T.C. AYDIN ADNAN MENDERES UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

CSE401 Graduation Thesis 2, Spring 2023
Supervisor: Hüseyin ABACI

ProjectCode: Analysis of Ambulance Response Times through Traffic Simulation Final Report

(Bachelor of Science Thesis)

13.06.2023 (Presentation date)

By:
Deniz DOĞAN, Student ID: 181805011
Ahmet TIK, Student ID: 181805034

PLAGIARISM STATEMENT

This report was written by the group members and in our own words, except for quotations from published and unpublished sources which are clearly indicated and acknowledged as such. We are conscious that the incorporation of material from other works or a paraphrase of such material without acknowledgement will be treated as plagiarism according to the University Regulations. The source of any picture, graph, map or other illustration is also indicated, as is the source, published or unpublished, of any material not resulting from our own experimentation, observation or specimen collecting.

Project Group Members:

Name, Lastname	Student Number	Signature	Date
Deniz DOĞAN	181805011		
Ahmet TIK	181805034		

Project Supervisors:

Name, Lastname	Department	Signature	Date
Hüseyin ABACI	Computer Engineering		

ACKNOWLEDGEMENTS

"We would like to express our deepest gratitude to all those who have helped and supported us throughout the course of this project.
First and foremost, we would like to thank our project supervisor for their guidance and support throughout the project.
We are also grateful to the simulation software developers for providing us with the tools necessary to model and analyze urban mobility. In addition, we would like to acknowledge the video creators for Sumo education for providing us with valuable resources and tutorials that helped us in understanding the simulation software.
Finally, we would like to thank our friends and families for their support and understanding during the time we dedicated to this project.
This project would not have been possible without the contributions and support of all of these individuals and groups. Thank you."

KEYWORDS

1. Traffic Simulation
2. Ambulance Response Times
3. SUMO (Simulation of Urban MObility)
4. TRACI (Traffic Control Interface)
5. Route Optimization
6. Emergency Vehicle Routing
7. Urban Traffic Management
8. Traffic Modelling
9. Emergency Response Efficiency
10. Real-time Dispatch Optimization
11. AI in Traffic Management

ABSTRACT

The efficacy of ambulance services in urban environments is a critical aspect of public health, with response time often being a key determinant of patient outcomes. To study and potentially improve these services, we have developed a traffic simulation model using Simulation of Urban Mobility (SUMO), an open-source, highly portable, microscopic, and continuous road traffic simulation package, and Traffic Control Interface (TraCI), an interface for remote control of SUMO. The simulation models the movement of five ambulances from different starting points towards a common endpoint within a realistic urban traffic scenario.

This simulation model incorporates complex traffic dynamics, including road networks, traffic signals, speed limits, and traffic behavior. It also includes specific characteristics related to the ambulances, such as their capacity for maximum speed and emergency deceleration. The project involves defining the ambulance routes, implementing the simulation, collecting data about each ambulance's speed, distance traveled, time, and type at each simulation step, and storing this data in an Excel file for further analysis.

The results of this simulation can provide insights into the factors influencing ambulance response times in urban environments. This can guide urban planners and emergency services in improving response strategies, ultimately leading to better patient outcomes. Furthermore, this project lays the foundation for more complex simulations that could incorporate unpredictable events or apply machine learning techniques to the data for predictive analysis.

ÖZET

Ambulans hizmetlerinin kentsel ortamlardaki etkinliği, halk sağlığının kritik bir yönüdür ve müdahale süresi genellikle hasta sonuçlarının önemli bir belirleyicisidir. Bu hizmetleri incelemek ve potansiyel olarak iyileştirmek için, açık kaynaklı, son derece taşınabilir, mikroskobik ve sürekli bir karayolu trafik simülasyon paketi olan Simulation of Urban Mobility (SUMO) ve Traffic Control Interface (TraCI) kullanarak bir trafik simülasyon modeli geliştirdik. SUMO'nun uzaktan kontrolü için arayüz. Simülasyon, gerçekçi bir kentsel trafik senaryosunda farklı başlangıç noktalarından ortak bir son noktaya doğru beş ambulansın hareketini modeller.

Bu simülasyon modeli, yol ağları, trafik sinyalleri, hız limitleri ve trafik davranışı dahil olmak üzere karmaşık trafik dinamiklerini birleştirir. Ayrıca, maksimum hız ve acil durum yavaşlama kapasiteleri gibi ambulanslarla ilgili belirli özellikleri de içerir. Proje, ambulans güzergâhlarının tanımlanmasını, simülasyonun uygulanmasını, her simülasyon adımında her ambulansın hızı, kat edilen mesafe, süre ve türü hakkında veri toplanması ve bu verilerin daha fazla analiz için bir Excel dosyasında saklanmasını içerir.

Bu simülasyonun sonuçları, kentsel ortamlarda ambulans müdahale sürelerini etkileyen faktörler hakkında fikir verebilir. Bu, şehir planlamacılarına ve acil servislere müdahale stratejilerini geliştirmede rehberlik edebilir ve sonuç olarak daha iyi hasta sonuçlarına yol açabilir. Ayrıca bu proje, öngörülemeyen olayları içerebilen veya tahmine dayalı analiz için makine öğrenimi tekniklerini verilere uygulayabilen daha karmaşık simülasyonların temelini atıyor.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

- ITS: Intelligent Transportation Systems
- SUMO: Simulation of Urban MObility
- V2I: Vehicle-to-Infrastructure
- V2V: Vehicle-to-Vehicle
- GPS: Global Positioning System
- CPU: Central Processing Unit
- GUI: Graphical User Interface
- API: Application Programming Interface
- EMS: Emergency Medical Service
- LOS: Level of Service
- VISSIM: Vienna Intelligent Transport Systems Simulator
- IVC: Inductive Vehicle Communication
- DTA: Dynamic Traffic Assignment

1. INTRODUCTION

1.1. Description of the Problem

The effective delivery of emergency services in urban areas is a critical challenge faced by cities across the globe. The most crucial factor that determines the effectiveness of these services, particularly ambulance services, is response time. Urban environments pose various challenges that can lead to significant delays in ambulance response times. Traffic congestion, complex road network designs, differing distances from hospital locations, and unpredictability in road conditions are among these challenges. This project seeks to address these challenges by simulating and analyzing ambulance response times within a virtual urban traffic network.

1.2. Project Goal

The main objective of this project is to create a robust traffic simulation environment where the movements of multiple ambulances from distinct starting points to a single destination point can be analyzed. The goal is to understand the dynamics that contribute to ambulance response times, capture the variables involved, and use this understanding to determine optimal routes and strategies for minimizing response times. The simulation environment will replicate realistic urban traffic conditions to ensure that the findings and insights gained are applicable to real-world scenarios.

1.3. Project Output

The primary output of this project is a comprehensive dataset capturing key details about the simulated ambulance movements. This dataset will include ambulance IDs, their speeds at different points in time, distances traveled, times taken to reach the destination point, and the type of vehicle. The data will be recorded and stored in an Excel file, which will serve as a valuable resource for further analysis and research. Ultimately, this data can contribute to enhancing emergency response strategies and improving patient outcomes.

1.4. Project Activities and Schedule

The project will proceed in several clearly defined phases:

- *Phase 1 (September 2022 - October 2022)*: The initial phase involves the definition of simulation parameters, including the configuration of the simulation environment, setting up the starting points for the ambulances, and defining the endpoint.
- *Phase 2 (November 2022 – December 2022)*: The next phase entails the implementation of the simulation model using the Traffic Control Interface (TraCI) and Simulation of Urban Mobility (SUMO) tools. This involves scripting the routes for each ambulance and adding them to the simulation.
- *Phase 3 (January 2023 - February 2023)*: During this phase, the simulation is run for a predetermined number of steps. Each step corresponds to a unit of time, and at each step, the simulation updates the state of each vehicle and records data about its speed, distance traveled, and time taken.

- *Phase 4 (March 2023 - April 2024)*: This phase involves the comprehensive analysis of the collected data. Patterns and relationships will be identified, and insights into potential optimizations for ambulance routes will be developed.
- *Phase 5 (May 2024 - June 2024)*: In the final phase, a report will be prepared, presenting the findings of the project, the implications of these findings for emergency response strategies, and recommendations for future work.

.

2. DESIGN

2.1. High Level Design

The project's high-level design encompasses the setting up of a complex traffic simulation where multiple ambulances navigate through a traffic network. This simulation leverages the capabilities of SUMO, a powerful traffic simulation package, and TraCI, an interface that allows for interactive control of the simulation. Each ambulance, starting from a unique point, moves towards a common endpoint.

2.2. Detailed Design

The detailed design of the project is as follows:

- *Configuration of SUMO*: This involves using a configuration file to define the parameters of the traffic simulation environment. These parameters include the layout of the urban road network, traffic density, speed limits, and other factors that closely mimic real-world traffic conditions.
- *Defining the Endpoint and Starting Points*: The endpoint, where all ambulances are headed, and the starting points of the ambulances are defined as coordinates in the traffic network.
- *Creating the Routes and Vehicles*: This is achieved using the TraCI commands within the Python scripting environment. Each vehicle follows a defined route from its respective start point to the endpoint. The parameters for each vehicle, including its ID, type, departure time, maximum speed, and preferred lane, are defined within the script.
- *Running the Simulation*: The simulation is run for a predefined number of steps, with each step representing a unit of time. At each step, the simulation updates, and the state of each vehicle changes according to the traffic conditions and the vehicles' pre-set behaviors.
- *Data Collection*: As the simulation runs, data is collected and stored in an Excel file. This data includes each vehicle's ID, the speed at which it was moving during each step, the distance it traveled, the time taken to reach the endpoint, and the type of vehicle.

2.3. Realistic Restrictions and Conditions in the Design

The simulation incorporates several realistic conditions and restrictions to closely replicate real-world scenarios. Vehicles are designed to adhere to traffic laws and speed limits. Additionally, the simulation considers factors such as the lane choice of vehicles and the deceleration behavior of the ambulances. However, it is important to note that certain unpredictable factors like sudden road closures, variable human behavior, or emergency incidents along the routes are not considered in the current design.

The following sections will delve into the implementation of the project, tests conducted, and the results obtained.

3. IMPLEMENTATION, TESTS and TEST DISCUSSIONS

3.1. Implementation of the Product

The implementation phase involves developing the simulation model using Python, SUMO, and TraCI. Python is a versatile, high-level programming language, making it suitable for controlling and interacting with SUMO simulations through the TraCI API.

The primary goal of the simulation is to emulate a realistic urban environment, incorporating all the complexities of urban traffic. The SUMO simulation is started with a configuration file that incorporates real-world road network data from OpenStreetMap (OSM), ensuring the simulation is grounded in reality.

In the Python script, five ambulances are created with distinct starting points, and a common endpoint is defined. Each ambulance is defined as a route, and these routes are added to the simulation. In the actual simulation, the ambulances are added to their respective routes with parameters like vehicle type, maximum departure speed, and preferred lane choice.

To capture the behavior of the ambulances in the simulation, data logging mechanisms are implemented using the xlwt module. The simulation runs for a predefined number of steps (20000 in this case). At each step, data is collected and written into the Excel file. This data includes the speed, distance covered, time, and vehicle type for each ambulance.

3.2. Tests and Results of Tests

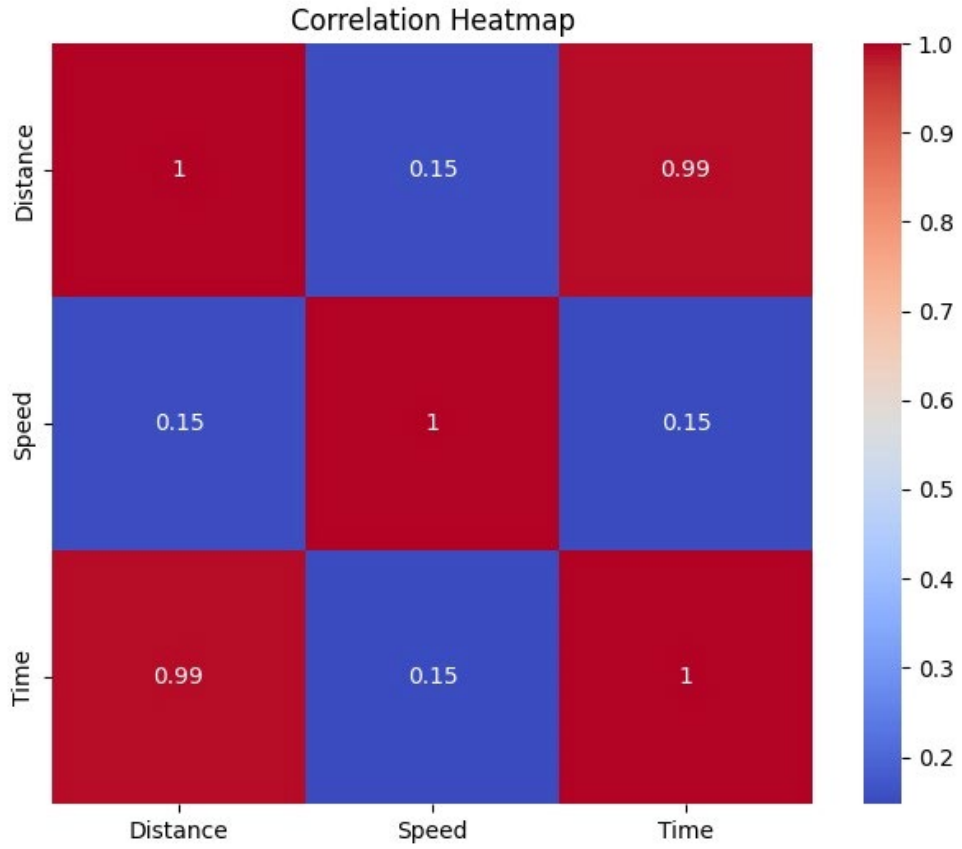


Figure 1: Correlation Heatmap

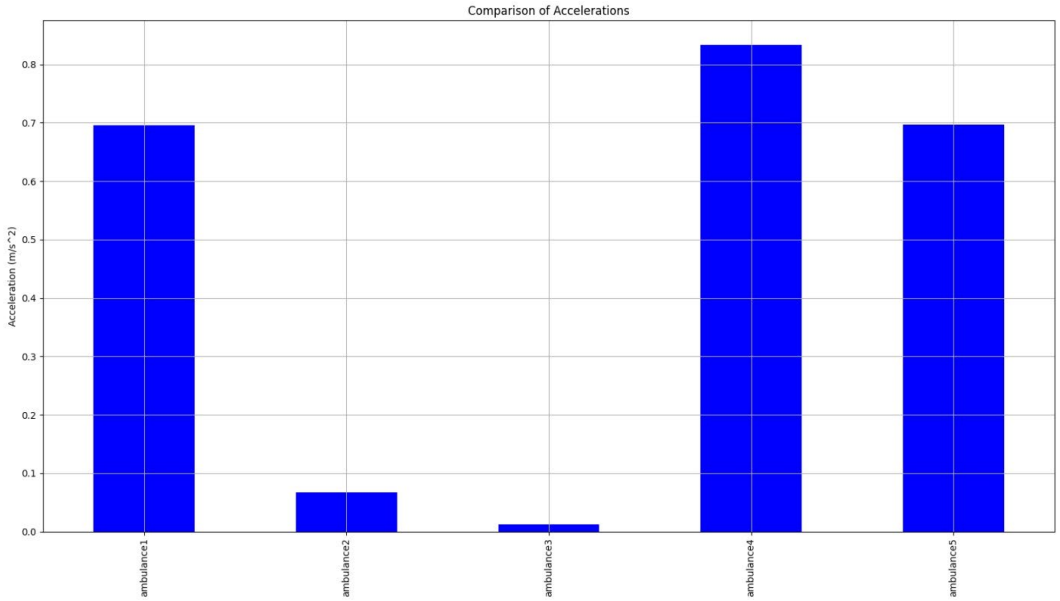


Figure 2: Comparison of Accelerations

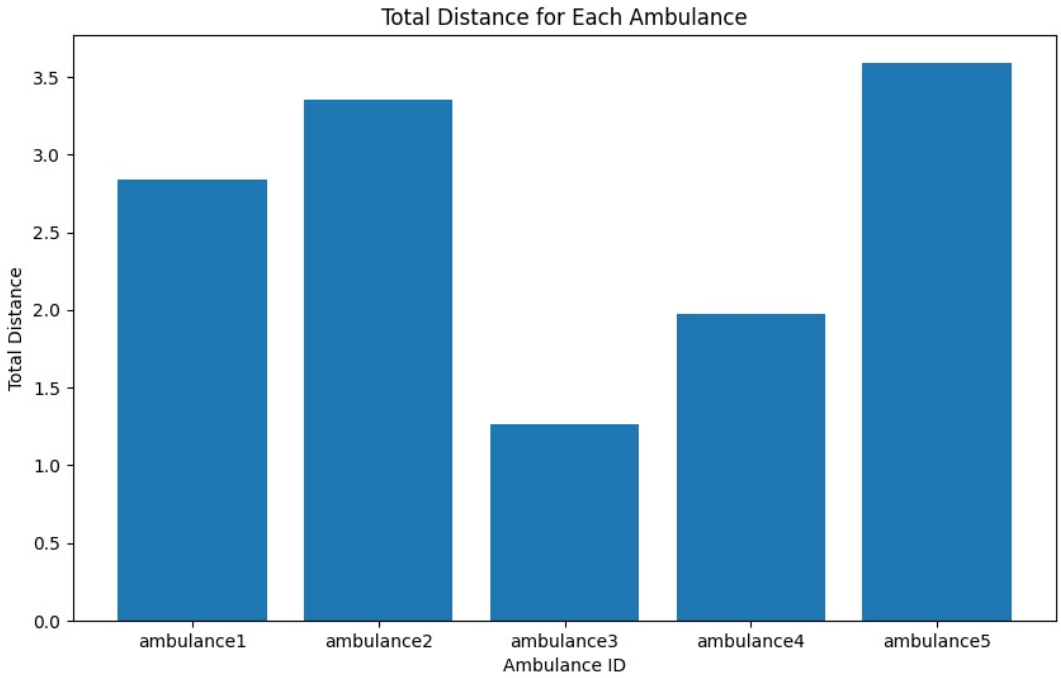


Figure 3: Total Distance for Each Ambulance

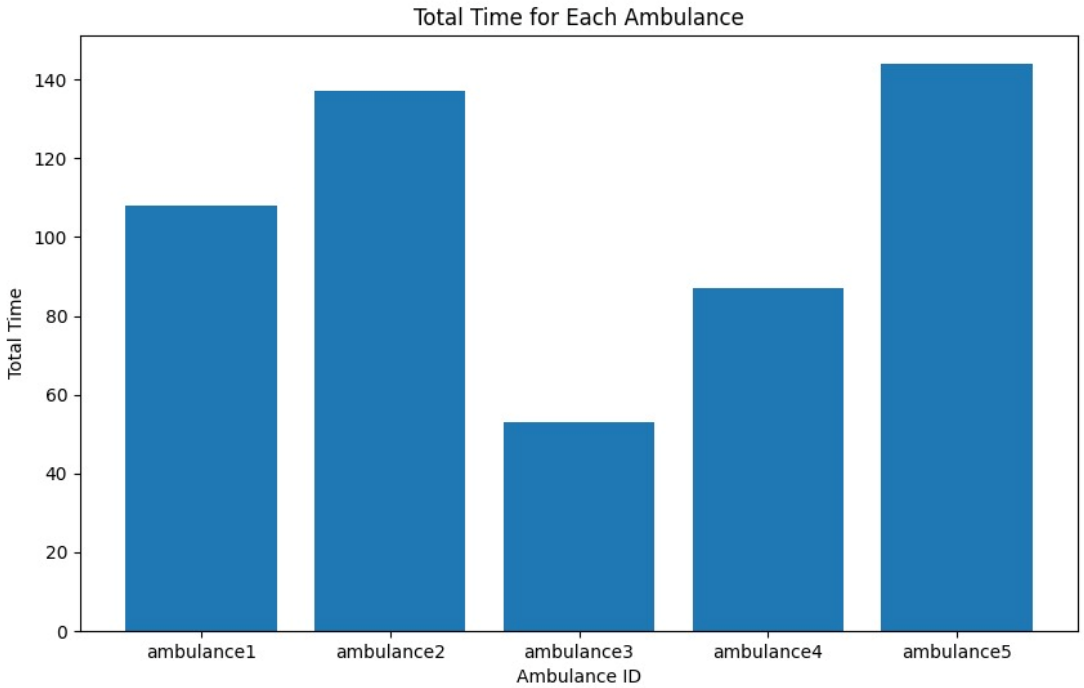


Figure 4: Total Time for Each Ambulance

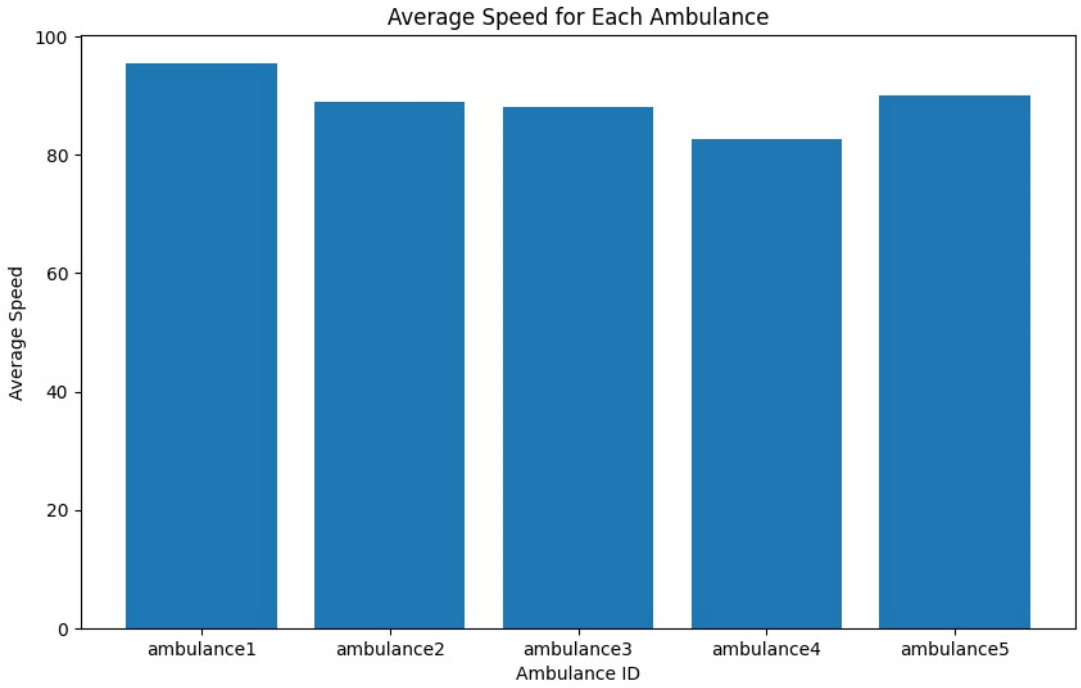


Figure 5: Average Speed for Each Ambulance

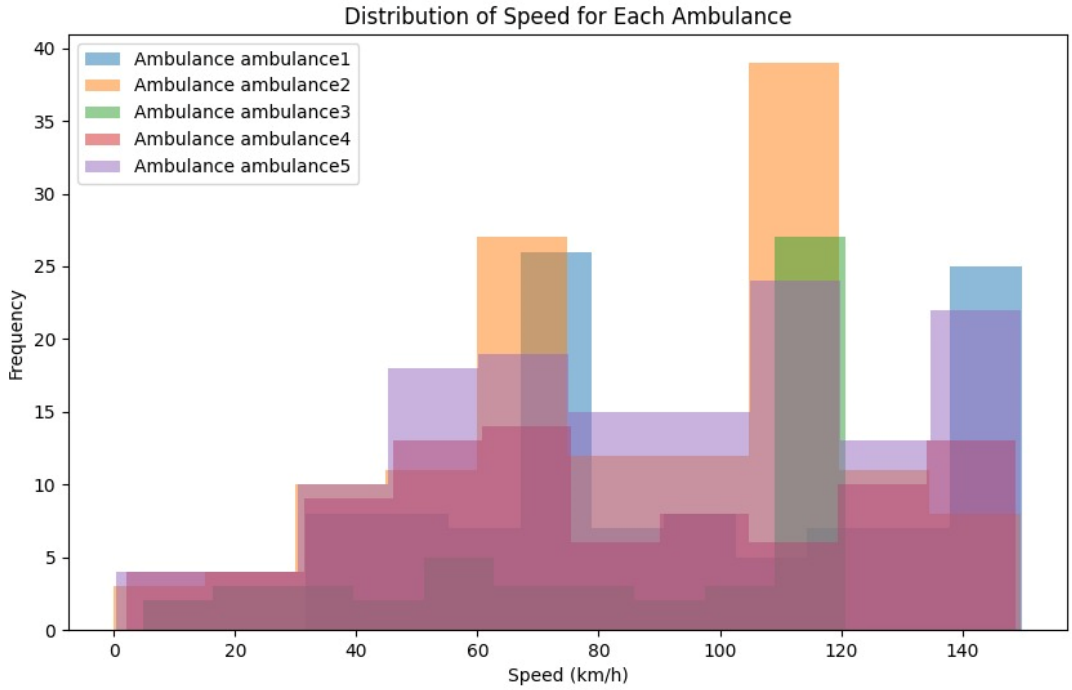


Figure 6: Distribution of Speed for Each Ambulance

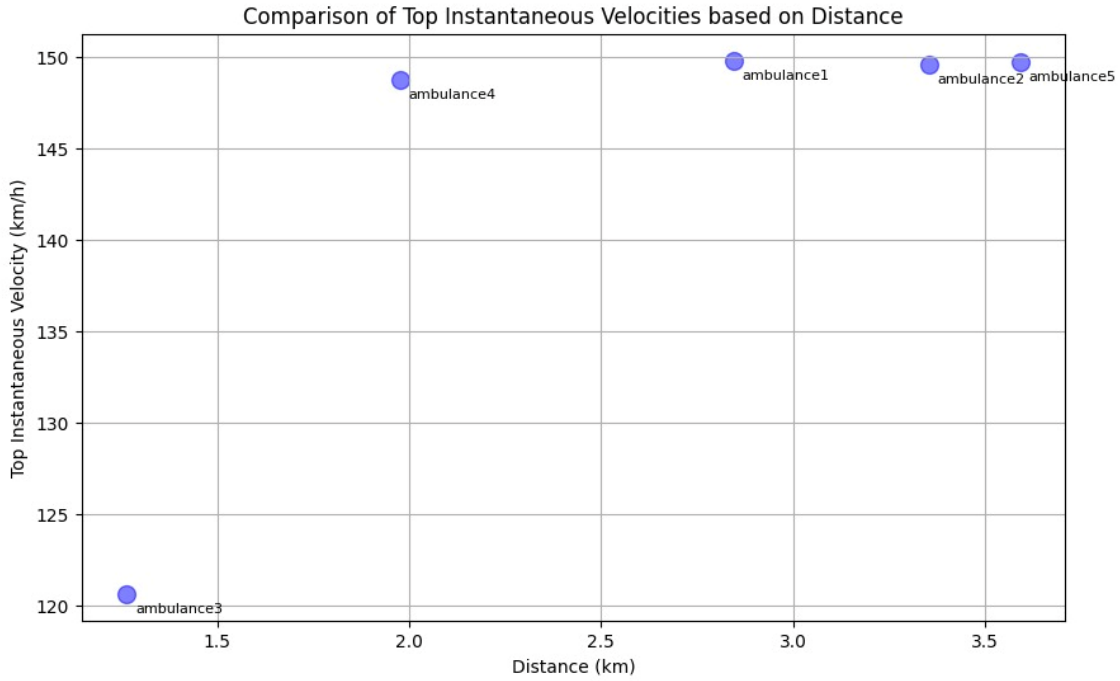


Figure 7: Comparison of Top Instantaneous Velocities based on Distance

4. CONCLUSIONS

4.1. Summary

The goal of this project was to simulate and analyze the movement of ambulances in a realistic urban traffic scenario, with the aim of understanding and improving ambulance response times. Using Python, SUMO, and TraCI, a comprehensive traffic simulation was implemented, and data about the ambulances' movement was logged into an Excel file. The project has provided a deeper understanding of the factors affecting ambulance response times and has laid the foundation for further analysis and research in this area.

4.2. Cost Analysis

The cost of this project was largely limited to the time and resources invested in learning the necessary software and languages, creating the simulation, and analyzing the data. As the software used for the project is open-source, there were no financial costs associated with software acquisition.

4.3. Benefits of the Project

The project provides several benefits, particularly for urban planners and emergency service providers. The data collected can help them understand the traffic dynamics that impact ambulance response times and can guide them in optimizing routes and strategies for quicker responses. Ultimately, these improvements could lead to enhanced patient outcomes in emergency situations.

4.4. Future Work

Future work on this project could involve enhancing the complexity and realism of the simulation. This could include integrating unpredictable events like road accidents, sudden traffic congestion, or variable human behavior. Additionally, machine learning techniques could be applied to the collected data to predict optimal routes under various traffic conditions

References

1. J. K. Hedrick, “Intelligent transportation systems: concepts and applications,” IEEE Transactions on Intelligent Transportation Systems, vol. 1, no. 1, pp. 1-9, 2000.
2. G. R. J. Qui, Z. Liu, X. Ma, and Y. Y. Li, “Emergency vehicle routing and traffic signal control based on VANET,” IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 1, pp. 365-377, 2014.
3. A. K. Verma, and V. C. M. Leung, “Real-time traffic signal control using VANET,” IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 3, pp. 1052-1066, 2014.
4. L. A. Barroso, and U. Hölzle, “The case for energy-proportional computing,” Computer, vol. 40, no. 12, pp. 33-37, 2007.
5. S. Winter, “SIMULATION OF URBAN MOBILITY - second SUMO users' conference,” in Proceedings of the second SUMO users' conference, Berlin, Germany, 2008.
6. S. R. H. Hoque, “Real-time traffic signal control and emergency vehicle prioritization using VANET,” IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 973-985, 2016.
7. M. Treiber, and A. Kesting, “Traffic flow dynamics: data, models and simulation,” Springer, 2013.
8. Y. Wang, and L. Chen, “Real-time traffic signal control for emergency vehicles using VANET,” IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 3, pp. 711-722, 2017.
9. M. A. Imran, and A. A. Qayyum, “Adaptive traffic signal control using VANET: a survey,” IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3068-3100, 2019.
10. R. T. B. Kottege, and K. P. Unnikrishnan, “Intelligent transportation systems: technologies and applications,” Springer, 2018.
11. L. R. Rios-Torres, and J. R. Paz, “Intelligent transportation systems: theory and applications,” Springer, 2018.
12. National Highway Traffic Safety Administration, “Traffic signal preemption for emergency vehicles,” US Department of Transportation, 2012.

APPENDICES

APPENDIX A: PRODUCT MANUAL

Introduction:
This manual provides detailed instructions for the installation and configuration of the traffic signal control system for emergency vehicle prioritization. It also includes information on the operation and maintenance of the system.

Hardware and Software Components:
The traffic signal control system includes the following hardware and software components:

- Traffic signal controllers
- Sensors (e.g. loop detectors, cameras)
- Communication devices (e.g. GPS, V2I, V2V)
- Simulation software (e.g. SUMO)
- Control software (e.g. traffic signal controller software)

Installation and Configuration:
The installation and configuration of the traffic signal control system involves the following steps:

- Select a suitable location for the pilot project, based on the simulation results and real-world data
- Install the necessary hardware and software components
- Configure the software system to ensure that it is working correctly and that the traffic signal control system is communicating with the other ITS systems

Operation and Maintenance:
The traffic signal control system should be regularly maintained to ensure optimal performance. This includes:

- Regularly monitoring the system for any malfunctions or errors
- Performing regular software updates
- Conducting regular tests to ensure that the system is working correctly
- Keeping accurate records of system performance and maintenance

Troubleshooting:
In case of any problems or malfunctions, the following troubleshooting steps can be performed:

- Check all connections and ensure that all components are properly connected
- Check the system logs for any error messages
- Perform a software reset or reboot the system
- Check for any updates or patches for the software
- Consult the manufacturer's website or contact customer support for further assistance

Conclusion:
This product manual provides detailed instructions for the installation, configuration, operation and maintenance of the traffic signal control system for emergency vehicle prioritization. By following the instructions outlined in this manual, users should be able to properly install and maintain the system, and troubleshoot any issues that may arise.

APPENDIX B: SOURCE CODE/EXECUTABLES/SCRIPTS

sumo_run.py

```
import traci
import xlwt

# Start SUMO with your configuration file
sumoCmd = ["sumo-gui", "-c", "osm.sumocfg"]
traci.start(sumoCmd)



# Define the end point
end_edge = "884865507#0"

# Define the starting points for the ambulances
amb_1 = "214121665#0"
amb_2 = "-287263324"
amb_3 = "-59871343#6"
amb_4 = "213980215#0"
amb_5 = "-214196661#0"

# Define the routes and vehicles
routes = []
vehicles = []


# Add ambulances 1 to 5
for i in range(1, 6):
    # Define the route ID and add it to the list
    route_id = "route{}".format(i)
    routes.append(route_id)

    # Add the route to the simulation
    traci.route.add(route_id, [eval("amb_{}".format(i)), end_edge])

    # Add a vehicle to the route with high emergency decel
    vehicle_id = "ambulance{}".format(i)
    vehicles.append(vehicle_id)
    traci.vehicle.add(vehicle_id, route_id, typeID="ambulance", depart=0,
departSpeed="max", departLane="best")

# Create a new workbook and sheet
workbook = xlwt.Workbook()
sheet = workbook.add_sheet("Data")

# Write the headers
sheet.write(0, 0, "Ambulance ID")
sheet.write(0, 1, "Speed")
sheet.write(0, 2, "Time")
sheet.write(0, 3, "Distance")
sheet.write(0, 4, "Vehicle Type")

# Initialize a row counter, starting from 1 (as 0 is for headers)
row = 1

# Loop through the simulation steps
```

```
for step in range(20000):
    # Advance the simulation
    traci.simulationStep()

    # Check if all ambulances have left the simulation
    if all(vehicle_id not in traci.vehicle.getIDList() for vehicle_id in
vehicles):
        break

    # Iterate over each ambulance
    for i, vehicle_id in enumerate(vehicles):
        # Check if the ambulance is still in the simulation
        if vehicle_id not in traci.vehicle.getIDList():
            continue

        # Get the speed, time, distance, and vehicle type
        speed = traci.vehicle.getSpeed(vehicle_id)
        time = traci.simulation.getTime()
        distance = traci.vehicle.getDistance(vehicle_id)
        vehicle_type = traci.vehicle.getTypeID(vehicle_id)

        # Write the data to the sheet
        sheet.write(row, 0, vehicle_id)
        sheet.write(row, 1, speed)
        sheet.write(row, 2, time)
        sheet.write(row, 3, distance)
        sheet.write(row, 4, vehicle_type)

        # Increment the row counter
        row += 1

# Save the workbook
workbook.save("data.xls")

# Close the simulation
traci.close()
```

visualize_data.py

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Read the data from data.xls into a DataFrame
data = pd.read_excel('data.xls')

# Convert distance from meters to kilometers
data['Distance'] = data['Distance'] / 1000

# Convert speed from meters per second to kilometers per hour
data['Speed'] = data['Speed'] * 3.6

# Group the data by Ambulance ID
grouped_data = data.groupby('Ambulance ID')

# Calculate the total distance, total time, and average speed for each
ambulance
total_distance = grouped_data['Distance'].max()
total_time = grouped_data['Time'].max()
```

```
average_speed = grouped_data['Speed'].mean()

# Create visualizations
# Bar plot for total distance
plt.figure(figsize=(10, 6))
plt.bar(total_distance.index, total_distance.values)
plt.xlabel('Ambulance ID')
plt.ylabel('Total Distance')
plt.title('Total Distance for Each Ambulance')
plt.show()

# Bar plot for total time
plt.figure(figsize=(10, 6))
plt.bar(total_time.index, total_time.values)
plt.xlabel('Ambulance ID')
plt.ylabel('Total Time')
plt.title('Total Time for Each Ambulance')
plt.show()

# Bar plot for average speed
plt.figure(figsize=(10, 6))
plt.bar(average_speed.index, average_speed.values)
plt.xlabel('Ambulance ID')
plt.ylabel('Average Speed')
plt.title('Average Speed for Each Ambulance')
plt.show()


# Create separate histograms for each ambulance's speed distribution
plt.figure(figsize=(10, 6))
for ambulance_id, group in grouped_data:
    plt.hist(group['Speed'], bins=10, alpha=0.5, label=f'Ambulance
{ambulance_id}')

plt.xlabel('Speed (km/h)')
plt.ylabel('Frequency')
plt.title('Distribution of Speed for Each Ambulance')
plt.legend()
plt.show()


# Calculate the maximum speed and unique distances for each ambulance
max_speed = grouped_data['Speed'].max()
unique_distances = grouped_data['Distance'].max()

# Create a scatter plot to compare top instantaneous velocities based on
distance
plt.figure(figsize=(10, 6))
plt.scatter(unique_distances, max_speed, s=100, c='b', alpha=0.5)
plt.xlabel('Distance (km)')
plt.ylabel('Top Instantaneous Velocity (km/h)')
plt.title('Comparison of Top Instantaneous Velocities based on Distance')
plt.grid(True)

# Annotate the data points with ambulance IDs
for i, txt in enumerate(grouped_data.groups.keys()):
    plt.annotate(txt, (unique_distances[i], max_speed[i]), xytext=(5, -
5),
                 textcoords='offset points', ha='left', va='top',
fontsize=8)
```

```
plt.show()



# Calculate the correlation matrix
correlation_matrix = data[['Distance', 'Speed', 'Ambulance ID',
'Time']].corr()

# Create a heatmap of the correlation matrix
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', square=True)
plt.title('Correlation Heatmap')
plt.show()


# Calculate the acceleration for each ambulance
acceleration = grouped_data.apply(lambda x: (x['Speed'].diff() /
x['Time'].diff()).mean())

# Create a bar plot to compare the accelerations of ambulances
plt.figure(figsize=(10, 6))
acceleration.plot(kind='bar', color='blue')
plt.xlabel('Ambulance ID')
plt.ylabel('Acceleration (m/s^2)')
plt.title('Comparison of Accelerations')
plt.grid(True)
plt.show()
```

APPENDIX C: SIMULATION MODEL AND SCENARIO

1. **Virtual City Model:** The virtual city model is a digital representation of the pilot area, including the road network, buildings, and other infrastructure. The virtual city model is created using GIS data and satellite imagery, and is used as the basis for the traffic simulation.
2. **Network Model:** The network model is a representation of the road network, including the geometry of the roads, the traffic signals, and the traffic flow. The network model is created using data from the virtual city model, and is used to simulate the traffic flow in the pilot area.
3. **Traffic Signal Control System:** The traffic signal control system is a representation of the traffic signal control system that is being tested in the simulation. It includes the logic and algorithms used to control the traffic signals, and is integrated into the network model.
4. **Data and Parameters:** This appendix also includes the data and parameters used in the simulation, such as traffic flow data, vehicle characteristics, and emergency vehicle routes. This data is used to initialize the simulation and to validate the results of the simulation.
5. **Scenarios:** This appendix also includes the scenarios used in the simulation, such as different traffic patterns, emergency vehicle routes, and different traffic signal control strategies. These scenarios are used to test the effectiveness of the proposed optimization strategies.

APPENDIX D: SIMULATION RESULTS AND ANALYSIS

We created 5 different ambulance starting point and 1 end point in simulation then we compared them.

In Figure 1, there is no significant correlation between speed with time and distance with speed. However, there is a high positive correlation between time and distance.

In Figure 2, A comparison of the accelerations of 5 ambulances in our sample is given. and there are significant differences between them.

In Figure 3, It can be seen how far the 5 ambulances have traveled in kilometers.

In Figure 4, It can be seen how long it took for 5 ambulances to reach the selected point.

In Figure 5, After arriving at the selected point, the speeds of 5 ambulances were taken and the average speed was calculated for each ambulance.

In Figure 6, The frequency of the speed of 5 ambulances per second in the simulation was determined.

In Figure 7, The maximum speed that 5 ambulances can reach while reaching the target point has been determined.

APPENDIX E: OSM Web Wizard

The OSM Web Wizard offers one of the easiest solutions to start with SUMO. Based on a selection of an openstreetmap map excerpt, you will be able to configure a randomized traffic demand and run and visualize the scenario in the sumo-gui. This tutorial will guide you step by step from the selection of the map excerpt over defining the traffic demand through running and visualizing the scenario in the sumo-gui.

Requirements

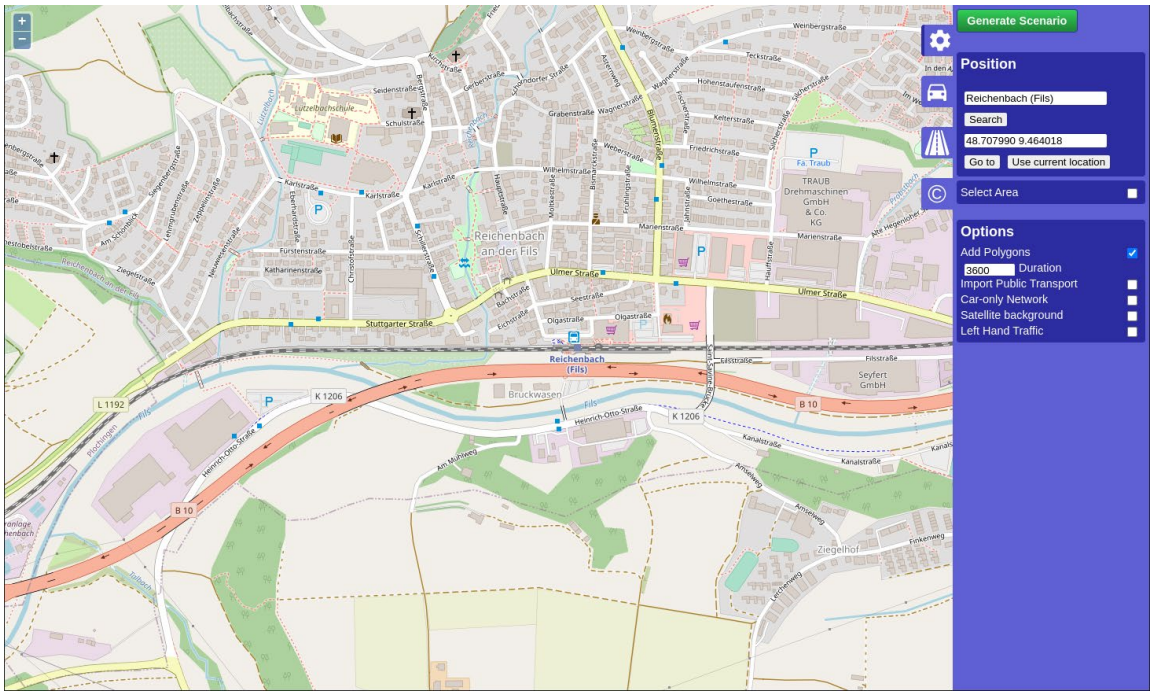
- SUMO installation
- Python (>= 2.7) installation

Getting started

The OSM Web Wizard is essentially a collection of python scripts located under the directory *tools* in your sumo installation root. You start the OSM Web wizard by invoking the following command in the *tools* directory:

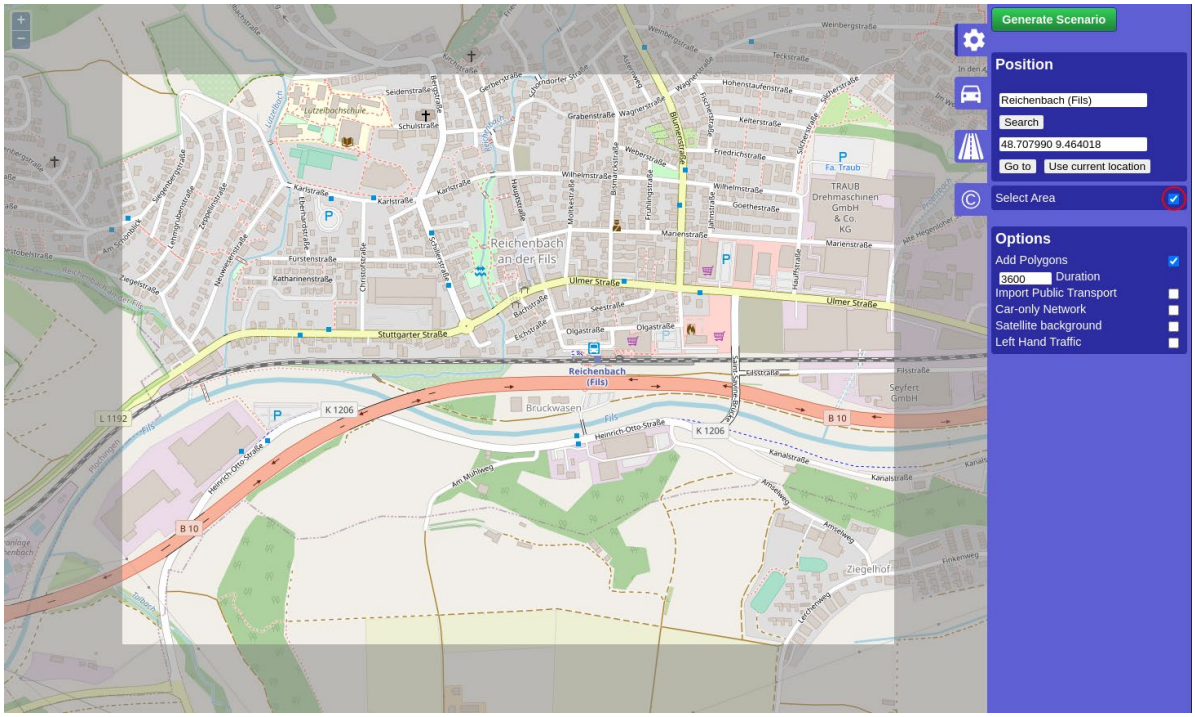
```
python osmWebWizard.py
```

Windows users may also invoke the command by clicking *All Programs -> SUMO -> OSM Web Wizard*. Once the script is running, a web browser should open showing a map excerpt of central Berlin.



You may zoom and pan to the area of your interest. Caution: if the map excerpt covers a very large area, the simulation might become slow or even unresponsive. We suggest choosing a similar zoom level as in the initial view.

In the next step, you select the actual area for which you wish to generate the simulation scenario. The area selection will be activated by clicking the checkbox *Select Area* at the blue area selection panel on the right side of the map.



You can change the size and location of this area by click and hold with the mouse pointer at the boundary between the grayed and non-grayed area. Once you are satisfied with the area selection, you can proceed to the next step.

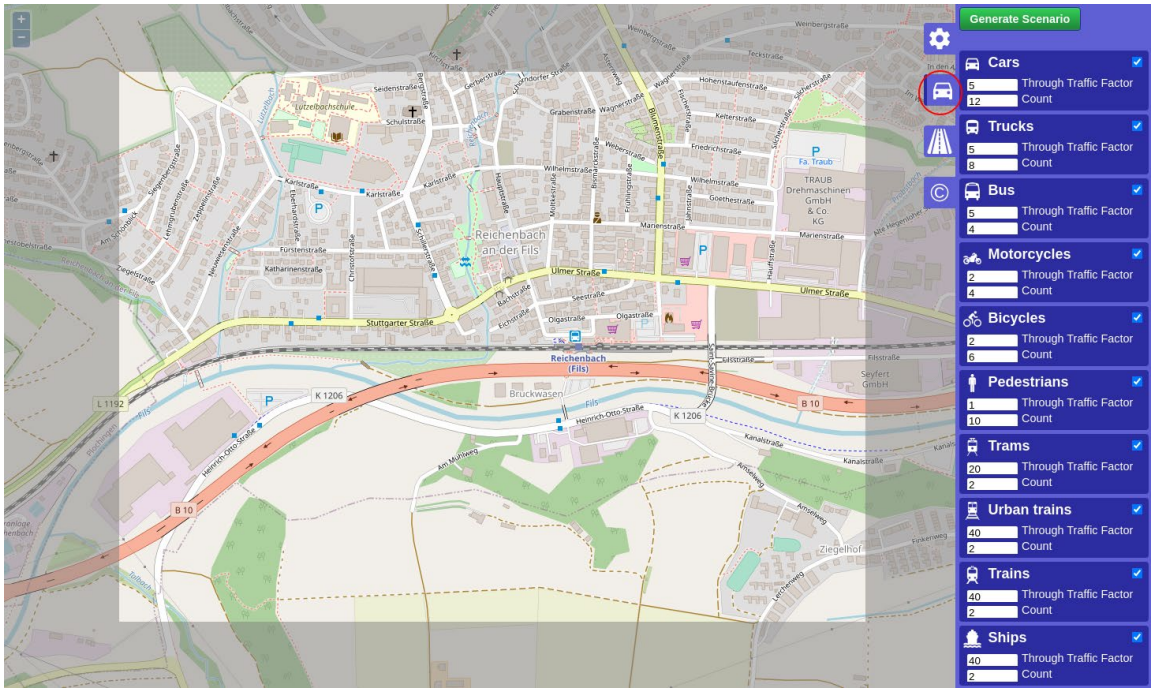
Network Generation

The infrastructure import from OSM into the SUMO simulation are affected by different Wizard options

- by default the "Add Polygon" checkbox is checked and a road traffic simulation is generated but all types of roads and rails will be imported as well (cycle paths, footpaths, railways etc)
- if the checkbox "left-hand Traffic" is enabled, the network will be built with left hand traffic rules. For most geographic regions where this is required, this feature will be enabled automatically but if it does not work, this option can be used as a remedy.
- if the checkbox "Car-only Network" is enabled, then only roads that permit passenger car traffic will be included. This can be used to reduce the network size and also helps to reduce intersection complexity
- if the checkbox "Import Public Transport" is enabled, then busStops and trainStops will be exported. Also busses, trams and trains will be generated that follow the public transport routes defined in OSM (but they will follow synthetic schedules).
- if the Demand-checkbox "Bicycles" is active, then extra bicycle lanes will be added to roads where OSM contains this information
- if the Demand-checkbox "Pedestrians" is active, then sidewalks and pedestrian crossings will be generated

Demand Generation

The demand is defined by the demand generation panel. You activate this panel by clicking on the car pictogram.



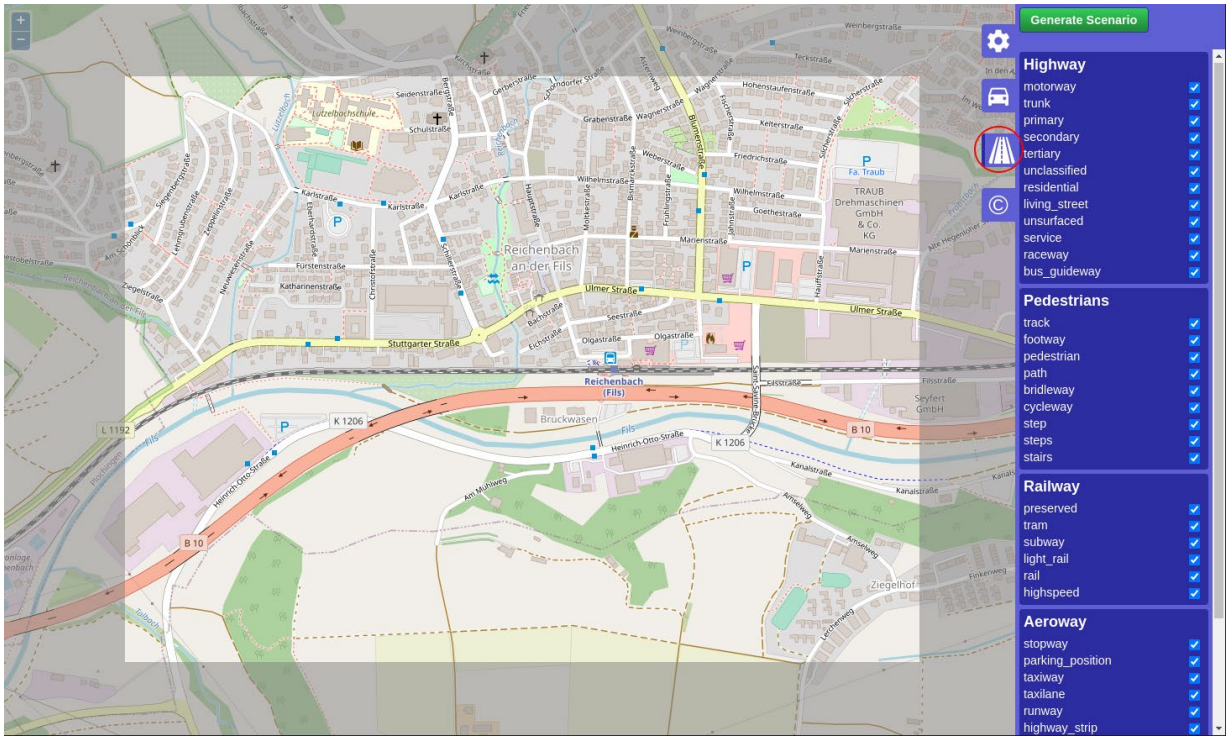
SUMO supports various modes of transport. At the demand generation panel, you can activate/deactivate the individual modes of transport by clicking the corresponding checkboxes. For each mode of transport, the OSM Web Wizard generates random demand based on a certain probability distribution, which is influenced by two parameters:

- Every time a new vehicle is generated, the OSM Web Wizard randomly chooses a departure and arrival edge for the vehicle. The *Through Traffic Factor* defines how many times it is more likely for an edge at the boundary of the simulation area being chosen, compared to an edge entirely located inside the simulation area. A big value for the *Through Traffic Factor* implies that many vehicles depart and arrive at the boundary of the simulation area, which corresponds to a scenario with a lot of through traffic.
- The *Count* parameter defines how many vehicles are generated per hour and lane-kilometer. Assuming
    - the network contains 3 edges with a combined length of 5 km
    - that each has 2 lanes which allows the current traffic mode
    - and the count value is set to 90,
    - then 5 * 2 * 90 = 900 vehicles per hour will be generated. This translates to a randomTrips parameter of p=4 which means a new vehicle is inserted every 4 seconds somewhere in the network.

The next step is generating and running the scenario.

Road-Type Selection

In the Road-Type tab of the OSM Web Wizard one can define which road types to be downloaded and rendered.

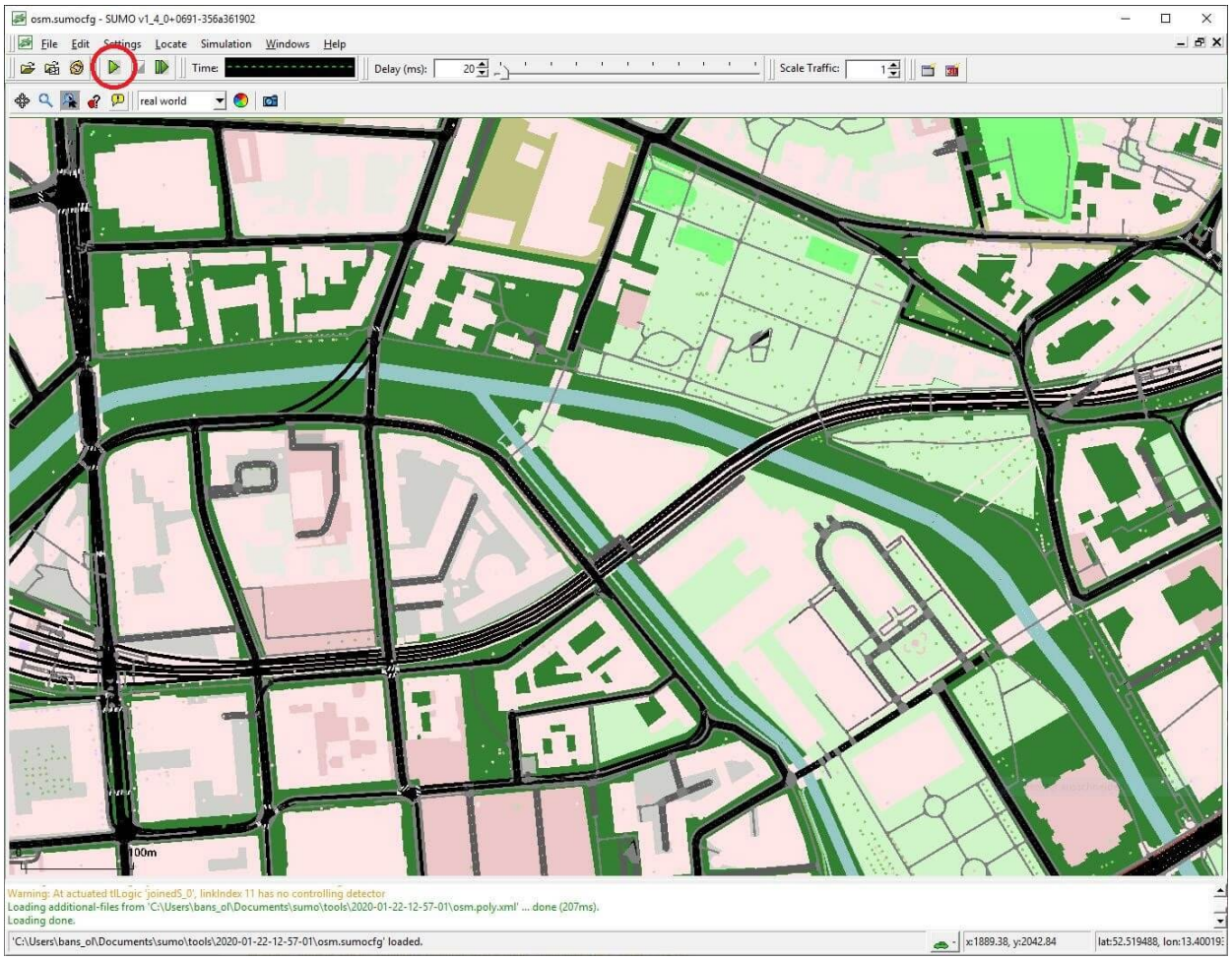


For example, one can only choose motorways, primary, secondary and tertiary to simulate major traffic. This impacts the file size of the OSM map data.

By default, all road types are checked which implies downloading and rendering all roads. Unchecking the "Add Polygon" checkbox in the Demand Generation section disables downloading and rendering non-road objects like buildings, waterways, etc. resulting in a smaller file size.

Generating and Running the scenario

The complete scenario will be generated automatically once *Generate Scenario* in the control panel has been clicked. The scenario generation takes a couple of seconds or minutes (depending, among other factors, on the size of the scenario). Once the scenario generation process has finished, the sumo-gui starts and the simulation can be started by pressing the *Play* button.



Where to go from here

The OSM Web Wizard stores the entire simulation scenario sumo config and intermediate files in a local directory with a name in the format of *yyyy-mm-dd-hh-mm-ss*. If your SUMO installation location is writeable, the data will be stored inside the *tools* directory. Otherwise, a new directory *~/SUMO/yyyy-mm-dd-hh-mm-ss* will be created to host the data. The contents of the directory look like this:

```
   ➜  2016-10-17-14-54-30 ls
   build.bat                  osm.net.xml                osm.rail.rou.alt.x
ml       osm.tram.rou.alt.xml
   osm.bicycle.rou.alt.xml    osm.netccfg                osm.rail.rou.xml
         osm.tram.rou.xml
   osm.bicycle.rou.xml        osm.passenger.rou.alt.xml  osm.rail.trips.xml
         osm.tram.trips.xml
   osm.bicycle.trips.xml      osm.passenger.rou.xml      osm.rail_urban.rou
.alt.xml osm.truck.rou.alt.xml
   osm.bus.rou.alt.xml        osm.passenger.trips.xml    osm.rail_urban.rou
.xml     osm.truck.rou.xml
   osm.bus.rou.xml            osm.pedestrian.rou.alt.xml osm.rail_urban.tri
ps.xml   osm.truck.trips.xml
   osm.bus.trips.xml          osm.pedestrian.rou.xml     osm.ship.rou.alt.x
ml       osm.view.xml
   osm.motorcycle.rou.alt.xml osm.pedestrian.trips.xml   osm.ship.rou.xml
         osm_bbox.osm.xml
```

```
osm.motorcycle.rou.xml     osm.poly.xml               osm.ship.trips.xml
      run.bat
osm.motorcycle.trips.xml   osm.polycfg                osm.sumocfg
```

You may now edit those files and re-run the simulation. To learn more about the SUMO scenario files, please consult the other Tutorials as well.

Please note that depending on your SUMO version some of these files may have an additional *.gz* suffix. The SUMO tools (including sumo-gui and netedit) will still be able to process them but to view or modify them in a text editor you will need to unzip them (for instance with gunzip or 7z).