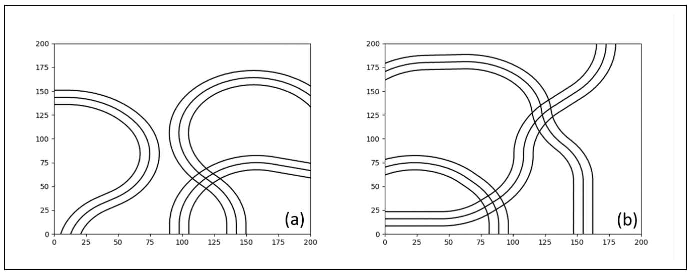

Figure. 10. Example of Road Networks.
(a) Invalid road network. (b) Valid road network.

B. GA

Generated road networks are used as genes for genetic algorithm. However, because they were difficult to handle freely, road networks consisting of poly lines changed its data structures. And define crossover and mutation methods for evolution. Furthermore, we have attempted to improve our algorithm by going beyond publishing existing papers.

B.1. GA Flow

Within our work, we applied genetic algorithm to generate test suites. We iterate 40 times with 25 populations. To evolve the generation, we used tournament selection among 5 populations to select parent for crossover operation. As a crossover operator, we used join crossover for the single road generation and used both of it with half probability for multi road generation. After the crossover, we mutate the child with mutation operator and insert to the child population. We generate child population until its size became equal to the entire population size. Among them, only best 90% of child population would survive within the next generation. For the remaining 10%, we used elitism so that 10% of best parent would survive within the next generation. We tested our work among small map and large map which are defined as (200, 200) map and (400, 400) map. Each of map is tested with both single road and multi road generation for 3 times.

B.2. Fitness Calculation

The evolving purpose of GA is to find the road population which is difficult for ego-car to go, and it includes the road in which has many OBEs occur. However, the number of OBEs is not suitable for use as a fitness value because there is a high probability that there are multiple roads with the same value. Therefore, we used bounded lateral deviation for the fitness value, which means the distance between the center of the road and the ego-vehicle. Figure 11 shows which values are used as fitness values.

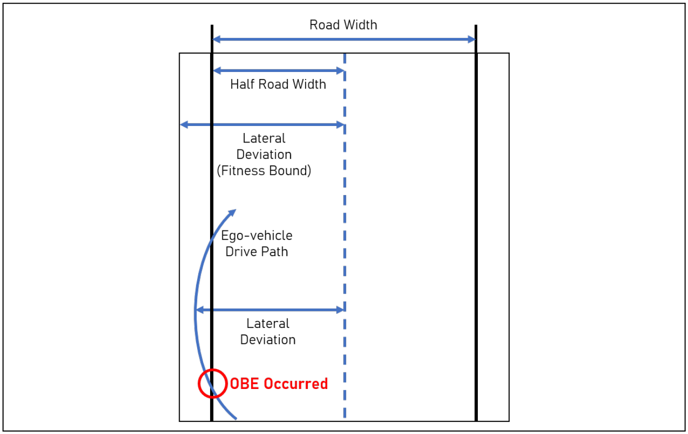

Figure. 11. Fitness value calculation.
The lateral deviation is used to calculate the fitness value, and OBE occurs when the lateral deviation becomes larger than the half road width as shown in the figure above.

As shown in figure 11, the longest value among the distances between the center of the road and the car is used as the fitness value, and when this value is greater than half the width of the road, OBE has occurred. However, if the simulation fails, unusually high values can occur and these values can hinder the evolution of efficient GA. Therefore, we used the value of the lateral deviation bounded from a certain value. In order to get better evolving efficiency on GA, we bounded the lateral value at a larger value than half road width. As mentioned above, calculated fitness values using the above method are sent to the GA program and used for population evolution.

B.3. Search Operator

Road networks are new data structures that we created for test case generation of simulation of self-driving cars. So, to create the next generation of population using this structure, we need a new crossover and mutation method that fits this structure. So, we evolved road networks through various methods, such as mutate them in certain areas or cutting and connecting two different road networks.

Join Crossover

Join crossover is a method for creating a new road network by combining two road networks appropriately. Join crossover cut the roads in two road networks around a randomly assigned segment. Then connect the parts of the road that cut from two road to create a new road. If this road is not reached the map boundary, we created random road segment from the end of the road until it is intersected with map boundaries. Operate this method for every randomly selected roads from each road network (figure 12).

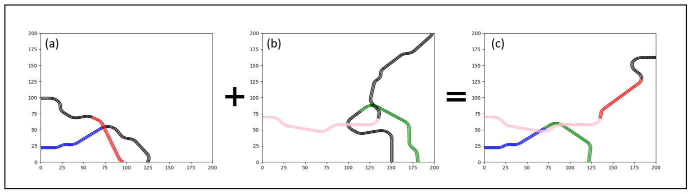

Figure. 12. Join Crossover operator.
(a), (b) Example of road networks. (c) Join crossover result of (a) and (b). Blue part of (a) and green part of (b) are connected, and red part of (a) and pink part of (b) are connected in (c).

Merge Crossover

Merge crossover is a method of creating a new road network by randomly selecting two or three roads from two road networks without any variation on the road itself in the road networks (figure 13).

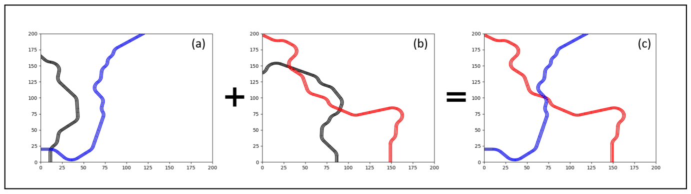

Figure. 13. Merge Crossover operator.
(a),(b) Example of road networks. (c) Merge crossover result of (a) and (b).

Mutation

Mutation is a method of applying variations to the generated road networks to give them diversity. Randomly set one of all road segments of the road network and eliminate all segments that lead to the beginning of that selected segment. Then created a new road segment based on the cut section (figure 14).

B.4. Similarity Test

If there are too many similar road networks in the population, the probability of local optimization increases for the road network that is evolved through genetic algorithm. Therefore, we

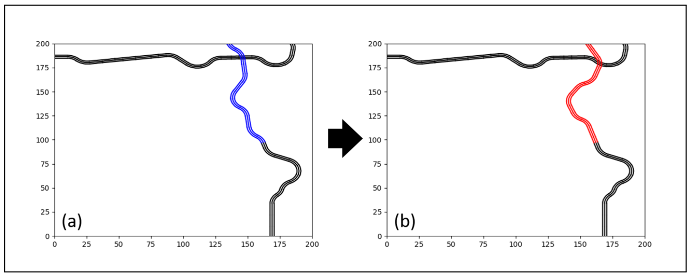

Figure. 14. Mutation operator.
(a) Example of road networks. (b) Result of mutation of (a). Blue part of (a) is changed to red part.

defined the differences between road networks and wanted to reduce similar road networks within a generation by comparing similarity with all offspring of current population.

It is difficult to compare similarity with the data structure of the road network. Therefore, we compare similarity by expressing all road segments of the road network in the form of a histogram. Histogram was classified based on the radius of curvature of the road segment. And the sign(+ or -) of the radius of curvature was determined by the direction of curvature. In the case of a straight line segment, the radius of curvature was set to zero to distinguish it from the arc road segment because the radius of curvature does not exist in this case. And each segment's weight was set to the segment's length. The following road network is expressed in histogram as follows (figure 15).

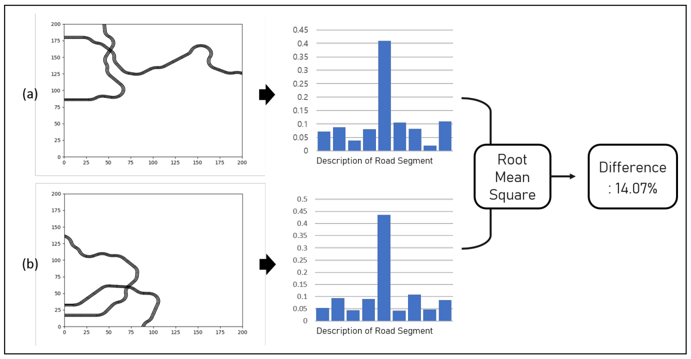

Figure. 15. Calculate similarity of two road networks.
(a), (b) Example of how to describe each road networks to histograms.

Then, we defined the difference as a RMS(Root Mean Square) of two histogram values of each road network(equation 5). h_1 and h_2 is a histogram of two road networks.

$$Difference = RMS(h_1, h_2) \quad \textbf{(5)}$$

The closer the difference is to zero, the more similar the two road networks are, and the larger the value, the more different the two road networks are. When a new generation of road networks was created, the similarity with all road networks of existing population was measured and added to population only if it exceeded the threshold. The size of the threshold was empirically chosen.

B.5. Improvement of Our Project

Beyond replicating the paper, we tried to develop the poor parts and limitations that exist in the paper and apply them to our system. We applied the improved method in the search operator and similarity check part.

The original crossover operator is done with the entity of map. Even though the fitness is evaluated with its actual path, the original crossover operators do not count on this fact within its operation. Within this context, we proposed logically improved crossover operator that is done with the entity of path. Also, we designed the crossover operator based on 2 main features. We assumed that the traits of the map, which is fitness, come from 2 things which are the geometrical shape of road itself and their intersection. These perspectives are also reflected to each of the crossover to prove that we actually enhanced their logical meaning.

Improved Join Crossover

The improved join crossover is done with the randomly selected join point from its path. Figure 16 shows the procedure of the join crossover. The red line within figure.16.a and figure.16.b shows the selected path of the map. Using this information, the join crossover operator selects the join point within road that including the path. These selected roads are displayed as blue line (figure.16.a) and green line (figure.16.b) so that we can verify that child road (figure.16.c) is consisted of only blue line and green line. We suggest that this operator logically enhance the meaning of the original join crossover operator. Considering the purpose of the crossover, its main purpose lies on that to combine the dominant trait. In our work, the dominant trait definitely lies on its actual path in the map. Therefore, if we randomly select the road from the the parent map selected for the crossover, we can not say that we actually crossover the dominant trait and preserve it to next generation. Within this context, by selecting the join point from its actual road, we can ensure that its dominant trait would be preserved to the next generation. To be more specific, we can ensure the preservation of the dominant traits that lies on the road's geometrical shape itself. (which is our first perspective as we described above) With this crossover, we logically improved the meaning of join crossover to accelerate the convergence of fitness by combing preserved road's geometrical shape.

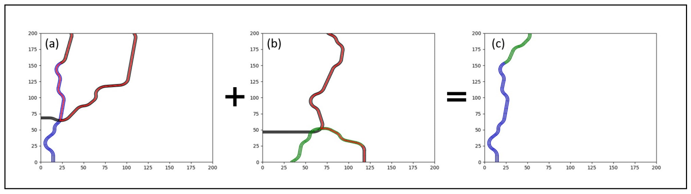

Figure. 16. Improved join crossover operator.
(a), (b) Example of road networks. (c) Improved join crossover result of (a) and (b).

Improved Merge Crossover

The improved merge crossover is done with the randomly selected road that including path. Figure 17 shows the procedure of the merge crossover. The red line within figure.17.a and figure.17.b shows the selected path of the map. Using this path, the merge crossover operator selects the road to be crossovered that including the path. These selected roads are displayed as blue line (figure.17.a) and green line (figure.17.b) so that we can verify that child road (figure.17.c) is consisted of only blue line and green line. Similar to join crossover, it also logically improves merge crossover. The only different thing is that the preserved dominant traits would be enlarged to the intersection of the road not just by road itself. Compared to join crossover, the merge crossover would preserve the geometrical shape of the road. Therefore, it would try to enhance the fitness by creating new intersection with the set of "proved to be generate intersection with great fitness" roads. (which is our second perspective as we described above) With this crossover, we logically improved the meaning of merge crossover to accelerate the convergence of fitness by generating intersection with proved set of roads.

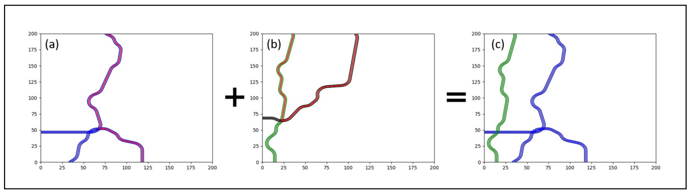

Figure. 17. Improved merge crossover operator.
(a), (b) Example of road networks. (c) Improved merge crossover result of (a) and (b).

Improved Similarity Crossover

Existing method for similarity test has two problem. First, straight line road segment affects to difference between two road networks a lot. Because when we made a road network, we randomly select whether it is an arc or a straight line. And the odds are half each. Therefore, the proportion of the straight line segment in the histogram accounts for half of the total (figure 18, center of a histogram is a straight line segment). Therefore, this scale should be matched similarly to other segments. Second, original method didn't describe the variance of road segments. So then, it cannot noticed that there is a big difference between these road networks (figure 19).

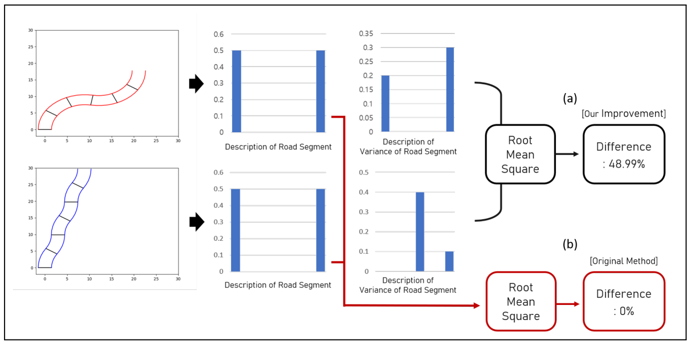

Figure. 18. Difference between original similarity test and improved similarity test.
(a) Difference of improved similarity test (b) Difference of original similarity test.

Therefore, We reduced the weight of the straight line segment to one-quarter so that the total is similar to the other segments around it. And we also describe the variance of road segment as a histogram. In addition to the road segments that we describe in the existing similarity test, we explained the changes of radius of curvature in the current road segment and the next road segment. Weight of them is a average value of length of current and next road segments. As a result, we can recognize this difference. Then, improved similarity test method is operated through this process (figure 19).

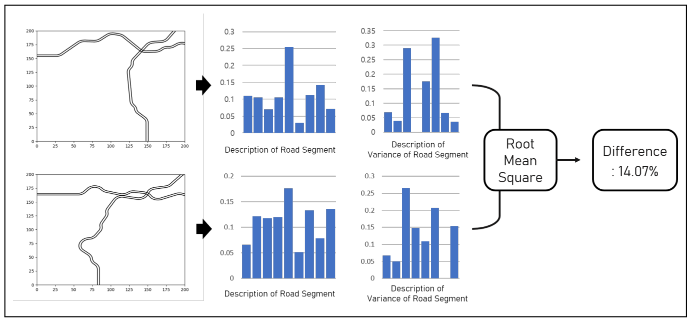
Description of Road Segment
Description of Variance of Road Segment
Root Mean Square
Difference : 14.07%

Figure. 19. Similarity Calculation.
Example calculation of two road networks with improved similarity method.

3. EVALUATION

To evaluate our model, we measured the fitness value, cumulative OBEs, vision errors, and map generation time by running the ego-car on the map we generated through the waypoint using Matlab Simulink simulator. The details are written below.

A. Evaluation Method

To evaluate the generated map with simulation, we need to generate the path to be evaluated. This path called waypoint would be transferred to the Matlab simulator so that fitness is evaluated with simulation. Within this context, to generate waypoint we convert map to the graph and find out the path.

A.1. Waypoint Generation

Below box is the procedure to generate the waypoint.

1. Converting Road to Graph
2. Finding Path
3. Waypoint Generation

Converting Road to Graph

To generate waypoint, we first convert the map to the graph. Each End point of road and the intersection of the node would be node with the priority in the below box.

Converting Road to Graph (Priority)
1. Each end of the road would be End Node.
2. There would be only one intersection within one segment.
3. Find intersection starting from the road 0 so that earliest road would be prioritized.

Figure 20 shows the graph that applied the priority 1 above priority 3. Even though there exists intersection between green road and blue road around End Node 3, it would not set its intersection as Node since there is already End Node 3 within same segment. Figure 21 shows the graph that applied priority 2 above priority 3. Even though there exists intersection between green road and blue road around Node 4, it would not set its intersection as Node since there is already Node 3 and Node 4 within the same segment. Figure 22 shows the graph that no duplicated priority is applied.

We devised this priorities to minimize the overhead of finding out the actual graph. Its implementation is done with 5 for loops to test over all the possible intersection test. To minimize cost, we skip the meaningless computation with the priority that we

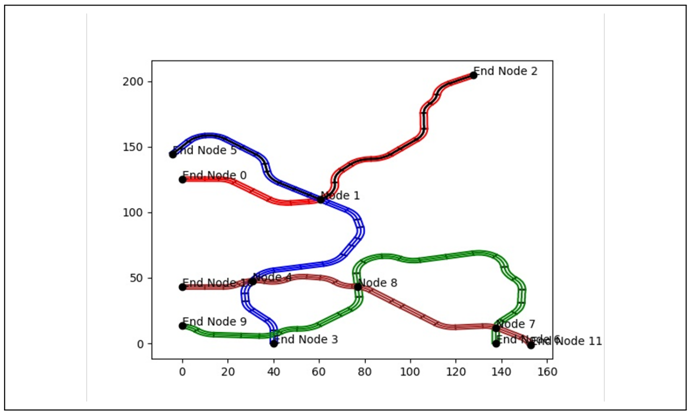

Figure. 20. Example graph 1.
Example graph that priority 1 has been applied as the highest priority.

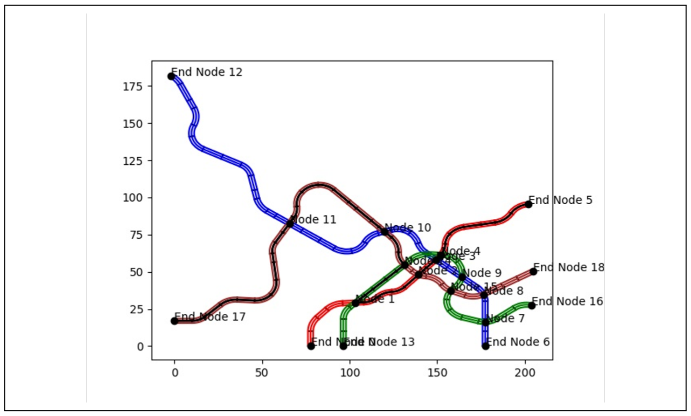

Figure. 21. Example graph 2.
Example graph that priority 2 has been applied as the highest priority.

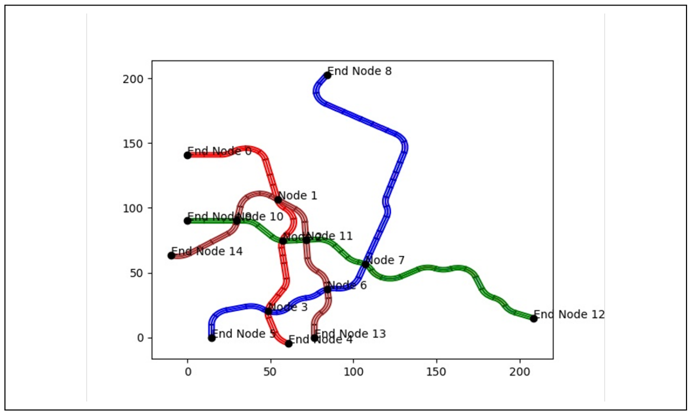

Figure. 22. Example graph 3.
Example graph that priority 3 has been applied as the highest priority.

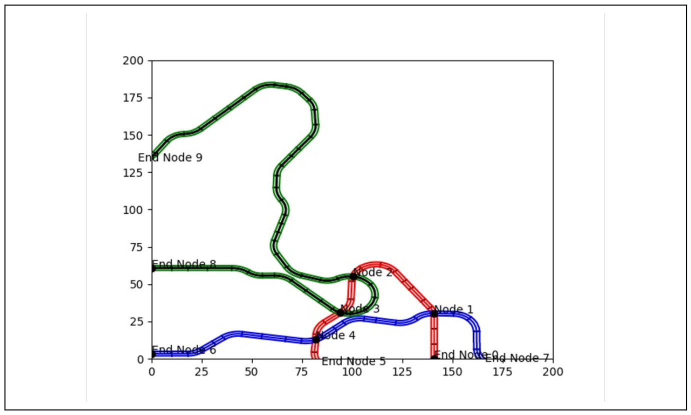

Figure. 23. Example graph.
Example graph corresponding to edge table in table 1

devised. All the nodes that figured out would be recorded with each roads. After finding out the node, we traverse over the roads to find out the edge. Using the fact that edge directly connect each nodes, we connect each nodes within the road to find out the edge while computing its length. The length here is defined as the number of segment between each nodes. Figure 23 shows the example of the graph and table 1 shows the corresponding edge table of the map.

Finding Path

Using the Graph, we find out the path that is the sequence of the nodes. We find the path with the conditions in the below box.

Finding Path (Condition)
1. Find the path that both start node and end node is End Node.
2. The path should not pass the same Node more than once.
3. For efficiency, find the longest path among 3 randomly found path (Long in terms of number of segment).

Starting from the randomly chosen start node, we try to find out the path that end with the end node. We find out path by selecting edge from that node with equal probability. Also, since finding out all the path is quite complex and meaningless in terms of efficiency, we find out 3 paths and choose the best path among them. Figure 24 shows the example of the path with corresponding map. We can verify that path figure.24.a is shown as black line within the map figure.24.b.

Waypoint Generation

Using the path, we generate actually waypoint that simulator can follow. Since the path we generate is the sequence of Node, we connect each node to generate waypoint with the condition in the below box.

Waypoint: The path that simulator would follow
1. The center line of the segment between each node would be path.
2. The segment that including Node would be divided to 2 parts and connected.

We can generate waypoint using the edge between node. Here, since the edge is consists of sequence of segment as we described, we sequentially connect segment center line between nodes. For the segment that including the node, we divide each segment as 2 part so that it connected as part 1 of segment 1 -> node -> part 2 of segment 2. The Figure 25 shows the example of generated waypoint. The roads within the map is shown as black line while its path waypoint is shown as red line.

Table 1. Edge table of figure 23.

Node1	0	1	1	1	1	2	2	2	3	3	3
Node2	1	0	2	4	7	1	3	9	2	4	8
Distance	3	3	7	8	4	7	5	30	5	2	11
Node1	4	4	4	4	6	7	8	9			
Node2	3	5	6	1	4	1	3	2			
Distance	2	1	9	8	9	4	11	30			

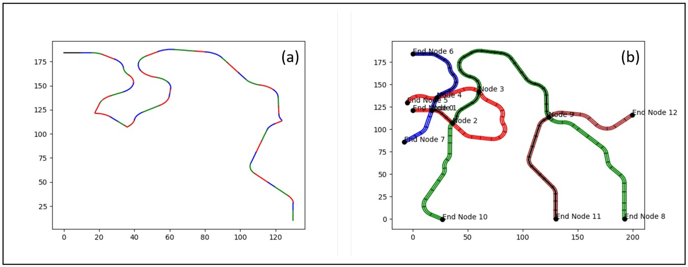

Figure. 24. Finding path.
(a) Path example. (b) Path within the map.

A.2. Communication Method

As mentioned earlier, in order to proceed with the simulation, the information of the population to be evaluated including the road network and way point from GA must be transferred to Simulink, and additional communication is required for the synchronization of the two programs. First, a .csv file format was used to transfer the information of the population generated by GA from the GA program to Simulink. GA program creates a csv file in the designated path, write information into the file, and sends a signal to Simulink and the name of the file. Then, Simulink reads the file with received filename to get the information. Depending on the size of the road network or the number of roads included, the number of coordinates to be transmitted is not constant and can be very large, so we thought it is safer to share information via a file rather than using a specific communication protocol. In addition, by using this method, there is an advantage that we can leave all the simulation data because it is saved on the path.

For the synchronization of the two programs, TCP/IP communication was used. TCP/IP communication is a packet based communication protocol. For this communication, both programs open a socket at run time and establish a communication environment. The process of sending the file name and signal from the aforementioned GA program to Simulink is performed through TCP/IP communication, and after the simulation is completed, the process of sending whether the simulation was successful along with the fitness value calculated from the simulation result from Simulink to the GA program is also done through TCP/IP communication. After sending the simulation related signal to Simulink, the GA program waits in a stopped state, and after receiving the fitness value from Simulink, it continues to evolve again using the value. A description of total process is shown in figure 26.

A.3. Simulink

In order to evaluate the road network, the process of making the ego-car to move according to the previously created simulation information (road network, navigation path) using the control of the autonomous driving system and checking the number of OBEs must be conducted. This requires a simulation program that can calculate the position of the ego-car and simulate the driving of the ego-car over time. In this paper, we used Simulink as a simulation program. Simulink is a graphical simulation program based on MATLAB, a programming language that provides a numerical computing environment. Simulink is widely used in the researching area because it is easily accessible and provides a variety of examples are provided officially. The Simulink simulator used in this paper is designed to receive the coordinates of the center line of each road existing in the road network and navigation path (way point) information and conduct a simulation based on the information. This is the road network information that is needed to be evaluated in the GA process.

First, in order to conduct simulation efficiently in Simulink, we translated the road network into good form to simulate. Good form of road map for Simulink is as follow. First, the start point of simulation path should be origin point. Then, the simulation path should start in the positive direction of the x-axis. Therefore, this conversion process is performed by applying rotational matrix and translation matrix and it is shown on figure 27. At first, we obtain translation vector which move start point of simula-

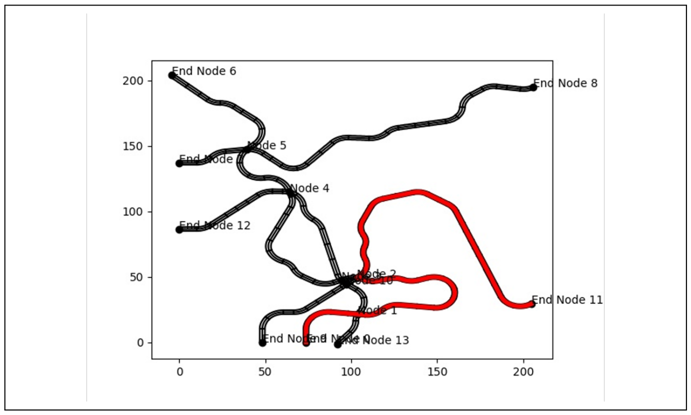

Figure. 25. Waypoint generation example.
Example of generated waypoint of graph.

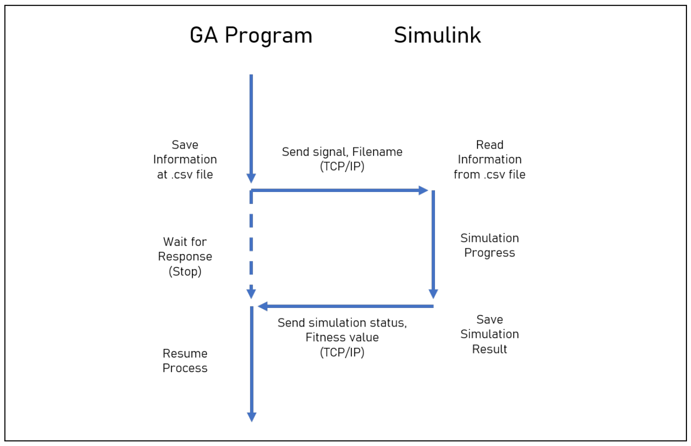
GA Program
Simulink
Save Information at .csv file
Send signal, Filename (TCP/IP)
Read Information from .csv file
Wait for Response (Stop)
Simulation Progress
Send simulation status, Fitness value (TCP/IP)
Save Simulation Result
Resume Process

Figure. 26. Communication Protocol.
The entire communication protocol between the GA program and Simulink proceeds in the order shown in the figure above.

tion path to origin point (figure.27.a, b). By applying the vector to all points in road network, we can obtain center started road network (figure.27.c). Then, obtaining rotational matrix which transform the simulation path to start in the positive x-axis direction and applying it to all internal points in the road network, we can convert Simulink format road network. (figure.27.d)

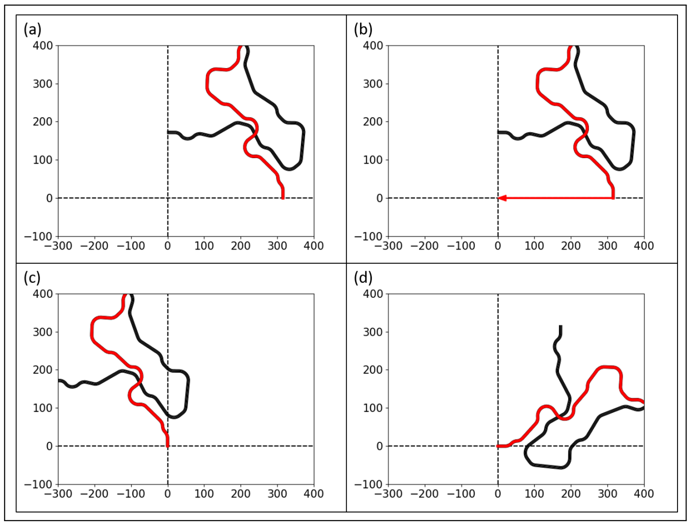

Figure. 27. Road network conversion process.
(a) Original road network. Red line is simulation path. (b) Translation vector to move start point of path to origin point. (c) Converted road network applied translation. (d) Simulink format road network.

Next, Simulink creates a virtual road map using the converted center line information, puts the ego-car at the start point, and allows it to drive along the way point. Figure 28 shows the behavioral mechanism of the designed Simulink simulator, and figure 29 shows an example of the total simulation process. For the Simulink design shown in figure 28, the Lane Keeping Assist with Lane Detection Example provided on the MATLAB homepage was referenced: https://kr.mathworks.com/help/mpc/ug/lane-keeping-assist-with-lane-detection.html.

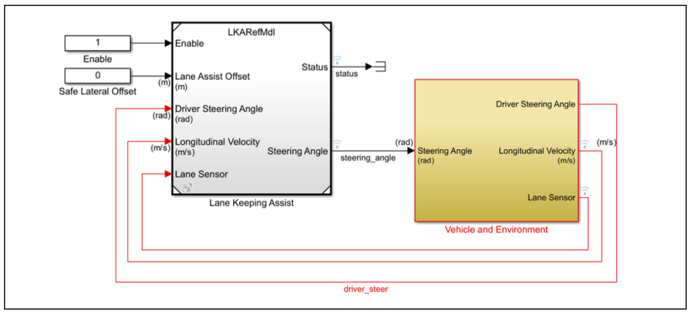

Figure. 28. Simulink simulator mechanism design.
The simulation proceeds according to the mechanism designed in the figure.

As you can see in figure 29, depending on the option provided by the user in the driving process, it can drive based on perfect knowledge of the road, or drive using lane keeping functionality using vision information. The vision sensor of the simulator provides road data that the ego-car can recognize through the

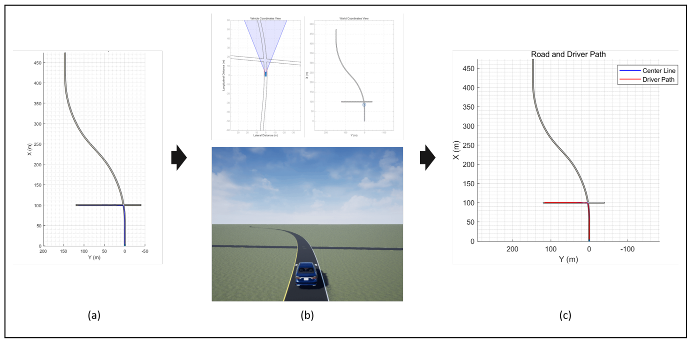

Figure. 29. Simulation process example.
(a) Visualized converted road map in Simulink. (b) Visualized simulation process in Simulink. Left is bird's eye scope and right is 3D-view. (c) Visualized result of simulation in Simulink.

camera within a given viewing angle based on the road information, and the simulator performs control based on this data. After the simulation is over, the fitness value is passed back to the GA program, which was waiting for the fitness value, and then the population in the GA process can continue to evolve through this value. After the simulation was completed, the information delivered to Simulink, the appearance of the created road network, and the path the ego-car moved during the simulation process were saved as files and pictures so that the simulation process can be reproduced later if necessary. For communication, the previously mentioned communication protocol was used, and the calculation of the fitness value was specified later.

B. System Evaluation

In our paper, we tested random in small map and K-DAsFault in both small and large map. We tested them with the GA flow above with additional similarity criteria. K-DAsFault filter similar test cases when their difference is less than 0.04. We tested each of them 3 times. With the result, we can analyze our K-DAsFault within 3 perspectives which are their validity with random, map size and number of roads as variables.

B.1. Fitness Evaluation

Figure 30 shows the fitness result over 40 generation. To validate our K-DAsFault, we compared fitness with the random in table 2.

Table 2. Compare result of K-DAsFault to random (Fitness value).

K-DAsFault	Compared to random
Small Single	1.34 times
Small Large	1.37 times
Multi Large	1.26 times

From table 2, we can verify that our K-DAsFault actually generate better fitness valued map compared to the randomly generated test suites. In other word, we can verify that our GA works great to generate tests suites better than random.

Comparing the Large Single road result with Large Multi road result, we can find out the relation between result and number of roads. In entire generation we observed, single road has

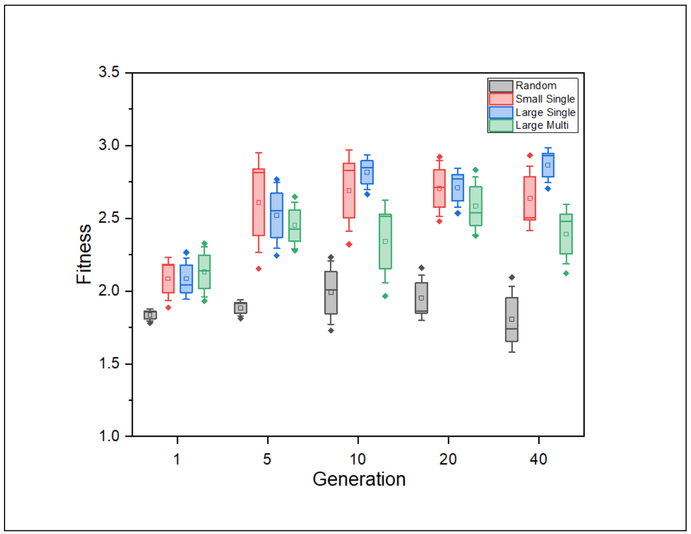

Figure. 30. Fitness value measurement result.
The graph shows the average fitness values of each road map on 1, 5, 10, 20, 40 generation.

better fitness than multi road. Comparing the Small Single road result with Large Single, we can find out the relation between result and map size. We can verify that large map has better fitness than small map on average.

B.2. Cumulative OBE Evaluation

Figure 31 shows the fitness result over 40 generation. Compared to fitness value which is actually for GA, cumulative OBEs are more practical value that we should evaluate. To validate our K-DAsFault, we compared the number of cumulative OBEs with the random in table 3.

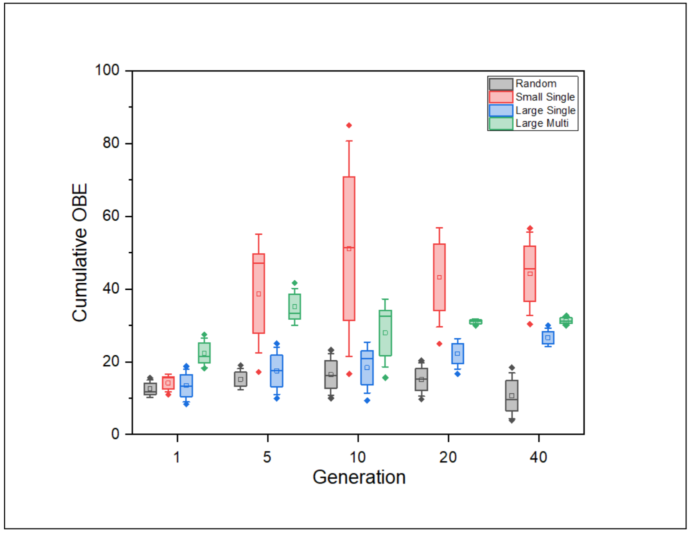

Figure. 31. Cumulative OBE measurement result.
The graph shows the average number of cumulative OBEs of each road map on 1, 5, 10, 20, 40 generation.

From table 3, we can verify that our K-DAsFault actually generate the maps that have more cumulative OBEs than random. In other word, we can verify that our K-DAsFault successfully expose more cumulative OBEs compared to random.

Table 3. Compare result of K-DAsFault to random (Cumulative OBE).

K-DAsFault	Compared to random
Small Single	2.75 times
Small Large	1.46 times
Multi Large	2.15 times

Comparing the Large Single road result with Large Multi road result, we can find out the relation between result and number of roads. In entire generation we observed, multi road has more cumulative OBEs than single road. Comparing the Small Single road result with Large Single, we can find out the relation between result and map size. We can verify that small map has more cumulative OBEs than large map on average.

B.3. Vision Error Evaluation

Figure 32 shows the the number of vision error occurred over 40 generation. The vision error is occurred mainly because the ego car totally fails to track the road. Vision error case is too out of bound to regard as the case that having maximum fitness. Therefore, we count this one as vision error which is not included in fitness value. But, since this case is also the case that ego car out of the road, we suggest that this error could be another metric to evaluate the result.

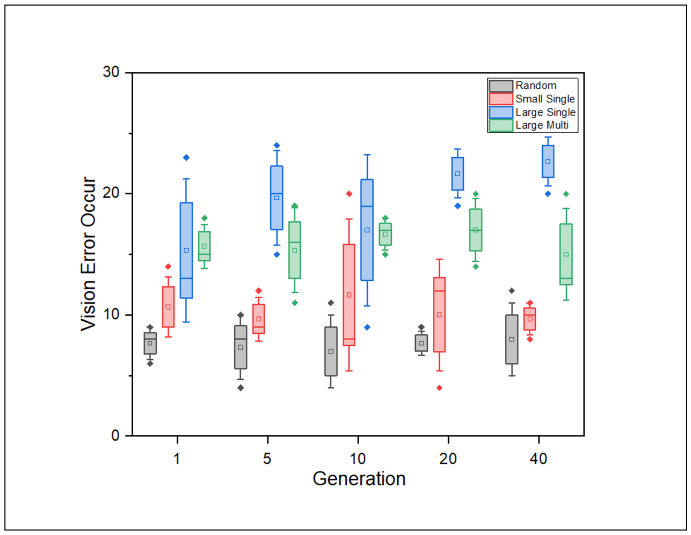

Figure. 32. Cumulative OBE measurement result.
The graph shows the average number of vision errors of each road map on 1, 5, 10, 20, 40 generation.

Table 4. Compare result of K-DAsFault to random (Vision Error).

K-DAsFault	Compared to random
Small Single	1.38 times
Small Large	2.55 times
Multi Large	2.12 times

From table 4, we can verify that our K-DAsFault actually generate the maps that have more vision error than random. This value as an another metric, we can verify that our K-DAsFault successfully expose more problems to ego car compared to random.

Comparing the Large Single road result with Large Multi road result, we can find out the relation between result and number of roads. In entire generation we observed, single road has more vision errors than multi road.Comparing the Small Single road result with Large Single, we can find out the relation between result and map size. In entire generation we observed, we can verify that large map has more vision errors than small map.

B.4. Time Evaluation

Figure 33 shows the time result over 40 generation. We measured time for generating the maps for each cases using K-DAsFault.

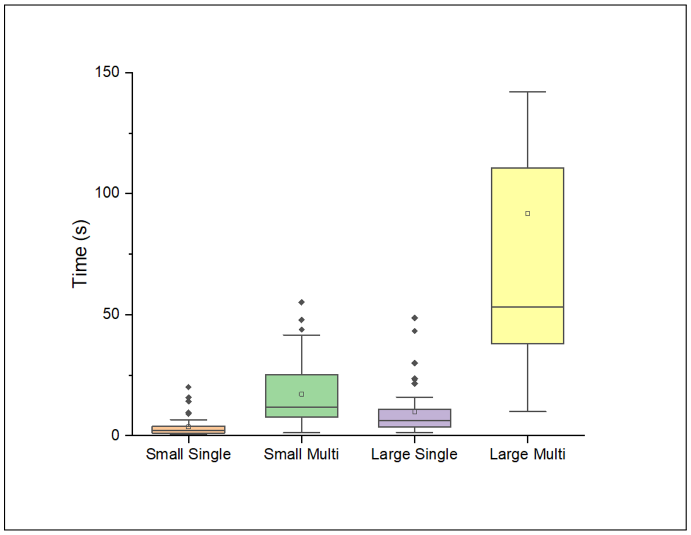

Figure. 33. Time measurement result.
The graph shows the average generation time of each road map on 1, 5, 10, 20, 40 generation.

From the result that comparing Small Single with Large Single and Small Multi with Large Multi, we can verify that large map takes longer time to generate than small map. From the result comparing Small Single with Small Multi and Large Single with Large Multi, we can verify that multi roads take more time to generate than single road.

4. DISCUSSION

To conclude, the overall tendencies of each result was almost uniform so that we can validate that K-DAsFault successfully expose problems to its map. However, there is some interesting point to be discussed. Even though fitness value and OBEs are somewhat related metrics, we can find out that their result is reversed for each cases. This tendency with map size is summarized in table 5.

In table 5, the entry shows that their result with inequality that have relation of 1>2. Within same context, table 6 summarize the tendency with number of roads.

As we described above, from table 5 and table 6, we can figure out that their result of fitness and OBEs are reversed even though their closely relative metrics. And, this is the reason why

Table 5. Map size test result.

Result	1	2
Fitness	Large	Small
OBEs	Small	Large
Vision Error	Large	Small

Table 6. Number of roads test result.

Result	1	2
Fitness	Single	Multi
OBEs	Multi	Single
Vision Error	Single	Multi

we introduced vision errors as a new metrics. Within the procedure of evaluating OBEs, actually it is quite ambiguous to definitely define its range. To be more specific, it would be quite hard to be fair for evaluating their numbers. Even though going out of the road for 10m is much more dangerous than going out for 5m, the number of OBEs are counted as if they are same one. Furthermore, for the case that the amount of car went out goes infinite, it would be much harder to evaluate. Therefore, for this case, we introduced new metrics that called vision error to evaluate with more fairness. Within this context, we can explain that the reason why the results are reversed. Even though large map has better fitness compared to small map, it has lower number of cumulative OBEs. However, large map has more vision errors compared to small map. Here, we suggest that we should compare their tendency with considering both of cumulative OBEs and vision errors not just OBEs. Some of having infinite error cases that are omitted for counting cumulative OBEs are counted as vision errors. Therefore, now we can explain the reversed tendency with these new metrics. We can do same thing for number of roads. From the time result, we can say that larger the map and more the number of roads, it takes more time to generate the map. This is obvious result that generating bigger map needs more segments of roads to be generated. Also, having multi roads takes more time to generate roads and increase the probability of having bad intersection. This one would be more critical so that we can verify that generating Small Multi map takes longer time that Large Single map.

5. CONCLUSION

In this project, we replicated proposed system for generating test case of lane keeping functionality, which is AsFault. The objective of our project is to generate test data with more OBEs and better fitness value compared to random generation. In this project, we improved the original AsFault and proposed new system, K-DAsFault, with our own PCG and GA to automatically generate and evolve road maps. K-DAsFault has improved road generator in terms of efficiency and logically improved search operator and similarity test method. For evaluation, we used graphical simulation program based on MATLAB Simulink, which has vision based lane keeping system. Using this simulator, ego-car drives along the virtual road observing driving path and lateral deviation during simulation. On evaluation, we tested random in small map and K-DAsFault in small and large map. We evaluated the simulation results using 4 metrics: fitness, cumulative OBE, vision error and time. K-DAsFault was 1.32 times better than random on fitness, 2.12 times on cumulative OBE and 2.02 times on vision error on average. Our evaluation showed that K-DAsFault can effectively generate test suits and successfully expose faults in lane keeping functionality.

6. REFERENCE

1. Alessio Gambi, Marc Mueller, and Gordon Fraser. 2019. Automatically testing self-driving cars with search-based procedural content generation. In Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA).