Software Requirements Specification

for

<Multi-agent reinforcement learning of traffic routing in networks of autonomous vehicles>

Version 1.0 approved

Prepared by <Alon Reicher, Liad Gam>

<Bar Ilan>

<9/8/23>

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

The purpose of this project is to design, develop, and implement a comprehensive simulation-based framework that leverages Multi-Agent Reinforcement Learning (MARL) techniques to optimize traffic routing in networks of autonomous vehicles. By creating an interactive environment where autonomous vehicles learn to make routing decisions based on the environment's properties.

## Intended Audience and Reading Suggestions

This document is intended for customers and for developers who will use the system.

## Product Scope

The product will include a simulation of urban traffic while integrating models of the effect of load on traffic speed.

Utilizing the simulator, the user could run a MARL algorithm for cooperative learning of vehicle routing while using information from other vehicles.

## References

[1] A. Leshem, Y. Sung et al., “Deep multi-agent reinforcement learning of Markov games for distributed traffic management in smart cities”, 2022.

# Overall Description

## Product Functions

The key product functions of the project include:

* Traffic Simulation: The project offers a traffic simulation environment that models the behavior of vehicles in a networked urban setting. Vehicles make routing decisions and respond to dynamic traffic conditions.
* Reinforcement Learning: The framework integrates RL algorithms to enable autonomous vehicles to learn and adapt their routing decisions based on real-time traffic data. This empowers vehicles to collectively optimize traffic flow and reduce congestion.
* Routing Algorithm Evaluation: The project provides the capability to evaluate and compare the performance of different routing algorithms, such as Q-learning and shortest path strategies, in terms of traffic efficiency and travel time reduction.
* Real-Time Traffic Management: Autonomous vehicles within the simulation actively manage traffic flow by adjusting their routes to respond to changing traffic conditions, accidents, and road closures.
* Visualization and Analysis: The project features dynamic visualizations and data analysis tools that allow users to observe the behavior of autonomous vehicles, analyze traffic patterns, and assess the impact of various routing strategies.
* User Interaction: Users can interact with the simulation by configuring simulation parameters, setting traffic scenarios, and observing the simulation in real-time. Users can also customize and control various aspects of the simulation environment.

## User Classes and Characteristics

The users that will be using this product are:

* Developers and Engineers: The project may also be used by developers and engineers working on autonomous vehicle systems and intelligent transportation solutions. They can contribute to the enhancement of the simulation environment and algorithms.
* Transportation Planners: These users are responsible for designing and optimizing urban transportation systems. They use the simulation to assess the impact of different traffic management strategies and routing algorithms on overall network performance.
* E-commercial Companies: The project may by be used by companies in order to manage logistics within their facilities.
* Researchers and Academics: Researchers utilize the simulation framework to conduct experiments, analyze data, and contribute to the advancement of knowledge in the field of autonomous transportation and MARL.
* General Users: Enthusiasts and individuals interested in autonomous vehicles and urban transportation can use the simulation to gain insights into the complex interactions between vehicles and traffic dynamics.

## Operating Environment

Python in a PC + a server.

## Design and Implementation Constraints

TBD

## User Documentation

In the "Simulator HLD" document.

## Assumptions and Dependencies

TBD

# External Interface Requirements

## User Interfaces

The project provides user interfaces that offer intuitive interaction and visualization capabilities for different user classes:

* Simulation Configuration Interface: Users can configure simulation parameters such as road network layout, and time of day. The interface allows users to set up various traffic scenarios to analyze the impact of different conditions on traffic flow.
* Real-Time Visualization Interface: The real-time visualization dashboard provides an interactive map display of the simulated urban environment. Users can observe the movement of autonomous vehicles, monitor traffic congestion, and assess the effectiveness of MARL algorithms.
* Analysis and Metrics Interface: Researchers and policymakers can access a dedicated interface to analyze simulation results and performance metrics. This interface allows users to generate reports, graphs, and charts to understand traffic patterns, travel time reductions, and algorithm effectiveness.

## Software Interfaces

* Python interpreter – Python 3.10
* Python IDE - PyCharm
* Python libraries –
  + Osmnx 1.6.0
  + Matplotlib 3.6.2
  + Datetime
  + Pandas 1.5.2
  + Json
  + Networkx 3.0
  + copy
  + Numpy 1.24.1

## Communications Interfaces

TBD

# System Features

## Traffic flow simulation

### Description and Priority

The Traffic Flow Simulation feature involves a high-priority simulator designed to replicate real-world traffic dynamics based on various parameters, such as the number of vehicles, road characteristics, weather conditions, and more.

**Requirements:**

1. Road's parameters
   1. Id
   2. Source node
   3. Destination node
   4. Type (e.g., highway)
   5. Length
   6. Speed limit
   7. Current speed
   8. Current estimated time
2. Car's parameters
   1. Id
   2. Starting node
   3. Destination node
   4. Current road
   5. Starting time
   6. Drive time
   7. Time until passing the current road
   8. Route
3. Graph's requirements
   1. Scalability
   2. Transforming real life maps into graph

## Visual representation of simulation

**4.2.1 Description and Priority**

The Visual Representation of Simulation feature provides a 2D graphical depiction of the ongoing simulation, offering an intuitive overview of the dynamic traffic scenarios.

This feature holds medium priority.

**Requirements:**

**1.Statistical Insights:**

a. Agent Speed Over Time

b. Agent Distance Covered Over Time

c. Average Travel Time Calculation

d. Road Speed Variation Over Time

e. Road Distance Covered Over Time

f. Number of Cars on a Road Over Time

g. Road Density Changes Over Time

h. Alignment of Visualization with Simulation Time Scale

**2.Map Visualization:**

a. Depiction of Road Conditions (Clear, Congested)

b. Illustration of Alternative and Original Agent Routes

c. Visualization of Traffic Light

d. Zooming Functionality for Adjusting View

**Interactive Interface:**

a. Toggle and Adjustment Options for Road and Agent Parameters

b. Customizable Settings for Simulation Parameters

c. Control for Adjusting Simulation Speed (Slow Down/Fast Forward)

## Multi agent reinforcement learning

4.3.1 Description and Priority

The Multi-Agent Reinforcement Learning feature integrates q learning algorithm to empower vehicles (agents) within the simulation to learn optimal navigation strategies, enhancing overall traffic flow through collaborative learning.

This feature is of high priority.

############

1. Penalty for agents using toll roads.
2. Creating sub goals for agents.
3. Collaborative communication of the agents.
4. Learning optimal path finding strategies under communication limitations.
5. Scalable network optimization via MARL with low complexity.
6. Exploiting the periodicity of the transportation.

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## ~~Safety Requirements~~

TBD

## ~~Security Requirements~~

TBD

## ~~Software Quality Attributes~~

TBD

## ~~Business Rules~~

TBD

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

**Annex**

**Ground Assumptions**

**Traffic axioms**

* Every car that enters the road will also leave it.
* Every car has an origin and a destination.The assumption is that there is one or more paths from origin to destination exists.
* Every start and end points are vertices. We assume that a vertex is connected to 2 edges at least, and an edge is a road (2 edges vertex- dead end: edge in and edge out).
* Car won't stop moving towards its end goal.
* Every edge has a certain number of cars in the free flow area. There will be a queue for the free flow area to determine cars' speed. The speed is the same for all cars.

**Agents' axioms**

* decision of next edge must be before previous edge turning queue at latest.
* Every "smart" agent in the simulator has the same thought process a smart agent could dynamically change his path according to information from other smart agents. Other "dumb" agents will determine an initial path and won't change it.

**make a power point animation of a graph of 2 possible car flows. Useful to understand, animation is very helpful.**

**Map generating method**

* The user will write as an input the required number of intersections (nodes).
* The simulator will generate a graph including the said number of nodes and random edges.
* The user will be able to save a map and load it.

\*Maybe not only random models

* The roads graph will be a directed graph, and every edge will have an "opposite" which has the same basic stats (length, max speed etc…) but is in the opposite direction.
* Each node will have a maximum of 4 next edges and minimum 1 next edge (representing real life junction).
* Edge is made of queues of each direction, and a single-lane movement part. The queues will be at the end of the edge, representing the stopping line before the junction. The queue is infinite.

There will be a different queue for every next immediate edge (one per edge).