CS454 Final Presentation

"K-DAsFault"

KAISTian-Designed Automatically Testing Self-Driving Cars with Search-Based Procedural Content Generation

Team 21

20160633 Jaemin Cho 20170057 Geon Kim 20170150 Jaeuk Kim 20170168 Jiin Kim 20170719 Seunghee Han

1. Introduction

Problem Statement

1. Autonomous driving has become an important part of the industry
2. Finding possible accident of autonomous vehicle is important

Objective of this Project

: Generate test data for virtual tests of autonomous vehicle

- Effectively expose problems in self-driving car software
- Target the lane keeping functionality of self-driving cars
- Maximize Out of Bound Episodes (OBE)

2. Procedural Content Generation(PCG)

The system that generates the virtual road network automatically.

Three steps to create random road network

1. Road Segment Generation
2. Road Generation
3. Road Network Generation

Validation method

- Self-intersection
- Partial overlap

2.1 Road Segment Generation

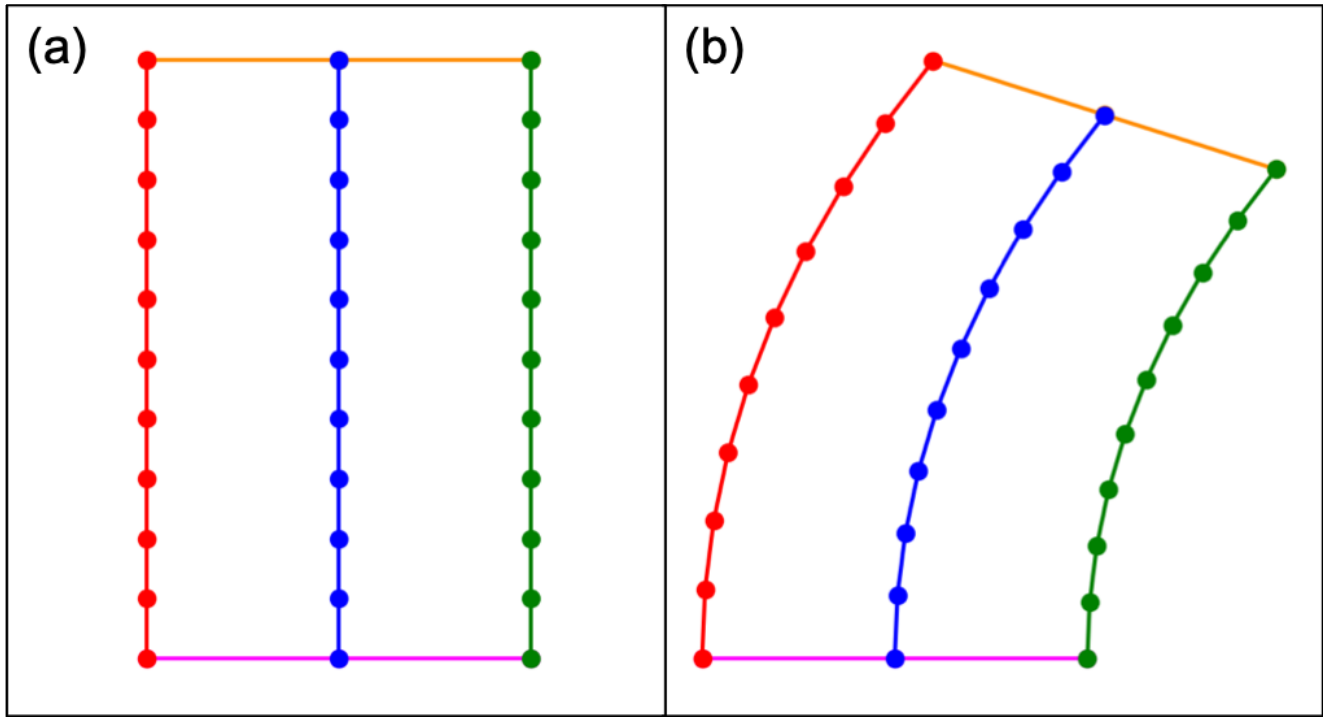
Road segment : the smallest unit of road network

Two types of segment

- Straight road segment (a)
- Arc road segment (b)

Represented by 5 lines

- Backline & Frontline
- Right edge & Left Edge
- Centerline



2.1 Road Segment Generation

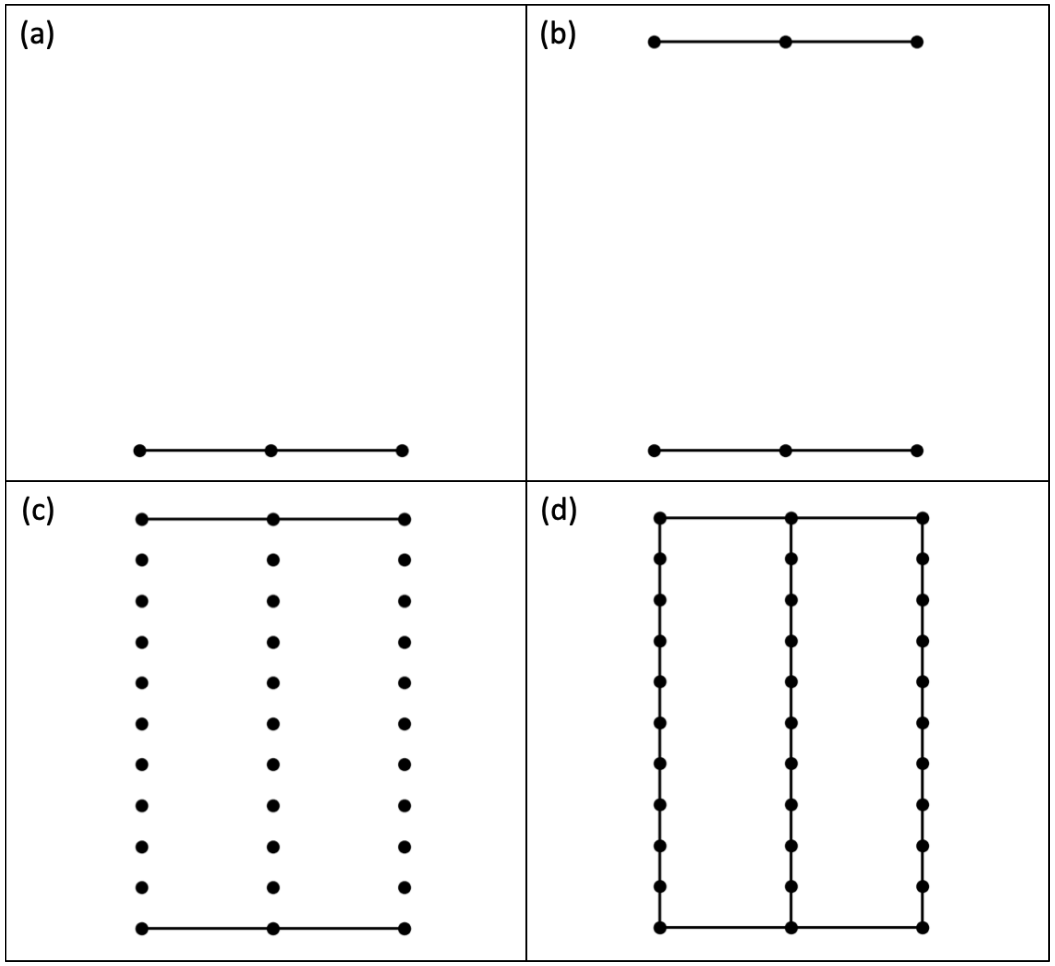
Straight road segment generation

Input :

- Backline of segment
- Length of segment
- Direction of segment

Process

1. Find orthogonal vector of backline
2. Apply the vector to backline



2.1 Road Segment Generation

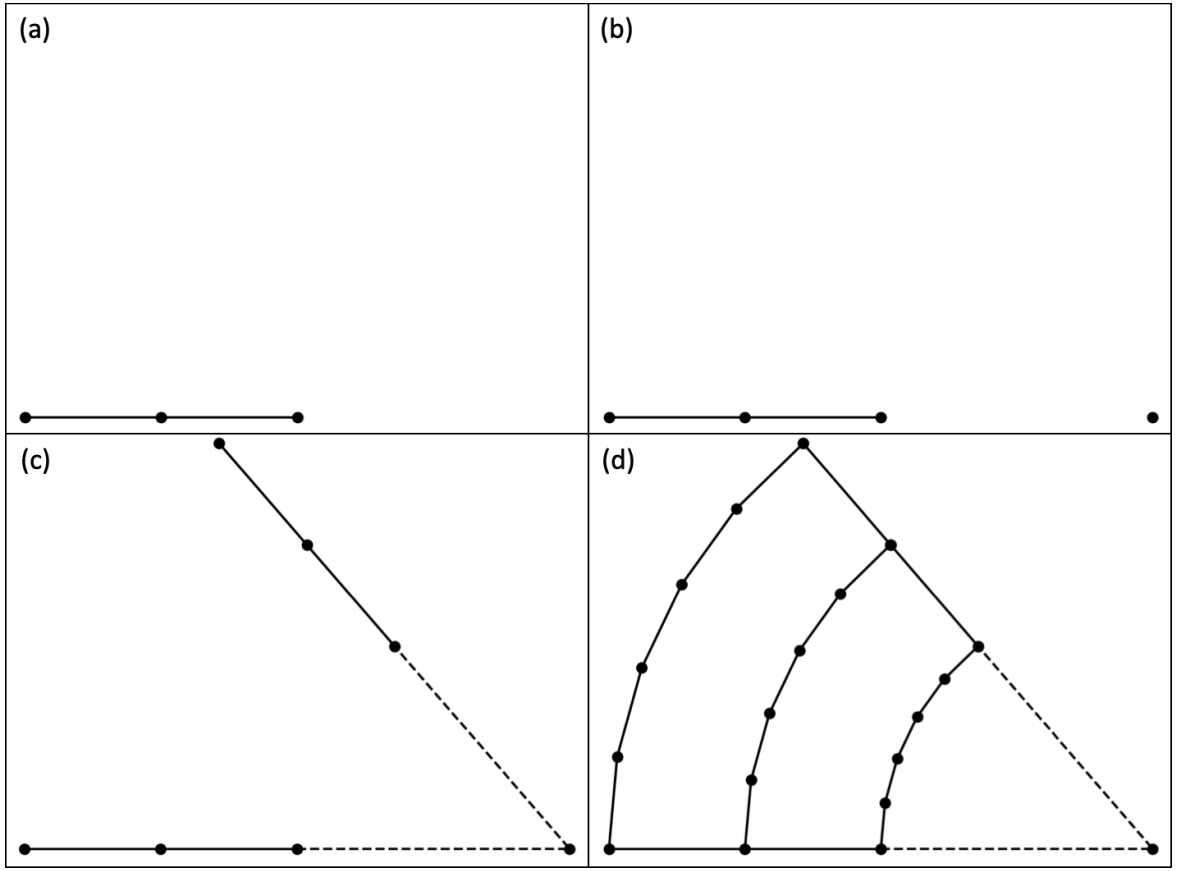
Arc road segment generation

Input :

- Backline of segment
- Radius of curvature
- Rotation angle

Process

1. Find rotation axis
2. Find rotation matrix
3. Apply rotation matrix to backline



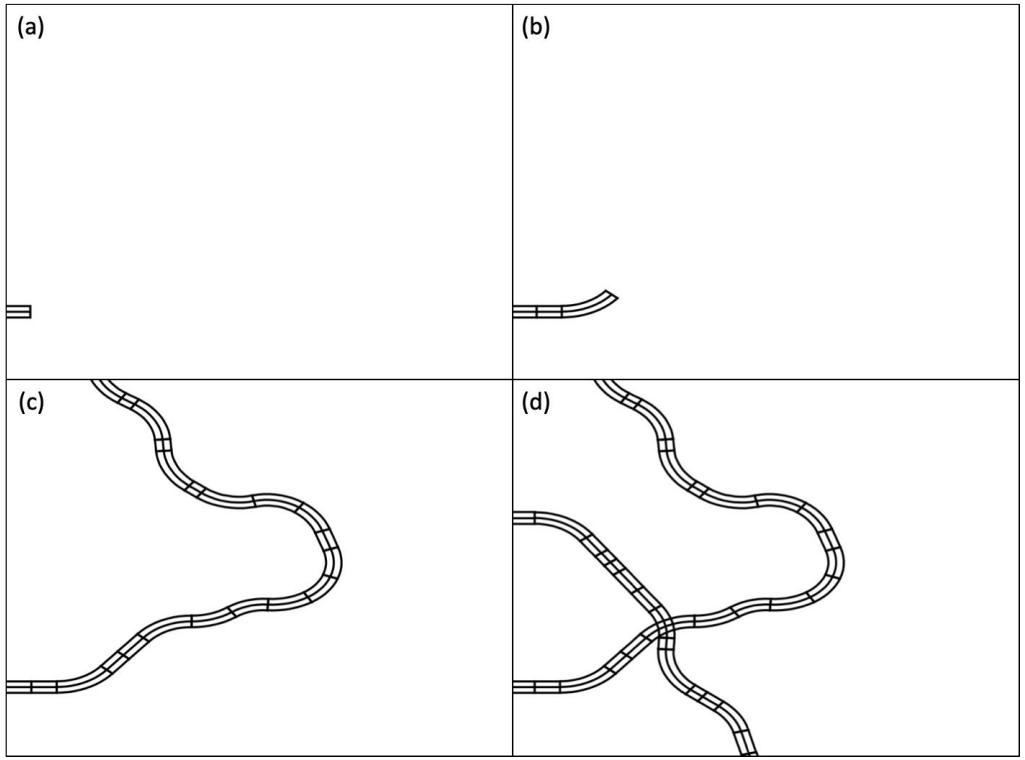
2.2 Road & Road Network Generation

Road generation process

- Create short straight road segment (a).
- Grow the road by generation random road segment (b).
- If the growing road reach boundary, road generation is done (c).

Road network generation process

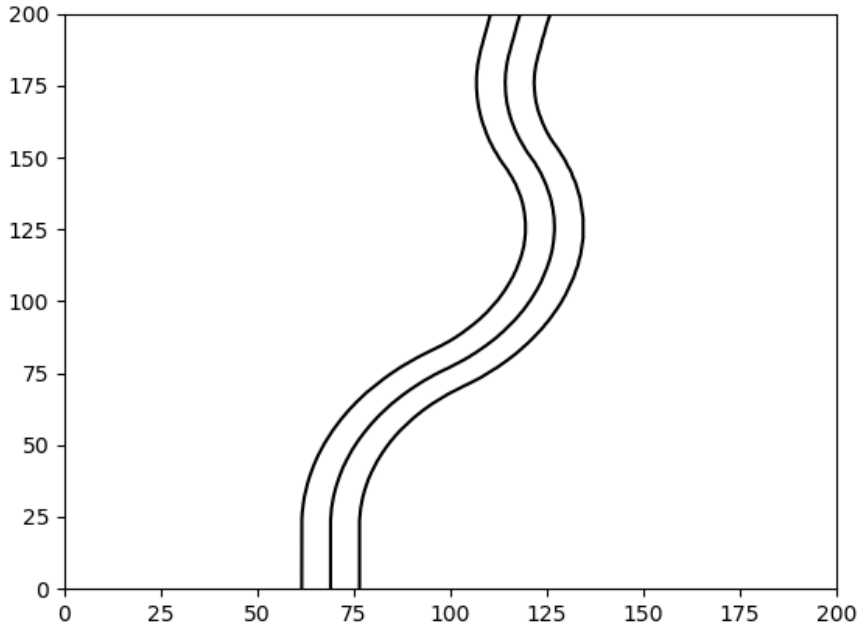
- Add new random roads (d).



2.3 Single Road Validation

Condition 1. Two endpoints of the road must cross the map boundary.

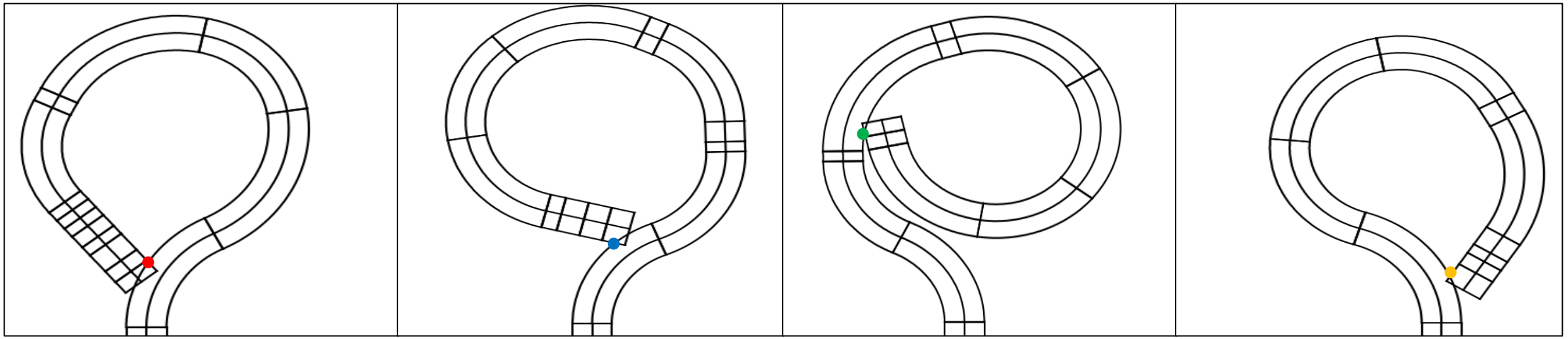
- It always satisfied because the segment is made repeatedly until it reaches the boundary of map.



2.3 Single Road Validation

Condition 2. Self-intersection should not exist.

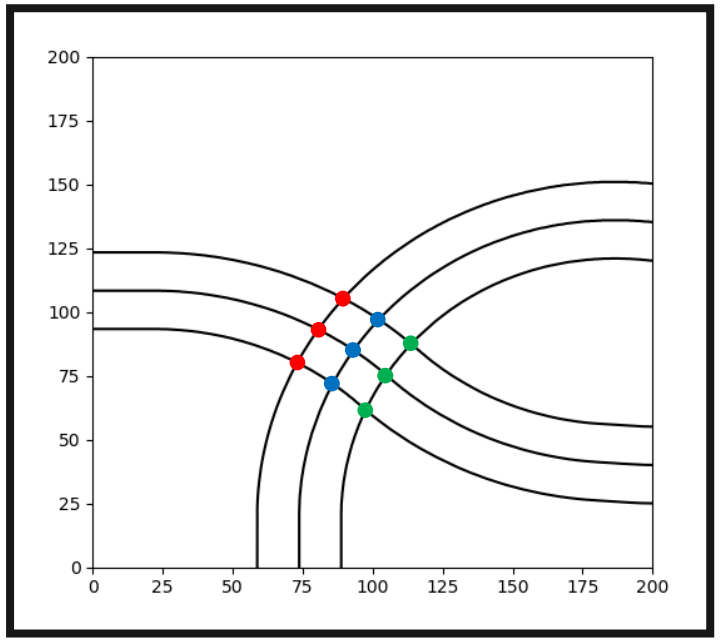
- Self-intersection is a point where two segments constituting a road cross each other.

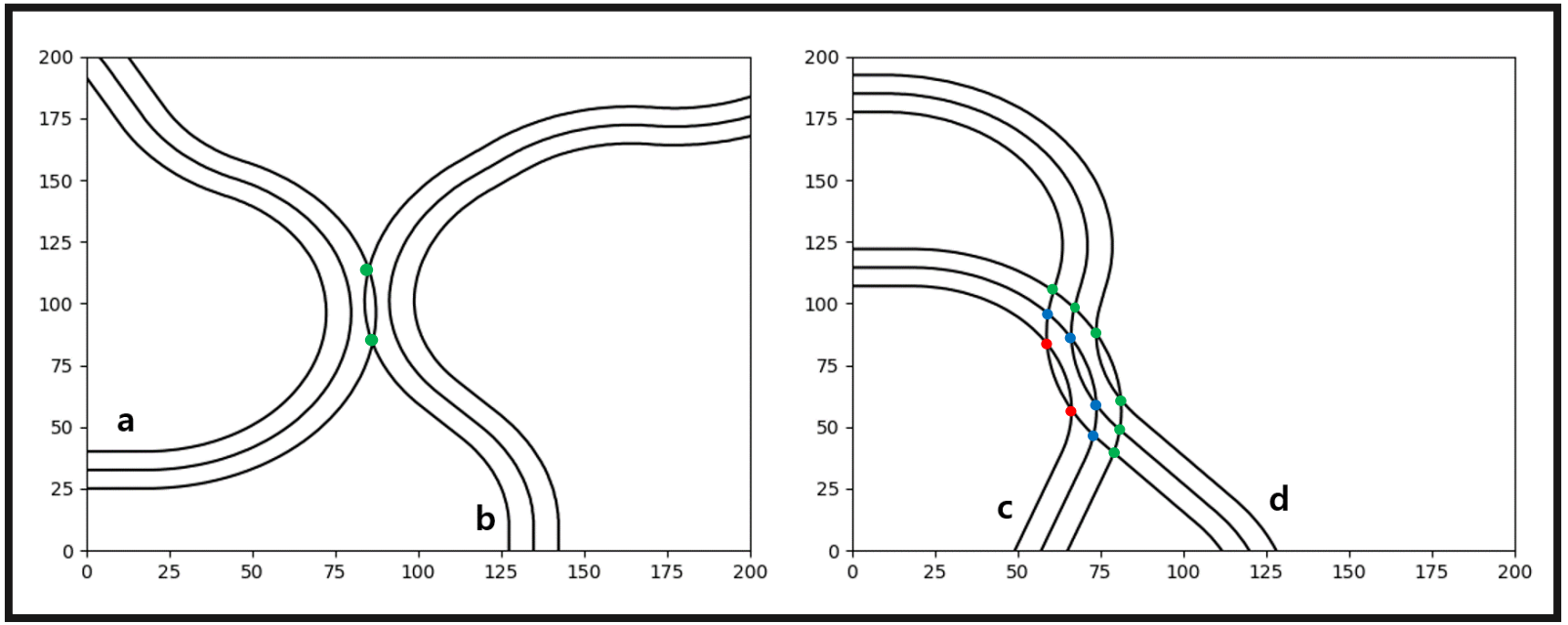


2.3 Road Network Validation

Condition 1. All roads must have only valid intersections.

- A valid road intersection has three cross-points on the left(red), center(blue), and the right(green) lines, respectively.
- If the number of cross-points are different, it is invalid (partial overlap)

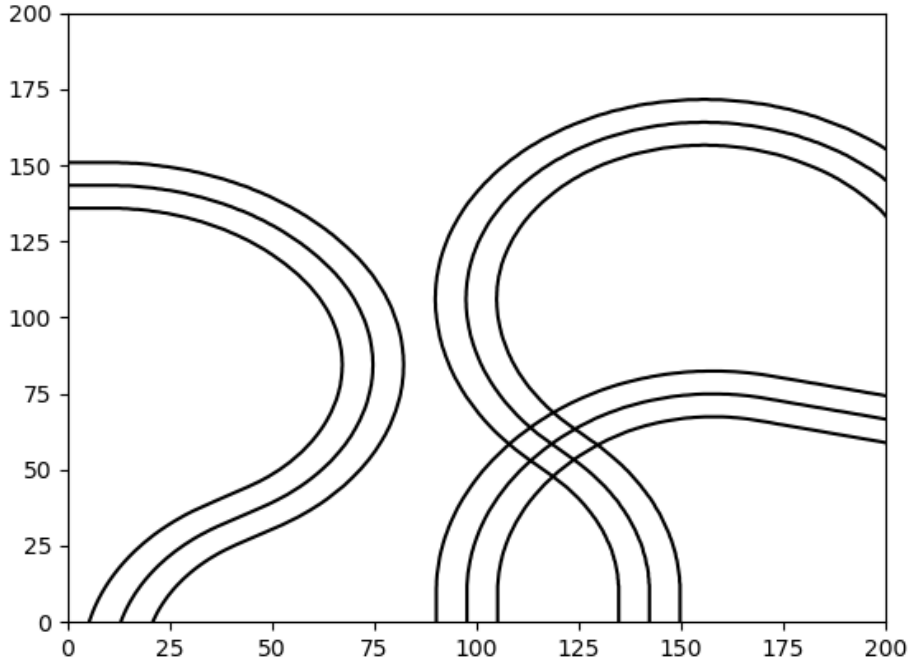


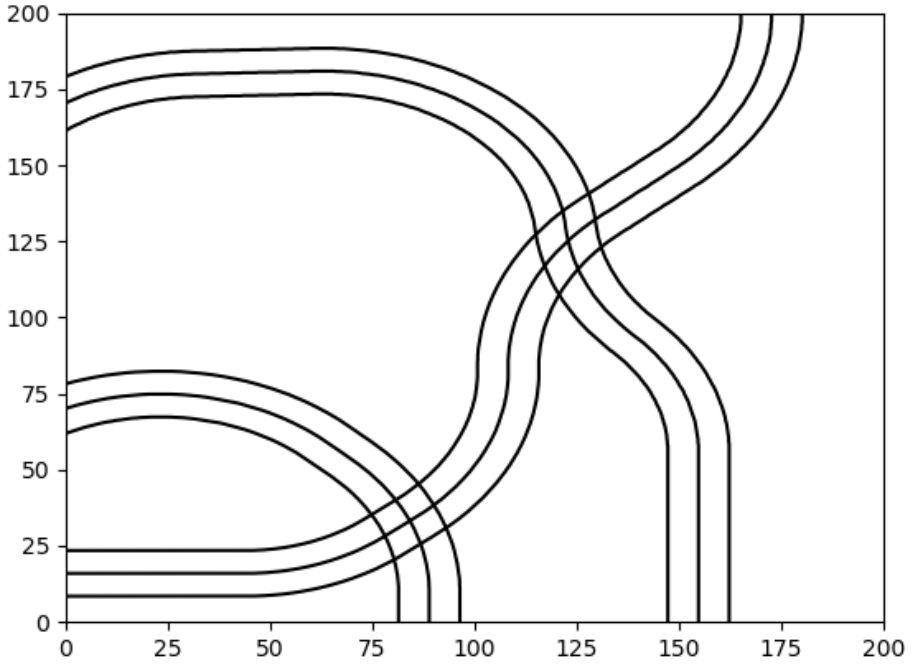


2.3 Road Network Validation

Condition 2. All roads must have at least one intersection.

- A road does not have intersection if the number of cross-points of each line is all 0.





3. Genetic Algorithm

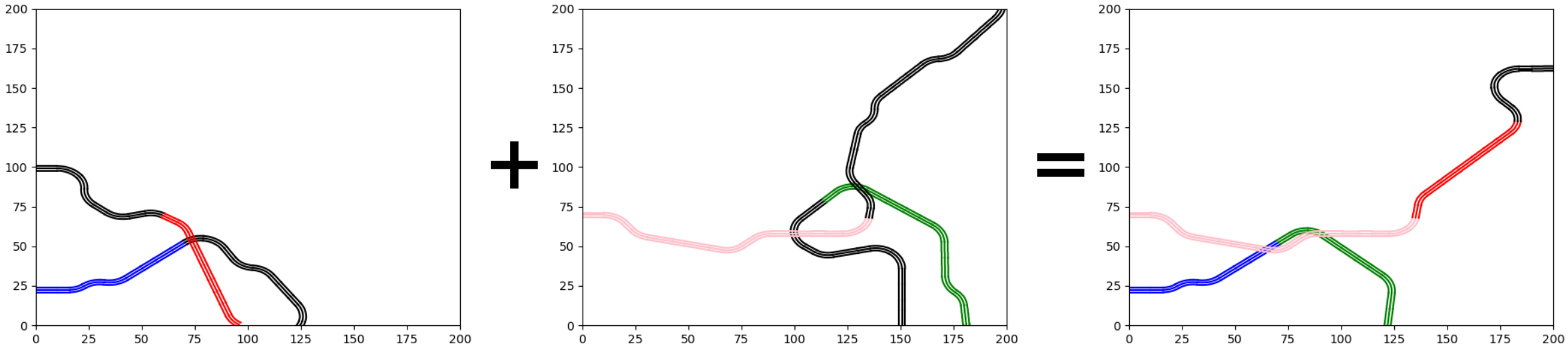
Genetic Algorithm Flow

- Population Size: 25
- Generation: 40
- Small Map: 200*200, Large Map: 400*400
- Using elitism: 10% of best parent
- Using tournament selection: Best one among 5 selected one
- Best path chosen among 3 calculated path

Make New Generation

1. Select two road network from population of current generation randomly.
2. Crossover them and make new road network.
3. Mutation this new road network.
4. Perform validation test and similarity test.
5. Add this road network in population of next generation.

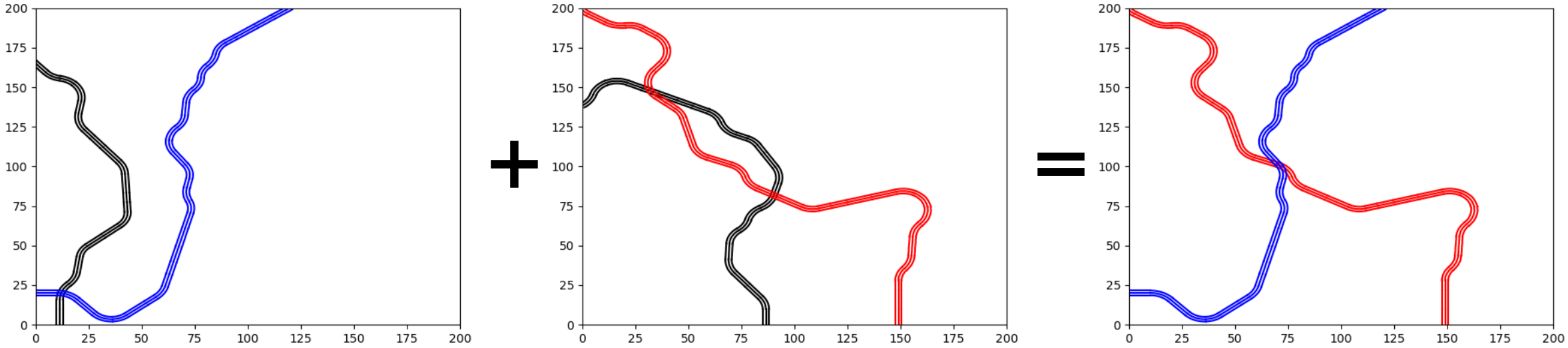
3.1 Join Crossover Method



Join Crossover Process

1. Select one point in all roads in two road network randomly.
2. Cut at the points into two parts for each roads.
3. Cross them and make them connected.

3.2 Merge Crossover Method

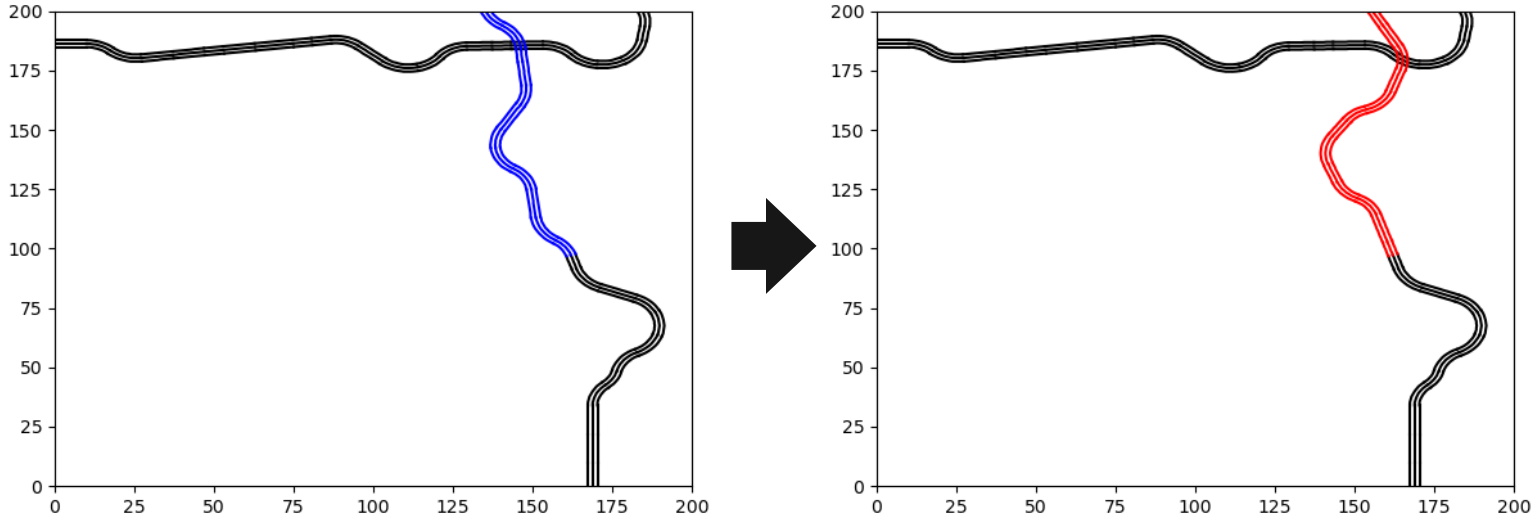


Merge Crossover Process

1. Select one road in each road network randomly.
2. Combine these roads in one road networks.

(This method is only used in multi road network)

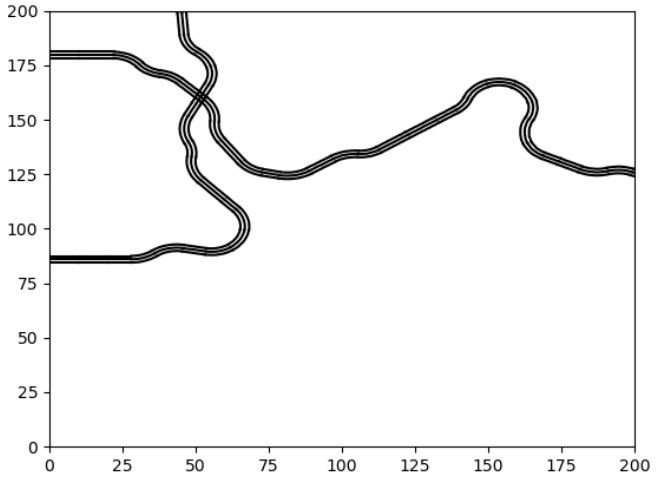
3.3 Mutation Method

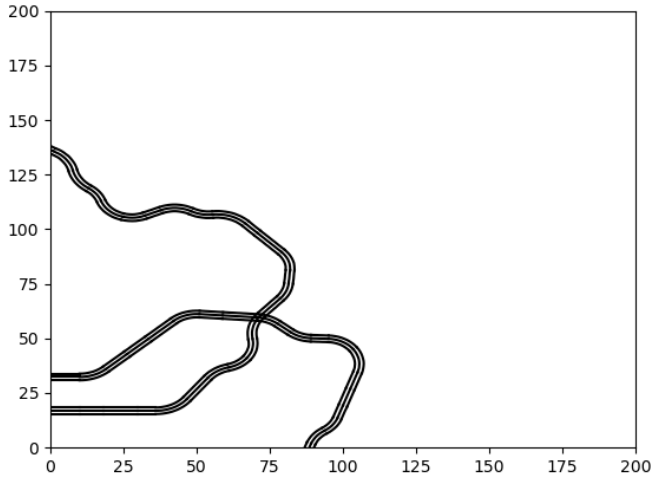


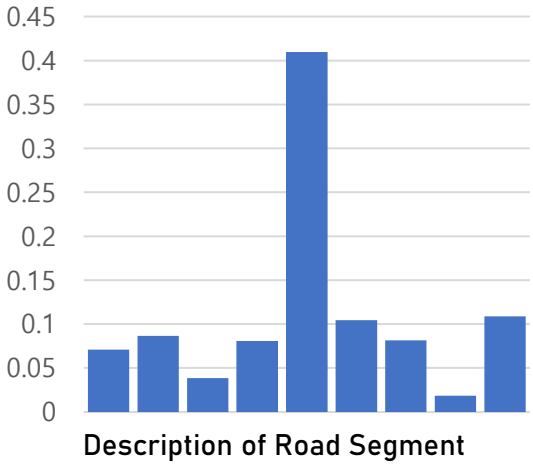
Mutation Process

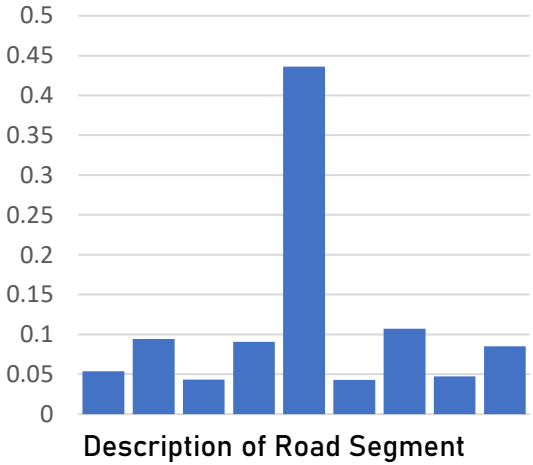
1. Select one segment on road network.
2. Delete all segments of it.
3. Generate random road segment from this section.

3.4 Similarity Test





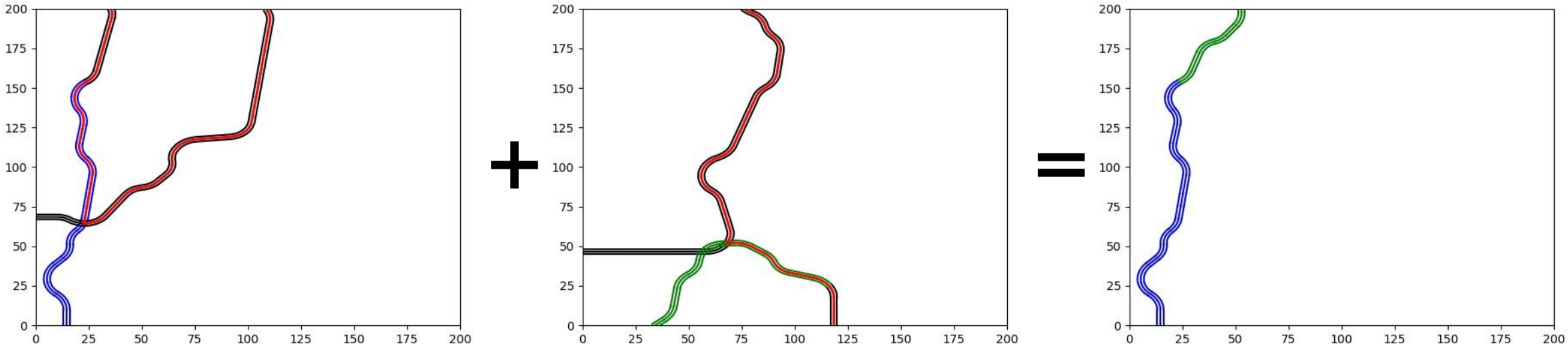




Root Mean Square

Difference : 14.07%

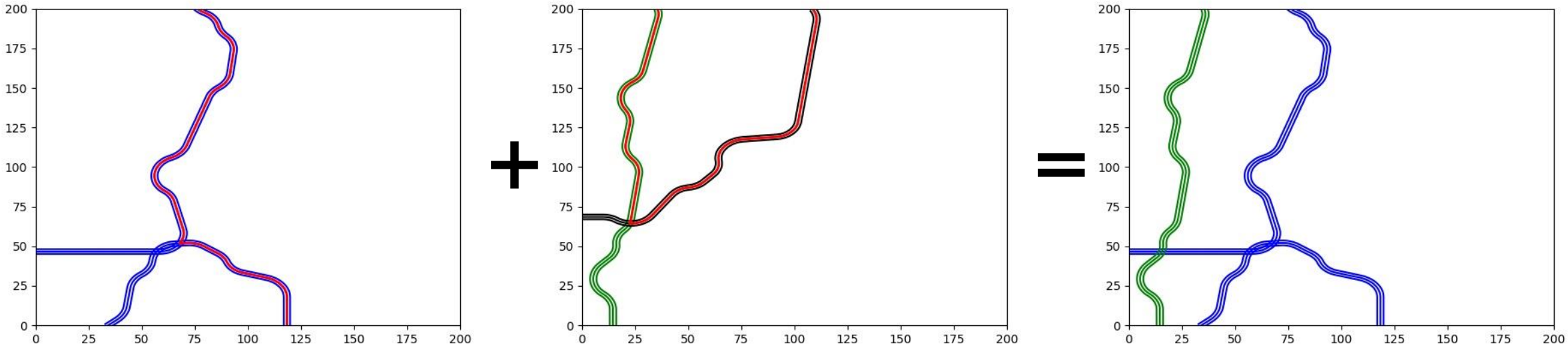
3.5 Improved Join Crossover Method



Improved Join Crossover Operator

- Select crossover point within the path.
- Since fitness is calculated with entity of path, logically enhance meaning of crossover.

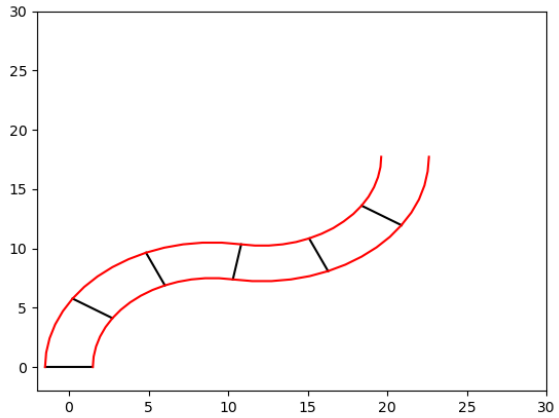
3.6 Improved Merge Crossover Method

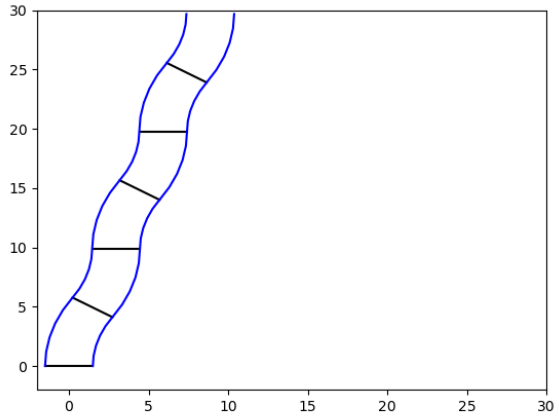


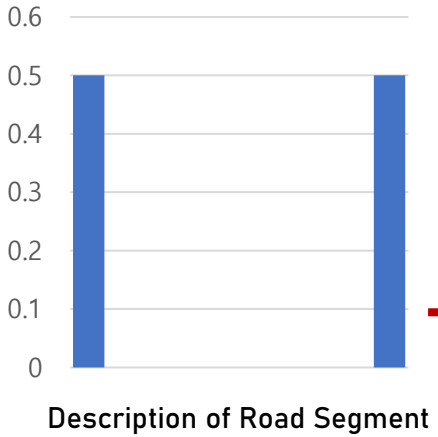
Improved Merge Crossover Operator

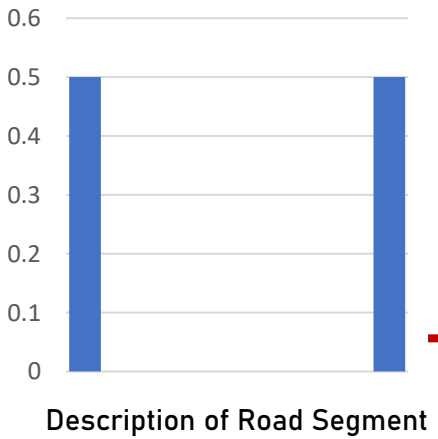
- Select merge road that includes the path.

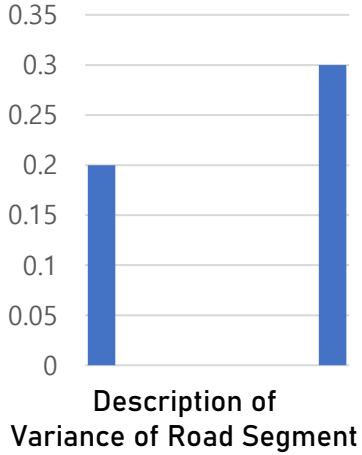
3.7 Improved Similarity Test

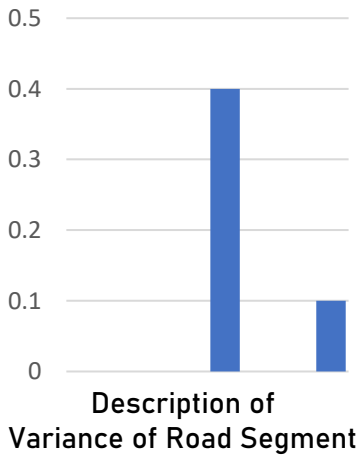












Root Mean Square

[Our Improvement]

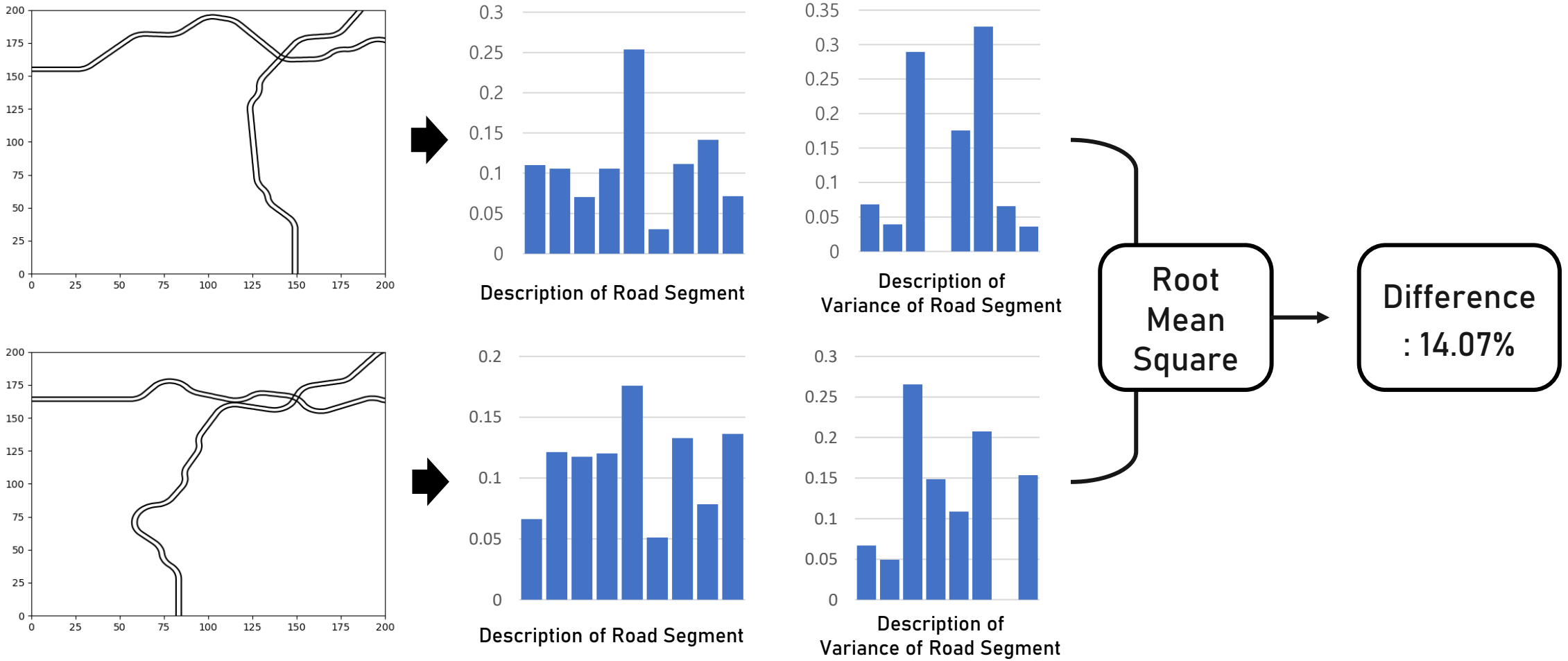
Difference : 48.99%

Root Mean Square

[Original Method]

Difference : 0%

3.7 Improved Similarity Test



4. Evaluation Method

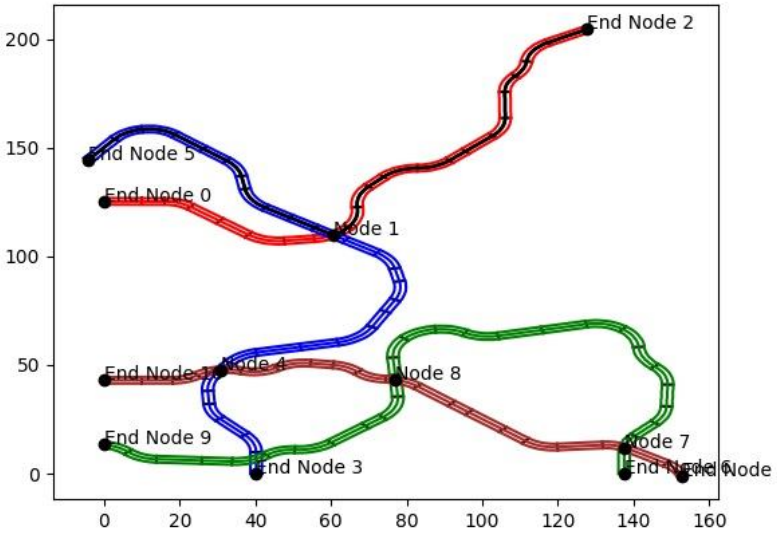
1. Convert generated road into good form to simulate.
   - Rotate road / Make simulation start point to (0,0) point.
   - Generate waypoint.

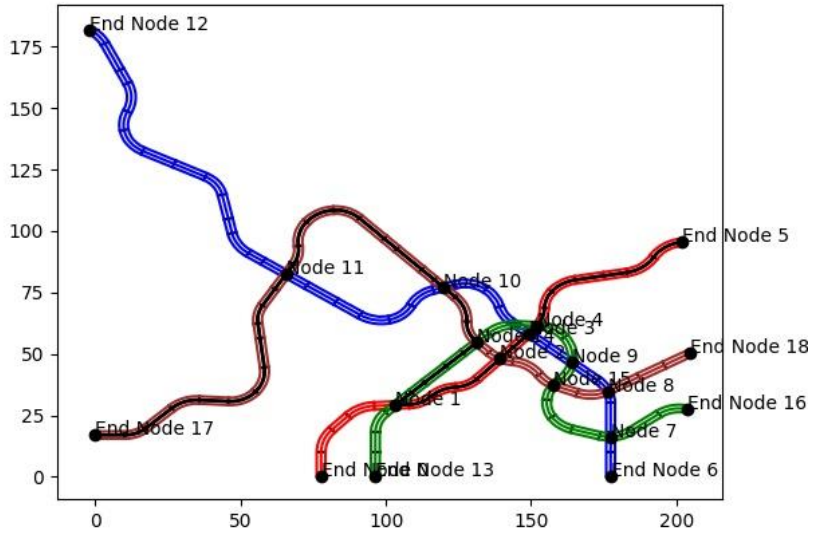
2. GA program send generated road and waypoint to Simulink.
   - Communication protocol (GA → Simulink)

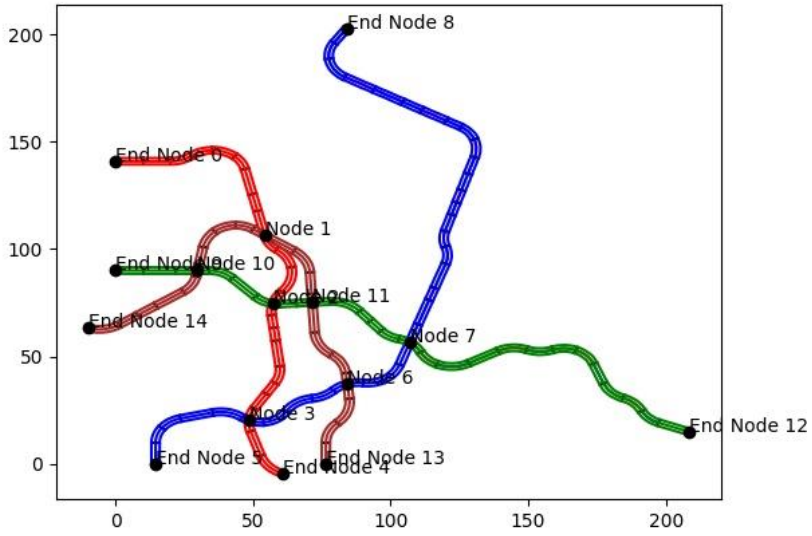
3. Simulink runs the simulation.

4. Simulink sends simulation result to GA program.
   - Communication protocol (Simulink → GA)

4.1 Making Graph



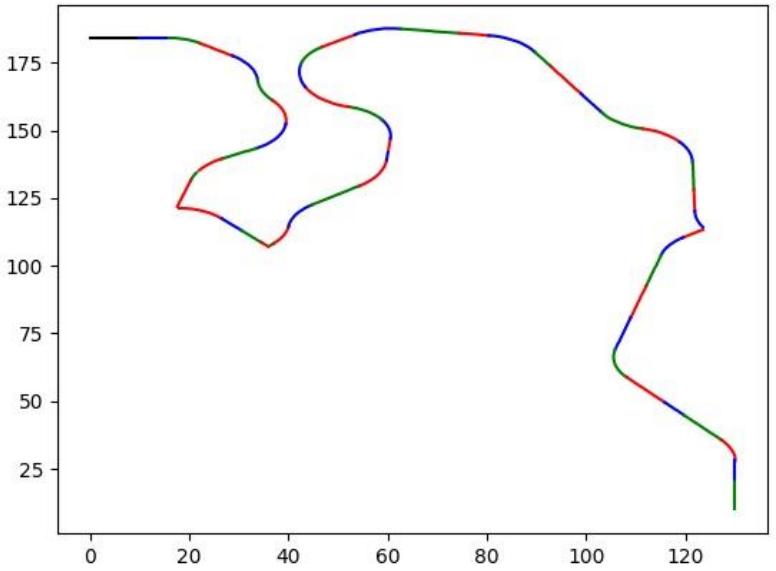


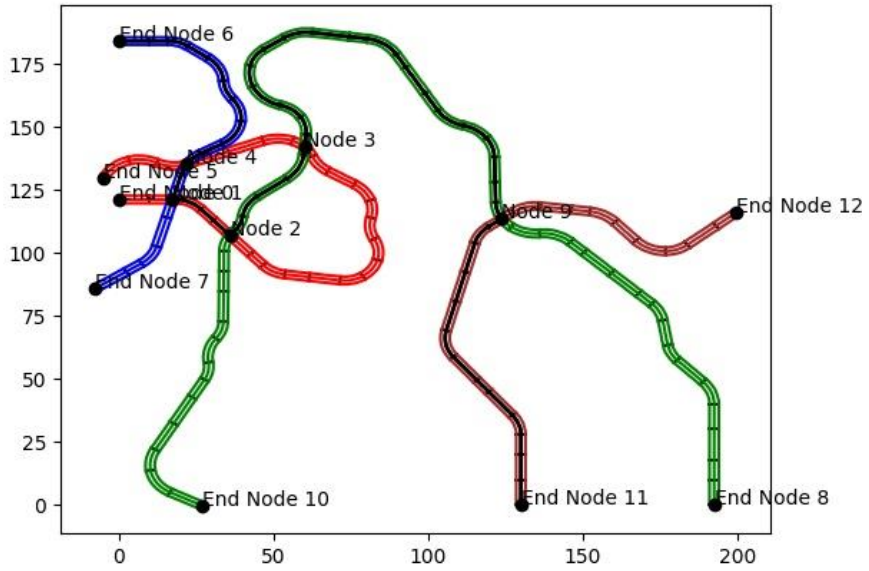


Converting Road to Graph (Priority)

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.

4.2 Finding Path

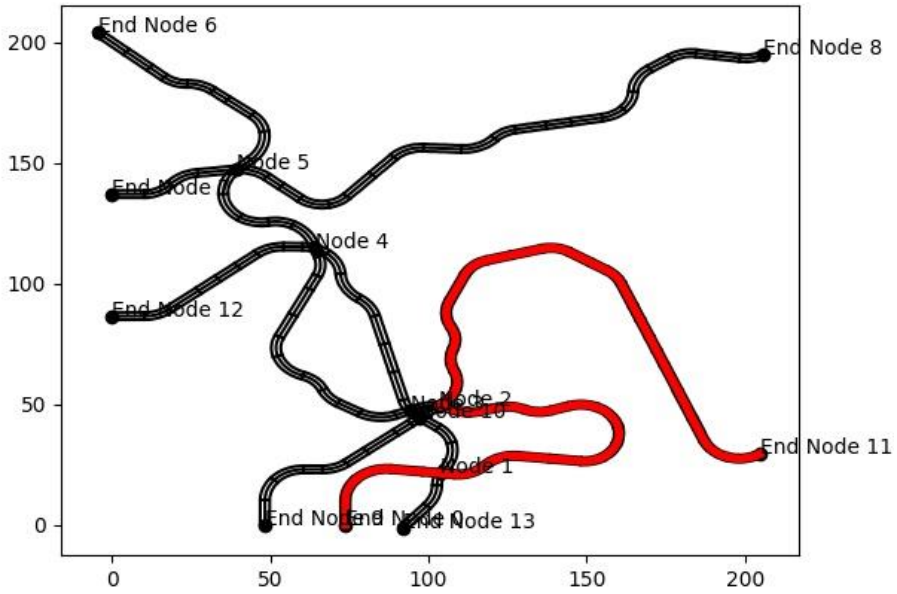




Finding Path (Condition)

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)

4.3 Waypoint Generation



Waypoint: The path that simulator would follow

1. The centerline of the segment between each node would be path.
2. The segment that including Node would be divided to 2 parts and connected.
   (Part 1 of segment 1 → Node → Part 2 of segment 2
   Part 2 of segment 1 and part 1 of segment 2 would not be included in the path.)

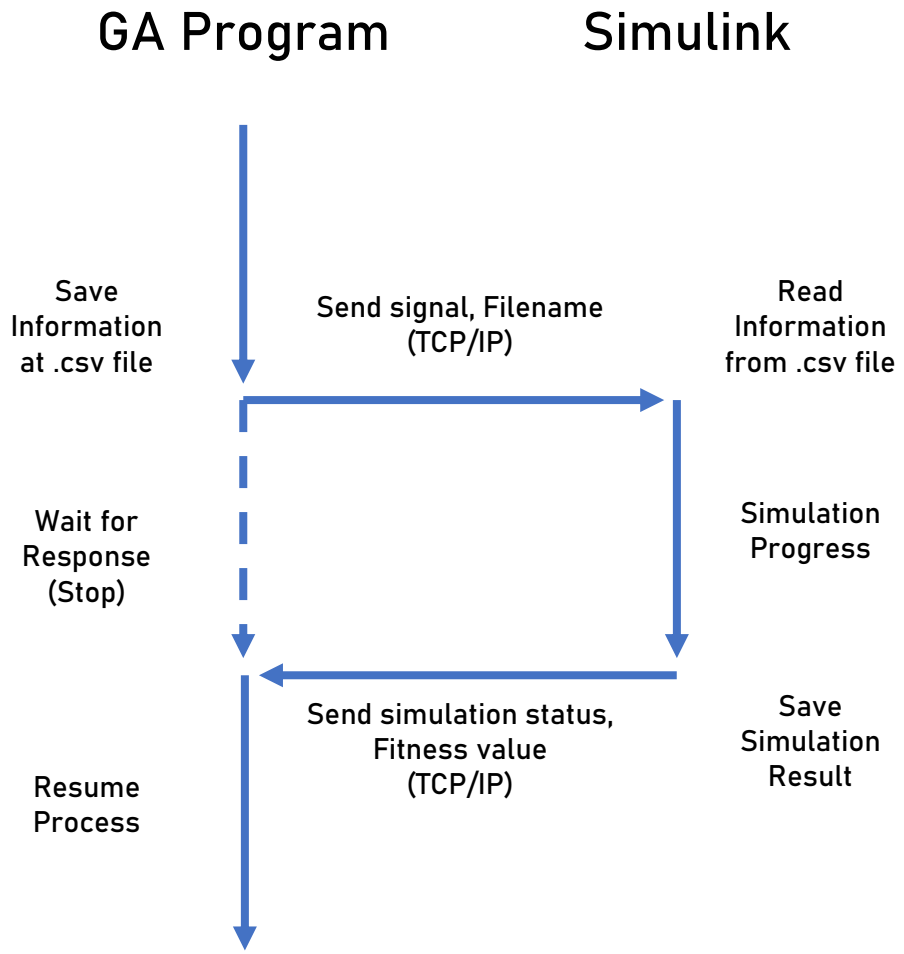
4.4 Communication Protocol

TCP/IP Protocol

- Simulation start / finish signal
- File name of csv file (Simulation data)
- Fitness value (Simulation result)

CSV file at designated path

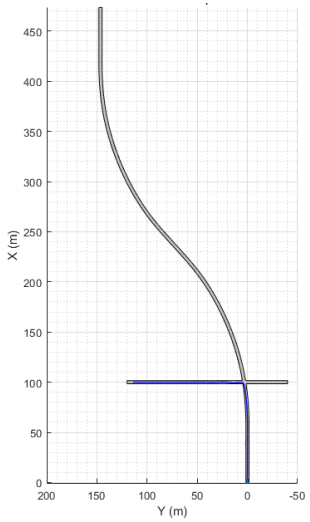
- GA program saves simulation data into csv file.
- Simulink reads csv file to get simulation data.

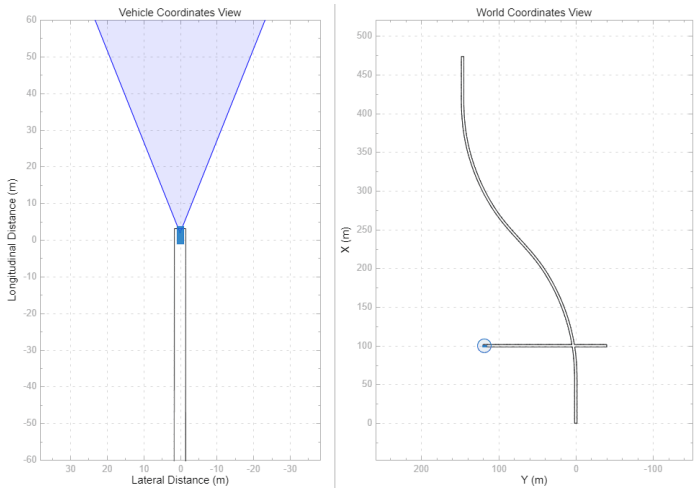


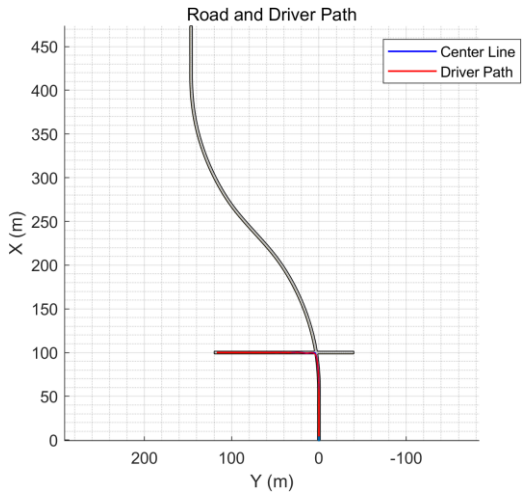
4.5 Simulink

Graphical simulation program based on MATLAB

- Vision based lane keeping system
- Ego-car drives along the virtual roads following the navigation paths
- Observe driving path and lateral deviation during simulation







Get map data

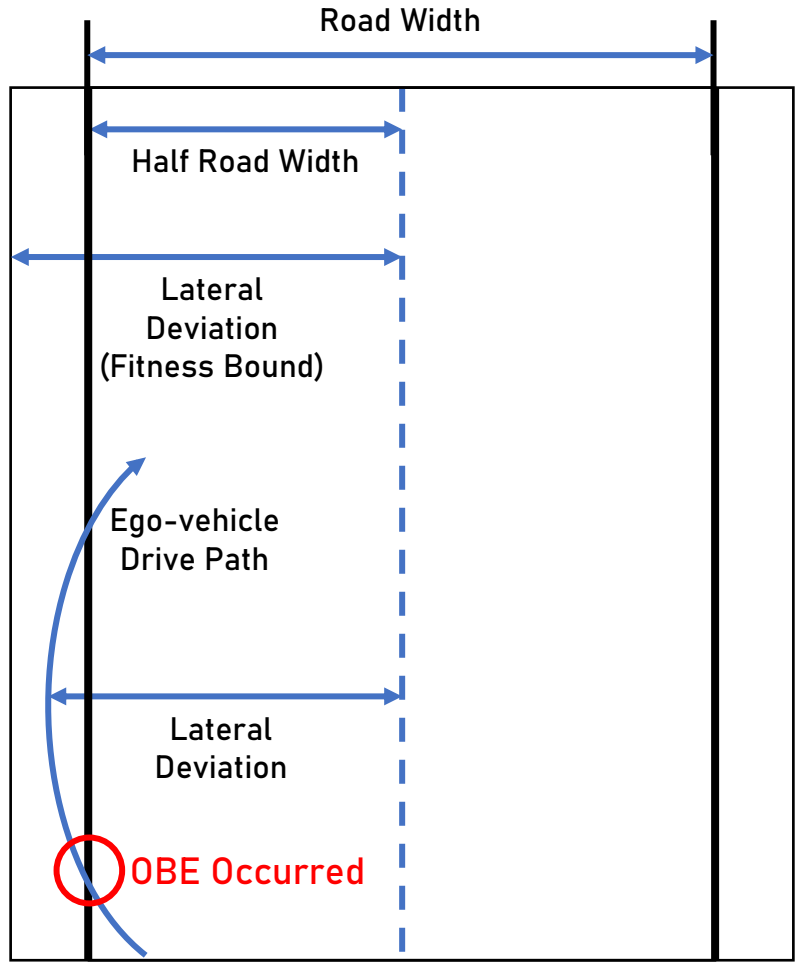
Simulation

Simulation Result

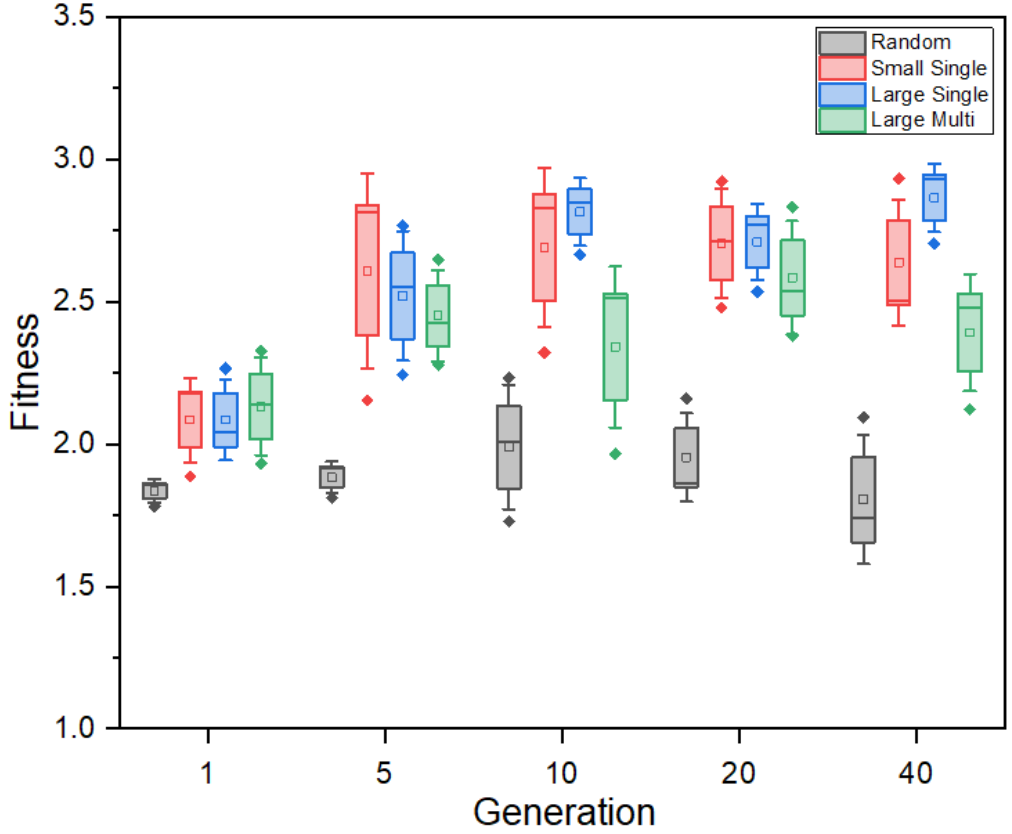
4.6 Fitness Calculation

Bounded lateral deviation

- Calculate fitness value using distance between the center of the road and the ego-vehicle.
- Bound larger than half road width
- OBE occurred if (Fitness Value >= Half Road Width).



5. Evaluation



Verifying K/DAsFault with Random

- Small Single : 1.34 times better
- Single Large : 1.37 times better
- Multi Large : 1.26 times better

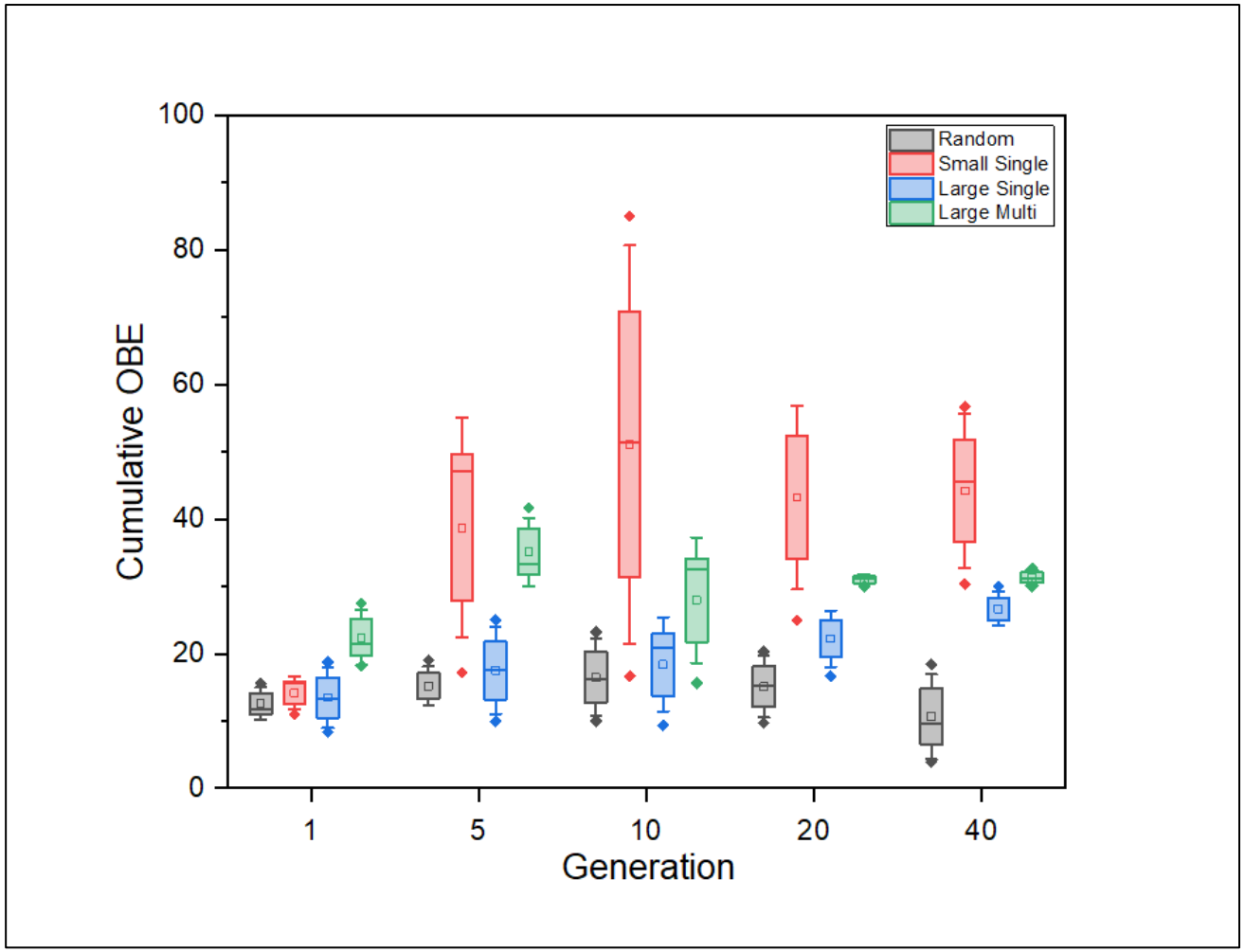
Verifying number of road

- Single road has better fitness than multi road

Verifying map size

- Large map has better fitness than small map on average

5. Evaluation



Verifying K/DAsFault with Random

- Small Single : 2.75 times better
- Single Large : 1.46 times better
- Multi Large : 2.15 times better

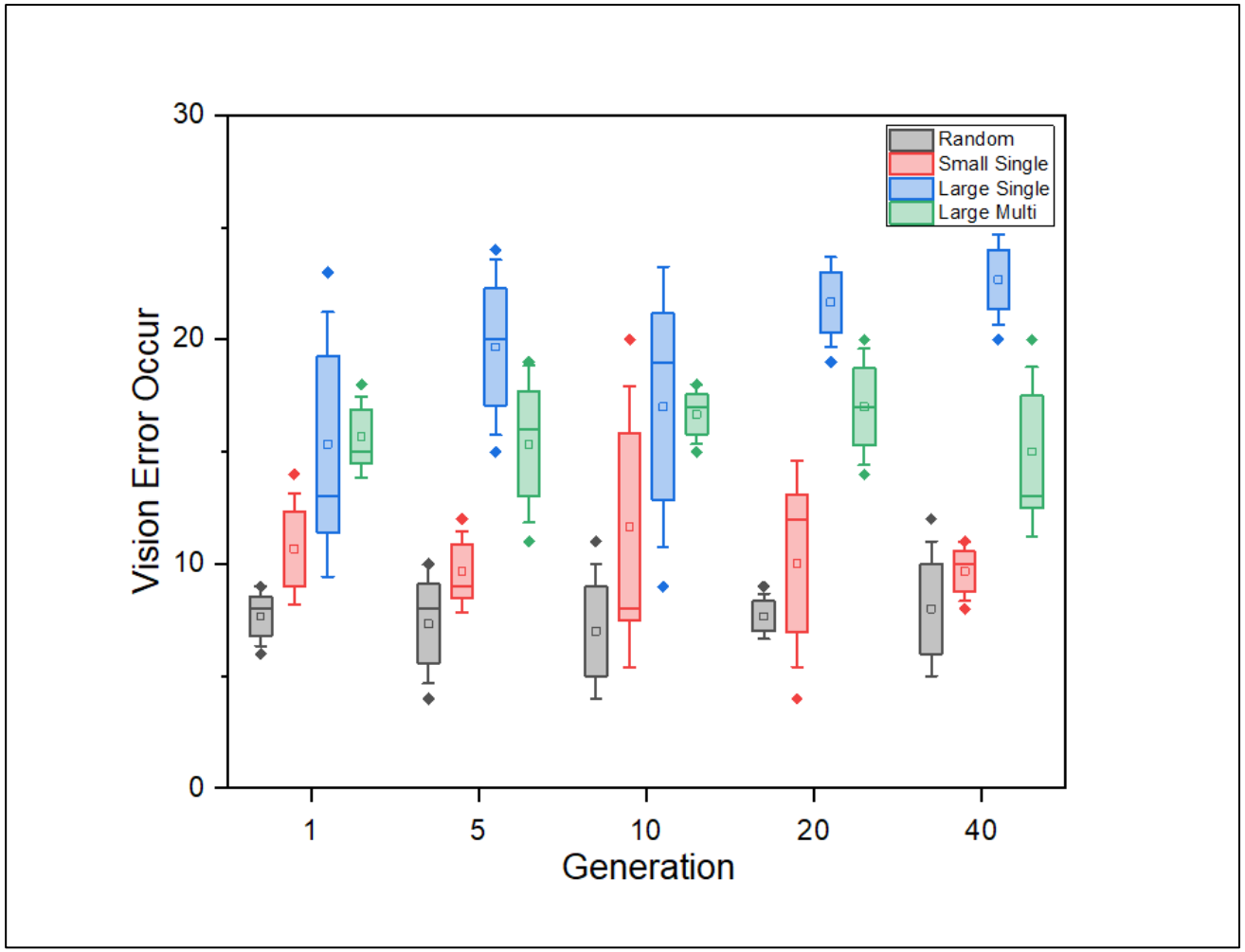
Verifying number of road

- Multi road has more OBEs than single road

Verifying map size

- Small map has more OBEs than large map on average

5. Evaluation



Verifying K/DAsFault with Random

- Small Single : 1.38 times better
- Single Large : 2.55 times better
- Multi Large : 2.12 times better

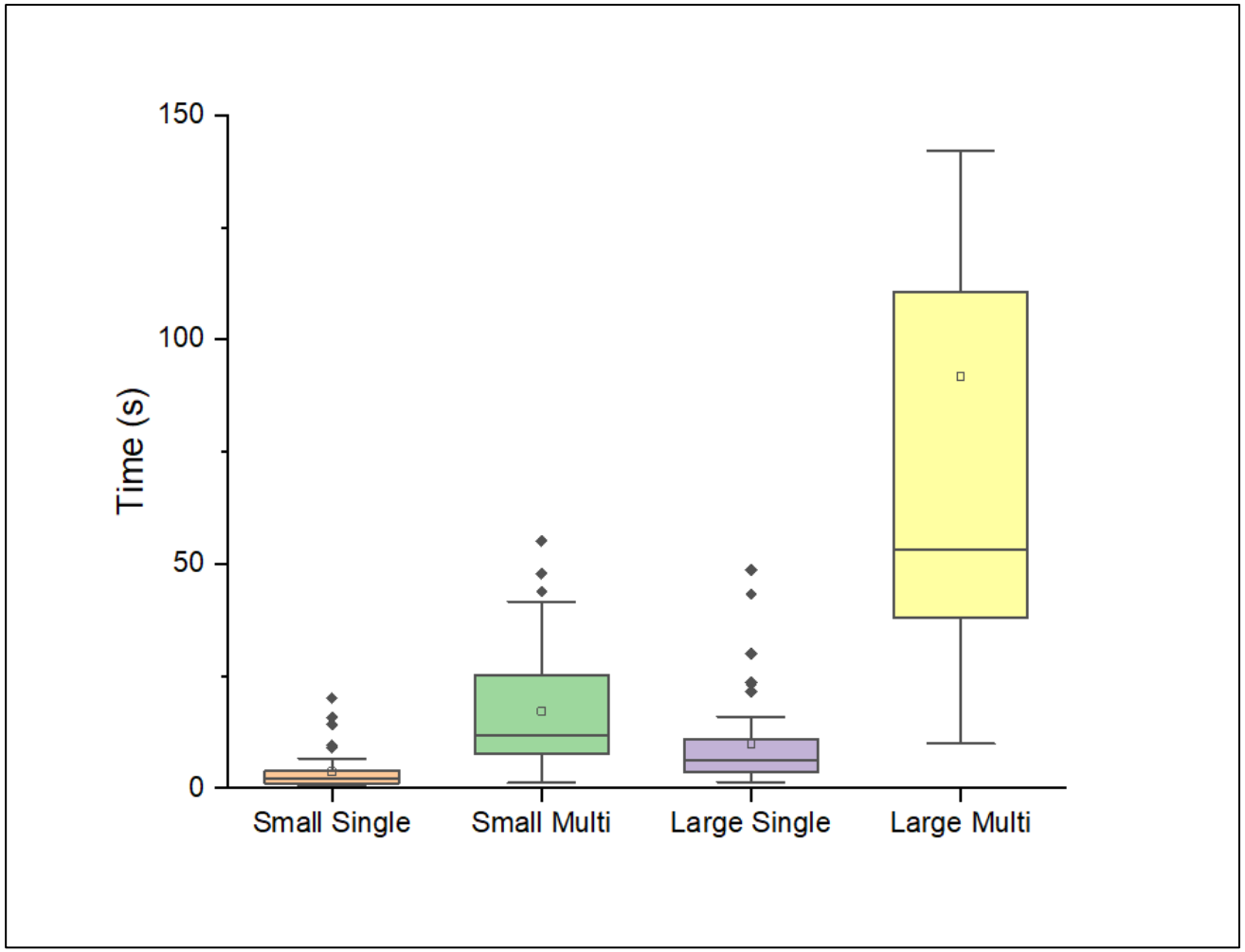
Verifying number of road

- Vision error more occurred within single road than multi road.

Verifying map size

- Vision error more occurred within large map than small map.

5. Evaluation



Small Single: 3.69950 sec

Single Multi : 17.15173 sec

Large Single : 9.79426 sec

Large Multi : 91.83004 sec

6. Discussion

In total, test data generated by GA was better than one for random

- Fitness value was higher at large map
- More OBEs occurred at multi-road map
- More vision error occurred at large map
- Map generation time is longer as size and number of roads in map increases

7. Conclusion

Replicate proposed system for generating test case of lane keeping functionality (AsFault)

- Generate test data with more OBEs and better fitness value compared to random one.

Improve the original AsFault : K-DAsFault

- Propose improved road generator in terms of efficiency
- Propose logically improved search operators and similarity test method
- Propose new evaluation method (Simulator)

Thank You ☺