



Scalable Data Analytics,
Scalable Algorithms, Software Frameworks
and Visualization ICT-2013 4.2.a

Project **FP6-619435/SPEEDD**
Deliverable **D8.4**
Distribution **Public**



<http://speedd-project.eu>

Final Version of Micro-Simulator

Authors

CNRS Rohit SINGHAL, Anton ANDREEV, A. KIBANGOU

January 2016

Project

Project Ref. no	FP7-619435
Project acronym	SPEEDD
Project full title	Scalable Proactive Event-Driven Decision Making
Project site	http://speedd-project.eu/
Project start	February 2014
Project duration	3 years
EC Project Officer	Alina Lupu

Deliverable

Deliverable type	Report
Distribution level	Public
Deliverable Number	D8.4
Deliverable Title	Final Version of Micro-Simulator
Contractual date of delivery	Jan 2016
Actual date of delivery	Jan 2016
Relevant Task(s)	WP8 Task T8.3
Partner Responsible	CNRS
Other contributors	
Number of Pages	22
Status & Version	Final Version
Keywords	Micro-simulator, Grenoble city, APIs, Kafka

Executive Summary.....	4
Project and work package goals	4
Work presented in the deliverable	4
How SPEEDD benefits from the work	4
1. Introduction	5
1.1 History of the document.....	5
1.2 Purpose and scope of the document.....	5
Context.....	5
Deliverable purpose	5
1.3 Relationship with other documents.....	5
2. Micro-Simulator	6
2.1 AIMSUN simulator.....	6
2.1.1 Input required by AIMSUN.....	6
2.2 Simulator for Grenoble city.....	8
3. AIMSUN APIs	12
3.1 Aimsun API Description.....	12
3.1.1 APIs using Matlab Interface	13
3.1.2 APIs using Kafka Interface.....	13
3.1.3 Aimsun Script	14
3.2 Aimsun Interface with SPEEDD Prototype	14
3.2.1 Input required to control traffic light phases	15
3.2.2 Input required to control Section speed limit.....	15
3.2.3 Input JSON format to control traffic light and speed limit	15
3.2.4 Output from Aimsun APIs	15
3.3 Output from AIMSUN Micro-simulator	15
4. Conclusion.....	22

Executive Summary

Project and work package goals

SPEEDD will develop prototypes for proactive event-driven decision making and robust forecasting. The motivation for proactive computing stems from social and economic factors, and is based on the fact that prevention is often more effective than cure. SPEEDD will contribute to the state of the art by:

1. Developing novel methods for real time event recognition and forecasting;
2. Providing innovative techniques for proactive event-driven decision-making;
3. Developing techniques for real-time visualization and explanation of large quantities of data.

WP8–Proactive Traffic Management Use Case–aims to forecast traffic congestions before they happen and to make decisions in order to attenuate them. Task 8.4 concerns the development of a microscopic simulator for the selected use-case, involving the full Grenoble city.

Work presented in the deliverable

This document presents the final version of a micro-simulator for Grenoble city, which includes a large portion of Grenoble selected for being crucial for traffic and often affected by congestions.

The simulator is based on commercial software Aimsun. This deliverable describes how the Grenoble simulator has been produced using Aimsun tools.

How SPEEDD benefits from the work

The reason why this simulator is essential for SPEEDD is twofold. First, it will provide synthetic data for the urban area, for which Sensors have not been deployed. Second, it will enable testing proactive decision-making by allowing to close the loop and see in real time the effect of decisions. Clearly such tests need to be performed in simulation, so as to avoid perturbing real traffic for testing purposes.

1. Introduction

1.1 History of the document

Version	Date	Author	Change Description
0.1	06/01/2016	Rohit Singhal(CNRS)	Set up the document
0.2	11/01/2016	Rohit Singhal(CNRS)	Document Formatting
0.3	14/01/2016	Anton Andreev (CNRS)	General improvements
0.4	14/01/2016	Rohit Singhal (CNRS)	Updated Version
0.5	15/01/2016	A. Kibangou (CNRS)	Minor Updates
1.0	26/01/2016	Rohit Singhal (CNRS)	Final version

1.2 Purpose and scope of the document

Context

Transportation and traffic congestion are crucial aspects of human civilization, especially starting from the second half of the last century when the latter became predominant due to the rapid increase in the number of vehicles. Traffic congestion results in excess delays, reduced safety, and increased environmental pollution. Traffic analysis and forecasting, necessary for a good management of transportation systems, require the analysis of massive data streams storming from various sensors, and this brings further difficult tasks (mainly about real-time processing of big quantities, geographically distributed and noisy data).

As a major tool for research on traffic estimation and control, CNRS is developing a simulator of Grenoble traffic. This is based on AIMSUN simulator for traffic systems. Version 1 of Grenoble simulator concerns two portions of the area: Grenoble South Ring, a 12km highway and a portion of Grenoble downtown which is particularly critical for traffic congestion, Version 1 was submitted last year (see Deliverable D8.1 and D8.2) and now this deliverable is more focused on the Aimsun model for the entire Grenoble city, which is considered as final version of the simulator.

Deliverable purpose

This document gives detailed information about the final version of the micro-simulator for Grenoble traffic.

1.3 Relationship with other documents

D8.4 is delivered at month 24th of the project, and describes the final version of the simulator. This simulator is based on the scenario described in D8.1 and D8.2.

2. Micro-Simulator

2.1 AIMSUN simulator

The CNRS simulator of Grenoble town is based on AIMSUN (<http://www.aimsun.com/wp/>). CNRS has a legal license to use AIMSUN micro-simulator with professional edition, which covers all the functionalities of this software. Now to develop model for Grenoble city, CNRS has bought the license to use Grenoble city map from HERE Company.

AIMSUN is a widely used commercial transport modeling software, developed and marketed by TSS-Transport Simulation Systems based in Barcelona, Spain. Microscopic simulators are the components of AIMSUN, which allow for dynamic simulations. They can deal with different traffic networks: urban networks, freeways, highways, ring roads, arterials and any combination.

The microscopic model simulates the movement of individual vehicles, based on statistical laws from car-following and lane-changing theories. Typically, vehicles enter a transportation network using a statistical distribution of arrivals. The microscopic model incorporates sub-models for acceleration, speed adaptation, lane-changing etc., to describe how vehicles move and interact with each other and with the infrastructure. This simulation is microscopic in the sense that it is at the individual car level (in analogy with simulations in physics at the molecular level), as opposed to macroscopic models which use a fluid-dynamic description of the traffic flow as a whole, with partial differential equations for flows and densities.

Aimsun provides the tools to carry out traffic operations assessment of any scale and complexity. The applications are many, some of the most common are:

- Impact analysis of infrastructure design such as highway corridor improvement/construction
- Evaluation of travel demand management (TDM) strategies
- Signal control plan optimization (Aimsun-TRANSYT link) and adaptive control evaluation
- Evaluation of Variable Speed policies and other Intelligent Transportation Systems (ITS)

Even on a laptop, the Aimsun micro-simulator can run a model of the whole town of Singapore with 10,580 intersections and 4,483 km of lanes 2-3 times faster than real time [<http://www.aimsun.com/wp/>]. High speed simulation makes it possible to:

- Simulate larger areas without leaving out problem zones
- Test a greater number of scenarios in the time available allowing for broader ‘what if’ analysis for a variety of traffic conditions
- Deal with time-critical situations

2.1.1 Inputs required by AIMSUN

The input data required by AIMSUN Dynamic simulators is a simulation scenario and a set of simulation parameters that define the experiment. This scenario is composed of three types of data:

a. *Network description*

The network is a collection of links connected to each other by nodes (intersections) and may have different traffic features. In order to build the network in AIMSUN, the user needs the following data:

1. Map of the area
2. Details of the number of lanes for each section, reserved and side lanes.
3. Possible turning movement for every junction, including details about the lanes from which each turning is allowed and solid lines marked on the road surface.
4. Detectors

b. *Traffic control*

There are several types of traffic control taken into account by AIMSUN: signals, give-way signs and ramp meters. Traffic signals and give-way signs are used for junction nodes, while ramp meters is for sections that end at junction node.

c. *Traffic demand data*

Traffic demand data in AIMSUN can be defined in two different ways, by the traffic flows at the sections or by an origin/destination (O/D) matrix. Several O/D matrices or Traffic states will be grouped into a traffic demand. The following data must be provided for those different types of traffic demand data:

1. Traffic flows
 - a. Vehicle type and the attributes
 - b. Vehicle classes
 - c. Flow in the input section for each vehicle
 - d. Turning proportion at all section for each vehicle type
2. O/D Matrix
 - a. Centroids definition: traffic sources and sinks
 - b. Vehicle type and attributes
 - c. Vehicle classes
 - d. Number of trips going from every origin centroid to any destination one.

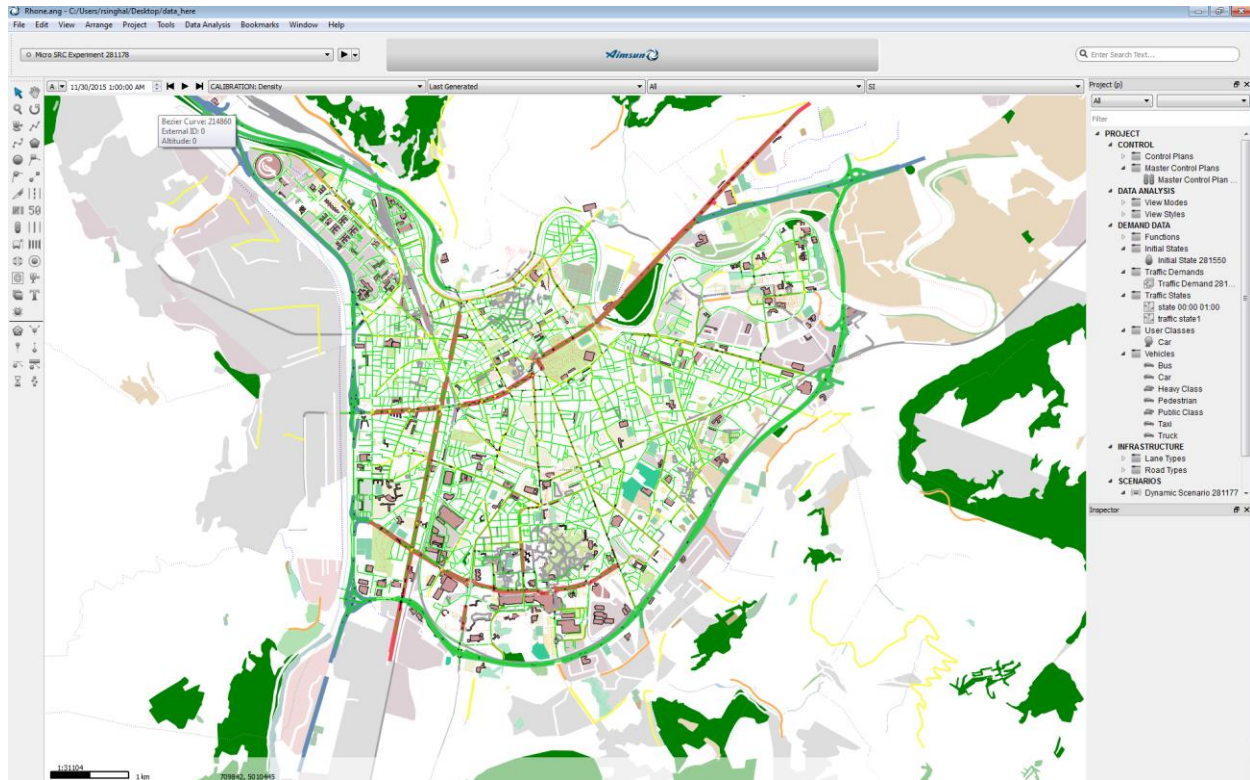


Figure 1: AIMSUN main screen

Figure 1 shows the AIMSUN main screen. In this screen, we can see the different tools available in the AIMSUN simulator for developers (left side of the screen).

2.2 Simulator for Grenoble city

This is the final part of AIMSUN simulator developed by CNRS, which is the full model of Grenoble city with South ring highway included. We also included many small roads and traffic from major roads connecting to highways. This was done in order to emulate big congestions in specific peak hours (i.e. mornings and evenings).

The map for Grenoble city is shown in Figure 2 (based on Google map). The map in the red box is indicated as Grenoble city map. The orange line in the map indicates the south ring highway, where the real sensors are placed. The corresponding AIMSUN map is shown in Figure 3. In this figure, all the lines in green color indicate the major highways.

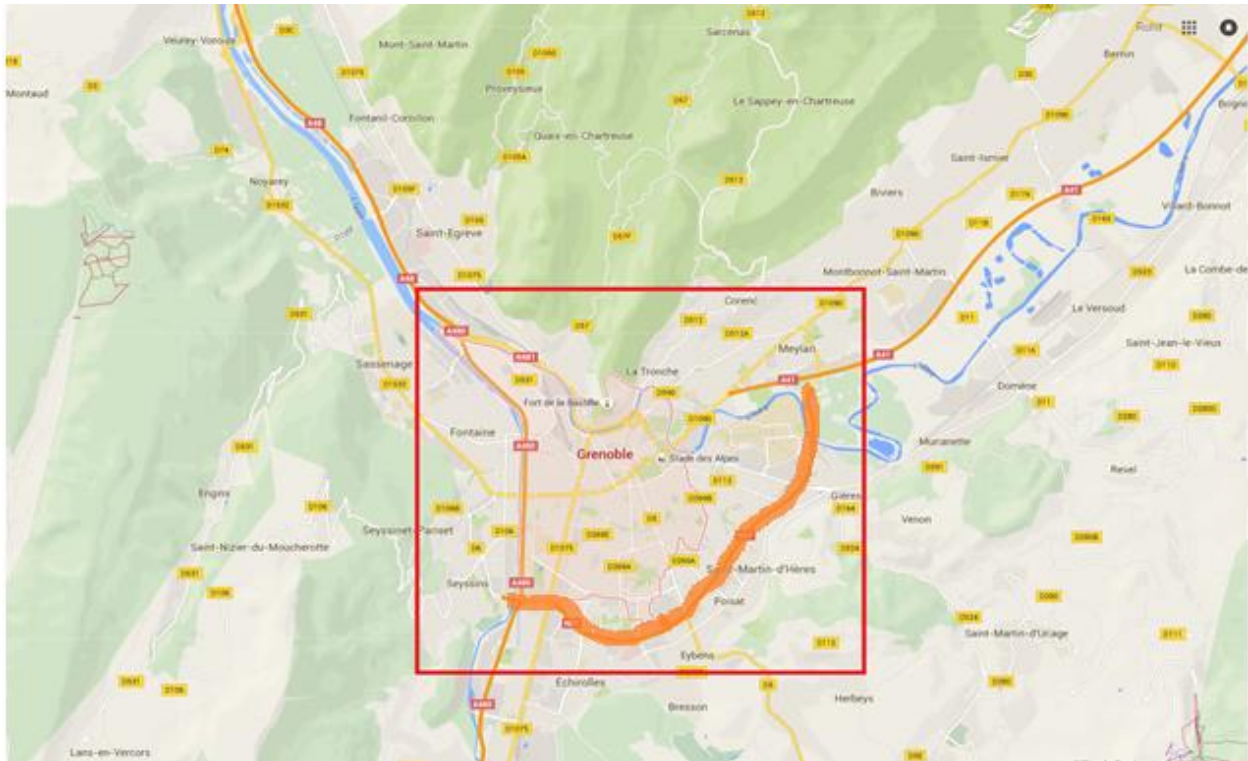


Figure 2: Google map of the Grenoble city



Figure 3: AIMSUM map of the Grenoble road traffic network

The images below (Figures 4 to 6) show details of the simulator of the full Grenoble city, with screenshots of the AIMSUN visualization of the traffic flow. As we can see in the images, there are many intersection points, where traffic signals are set up with green and red small boxes on the road at the starting of each intersections. The vehicles are moving and represented in blue color with commercial and public vehicles both taken into account. We have also considered the parking areas, where higher flow of vehicles is present. We have also considered: schools, small roads near private residences and the private residences themselves.

In this final version of Grenoble city, we are using HERE (previously Nokia Maps) map to setup the model in Aimsun. This will help us to make our model more efficient as compared to our previous version, where Open Street Maps (OSM) are used. With OSM we needed to apply many manual modifications at intersections and certain roads of the city, while in the HERE map, the quality of the maps is very rich. So not many manual modifications were needed and the number of errors in the map itself is very low. It can be compared with the real life roads and streets.

Now, we are considering the full Grenoble map, the complexity of the model increases, which means: (i) there will be more intersections and traffic lights that need to be maintained and (ii) the calibration needs to be done in more accurate and systematic way. (iii) We are using split ratio in order to calibrate the simulator of Grenoble city. Based on our research we have developed a script to automatically set the distribution of split ratio at all the intersection present in the model. During the calibration, we also need to keep in mind about the different traffic levels during peak hours and non-peak hours and this is also one of the most difficult tasks for calibration.

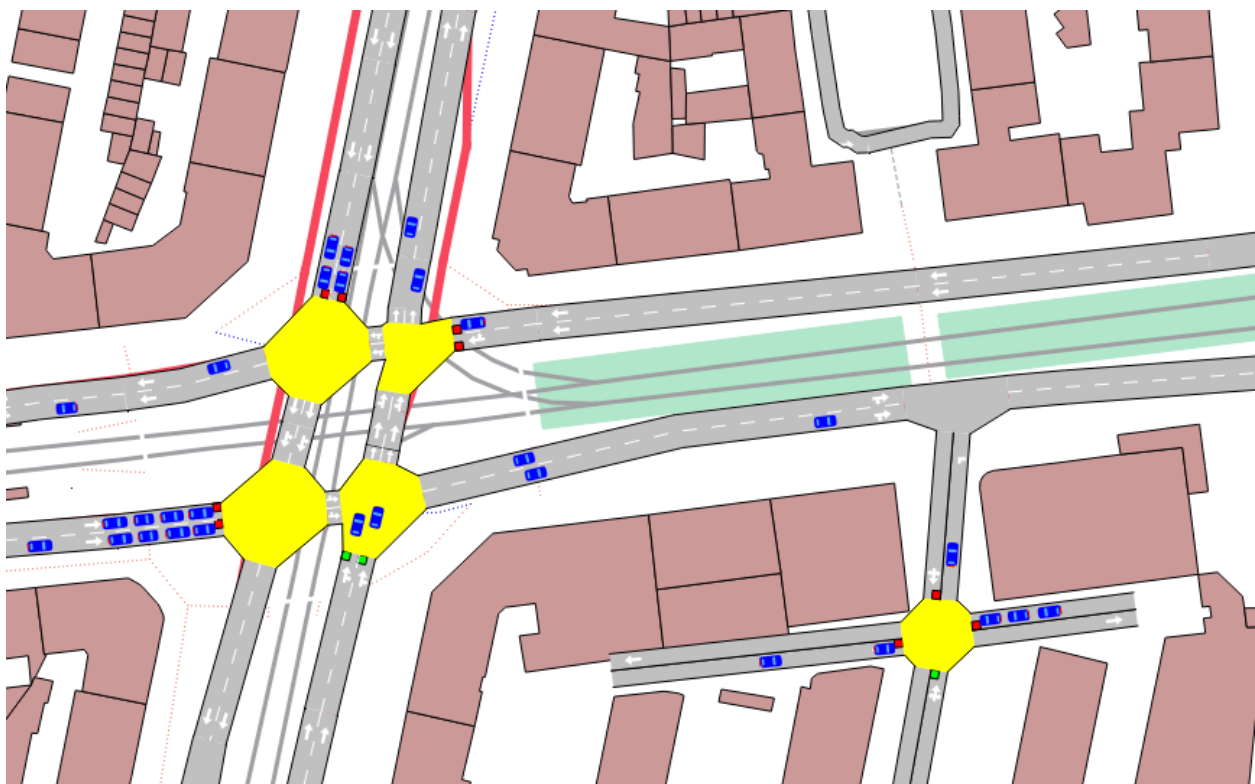


Figure 4: Screenshot of traffic simulation at Joseph Vallier Bd.

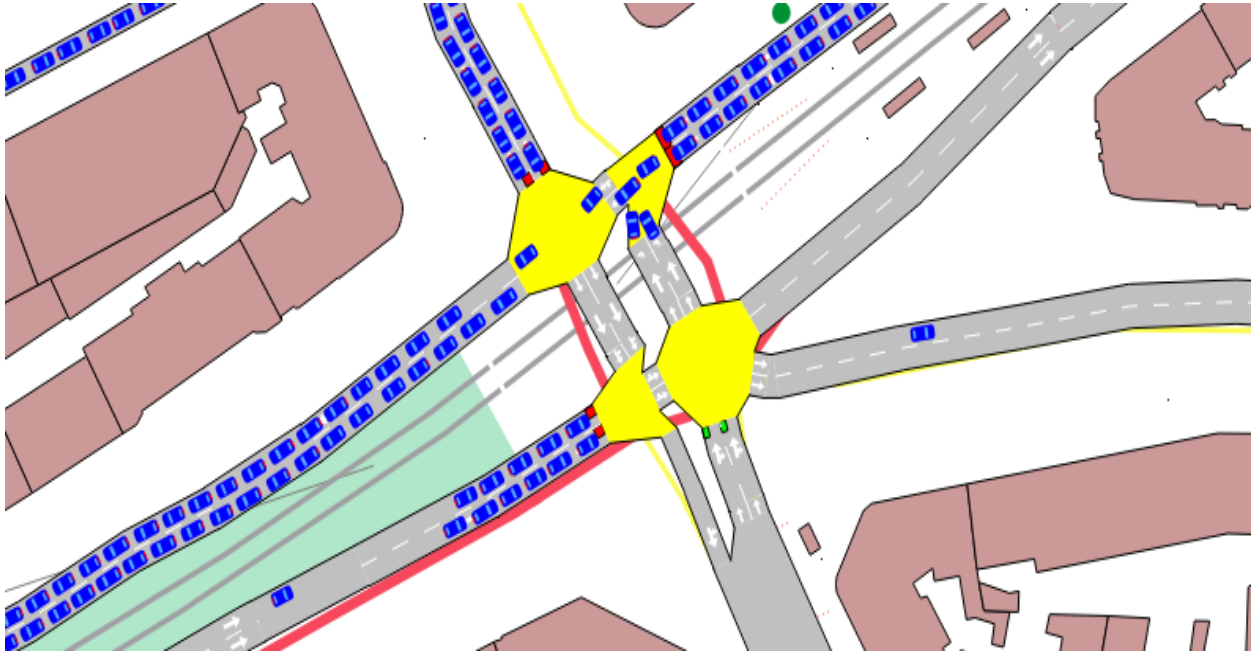


Figure 5: Traffic simulation at Marechal Foch Street during peak hours.

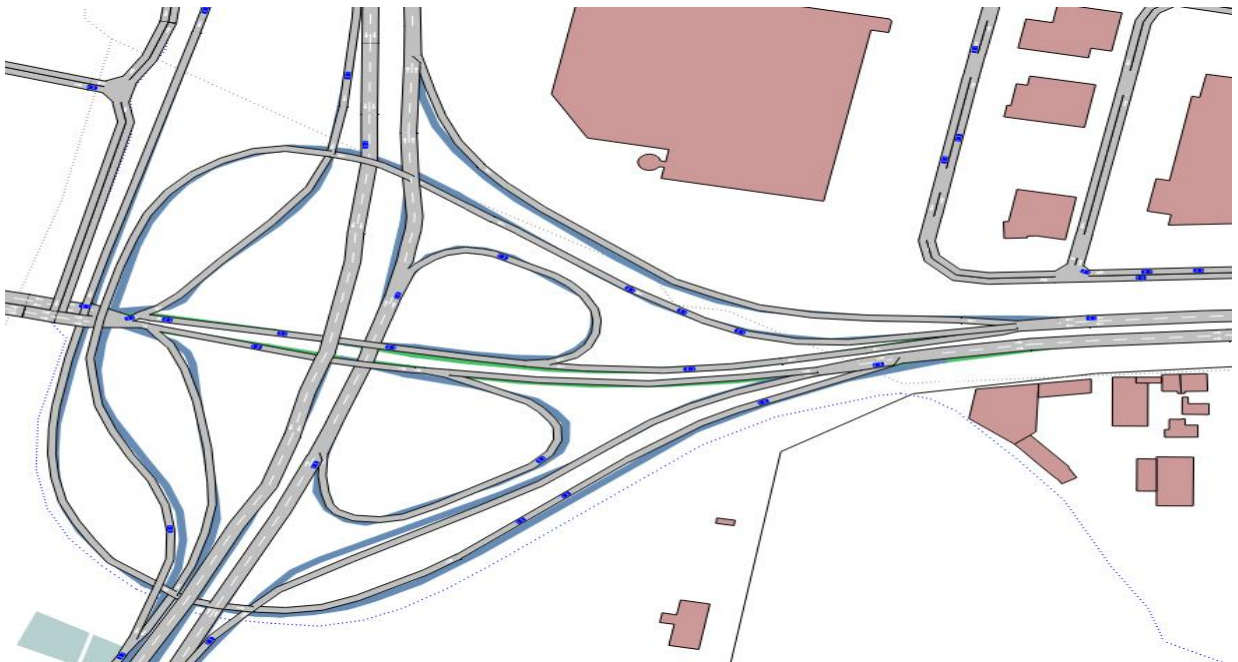


Figure 6: Traffic simulation at Le Rondeau, the busiest and complex area of the Grenoble south ring highway, during free flow hours.

3. AIMSUN APIs

3.1 Aimsun API Description

In order to control the different parameters in Aimsun Micro-simulator, CNRS has developed variety of APIs using different technologies and applications.

There are two types of APIs and one Aimsun plugin that have been developed for controlling and adding functionalities in Aimsun:

1. **Aimsun Interfaces using Matlab:**

It is a bridge between AIMSUN and Matlab which allows traffic and parameters data to be read from AIMSUN and control to be applied back into AIMSUN. It is a plugin dll developed in C++ but with wrapper .NET classes. It allows for developers to attach to specific hooks in the AIMSUN scenario execution workflow.

2. **Aimsun Interfaces using Apache Kafka:**

Kafka is a publish-subscribe messaging system. Same concept has been used for Kafka where instead of Matlab, we are using Kafka to send messages or events to Aimsun and then Aimsun sends back events to a Kafka client. Detailed description about kafka will be found in the following sections of this document. The programming language used to write this code is Python.

3. **Aimsun Script:**

This script has been developed in Aimsun User interface to add functionalities in the simulator. Language used to write this code is Python.

- Below are the list of APIs (Script) developed using Matlab Interface:

1. **API-M-SSL** : Section Speed Limit
2. **API-M-CSSL** : Compliance to Speed limit in the sections(value between 0 to 1)
3. **API-M-TLO** : Traffic Light offsets (Reaction time of vehicle)
4. **API-M-MET** : Upstream varying metering(for varying demand)
5. **API-M-VD** : Upstream varying demand
6. **API-M-MSV** : Maximum allowed speed for each vehicle
7. **API-M-GPS** : GPS traces for each vehicle

Besides above APIs, there are several scripts in Matlab to measure the Energy consumption of a vehicle, total time traveled by a vehicle in Aimsun and to measure the length of queue created during congestion or at traffic signal.

- List of APIs developed using Kafka Interface:

1. **API-K-SSL** : Section Speed Limit
 2. **API-K-JTL** : Junction Traffic Light
 3. **API-K-DATA** : Static data from a Detector that provides data every 15 sec for Speed, Count and Occupancy
 4. **API-K-ACI** : Create accident in random location during simulation.
- List of Aimsun Script:
 1. **API-A-INP** : Script to give simulation input data (State Input)

3.1.1 APIs using Matlab Interface

This APIs has been developed to control different parameters and variables of the Aimsun simulator remotely using Matlab.

This APIs has been developed in C++ Language and Matlab. Several types of scripts have been developed for different task like changing the speed of the section (road), controlling traffic lights, calculate length of queue, etc.

Detailed description about these scripts is as follows:

- **API-M-SSL**: This API has been used to change the maximum speed limit of the section¹ in Aimsun (Section is piece of road with set of lanes).
- **API-M-CSSL**: This API is used to set the percentage of vehicles that should follow the speed limit of a section.
- **API-M-TLO**: used to give time duration in seconds for acceleration of a vehicle waiting at traffic light.
- **API-M-MET**: used to set green light duration of the metering.
- **API-M-VD**: used to change the input matrix of the simulation.
- **API-M-MSV**: used to set the maximum speed for each vehicle during simulation.
- **API-M-GPS**: used to get GPS co-ordinates(X and Y) for each vehicle during simulation.

3.1.2 APIs using Kafka Interface

The list of APIs using KAFKA is listed below with detailed explanation:

- **API-K-SSL**: used to set the maximum speed of the section - same as Matlab interface.
- **API-K-JTL**: used to control the traffic light of a junction. It serves to set different parameters of traffic lights in Aimsun.

¹ A section is a piece of road with set of lanes.

- **API-K-DATA:** used to extract the output data from Aimsun, i.e. data produced by Detectors placed in the network.
- **API-K-ACI:** used to create incident or accident at random location in the network.

An API to communicate between AIMSUN and SPEEDD components has been developed in order to send data from AIMSUN to the SPEEDD architecture. The KAFKA message streaming systems is the technology we have chosen to transfer data between SPEEDD components and AIMSUN. Using this API, we are able to do two types of control actions in AIMSUN:

1. Control the traffic light phases
2. Control the speed limit of each section

There are two types of components present in KAFKA in order to make it work:

1. Consumer: The component which consumes data, in our case AIMSUN, behaves as a consumer when the control functions are used.
2. Producer: The component which produces data. The producer is again AIMSUN which sends traffic data using the SPEEDD protocol.

3.1.3 Aimsun Script

This **API-A-INP** script has been developed in the Aimsun Interface using Python language. Basically this script allows us to give the turning percentage for each node for the different possible turning direction in Aimsun, which is used to simulate data i.e. Input state data.

3.2 Aimsun Interface with SPEEDD Prototype

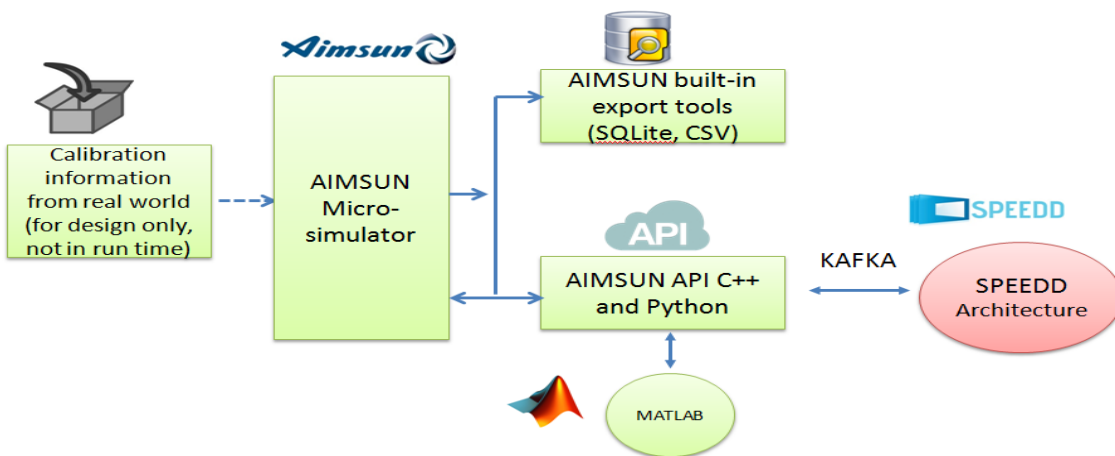


Figure 7: Interfacing AIMSUN with SPEEDD prototype

The above diagram shows the complete interface between Aimsun and the SPEEDD Architecture. There are several components present in this diagram to make them communicate with each other. Aimsun can be start remotely by a single command, but it cannot be stopped remotely during simulation.

3.2.1 Inputs required to control traffic light phases

The parameters required by Aimsun interface to control traffic light phases are as follows:

1. Junction ID (Intersection ID)
2. Phases ID
3. New Phase time (Seconds)

3.2.2 Inputs required to control Section speed limit

Parameters required by Aimsun interface to control section speed limits are as follows:

1. Section ID
2. New Speed limit (Km/Hr)

3.2.3 JSON format to control traffic light and speed limit

The JSON format that Aimsun API uses to control the elements is the following:

```
"junction_id","phase_id","phase_time","section_id","speed_limit"
```

3.2.4 Output from Aimsun APIs

The output format for the data received from Aimsun through Aimsun API will be following:

```
Simulation_time,detector_id,vehicle_speed,vehicle_count_car,vehicle_count_truck,density_car,density_truck,occupancy
```

Note: All the data sent by Aimsun will be the aggregation of every 15 seconds.

1. The simulation time will be in this format: YYYY-MM-DD HH:MM:SS.
2. Detector ID means the ID of the sensors placed on the sections in Aimsun.
3. Vehicle_Speed will be in Km/hr
4. Vehicle_Count_car will be the Number of cars passing through the sensor at each 15 seconds.
5. Vehicle_Count_truck will be the Number of trucks passing through the sensor at each 15 seconds.
6. Density_car will be the density of car in each 15 seconds.
7. Density_truck will be the density of truck in each 15 seconds.
8. Occupancy will be the occupancy for both cars and trucks.

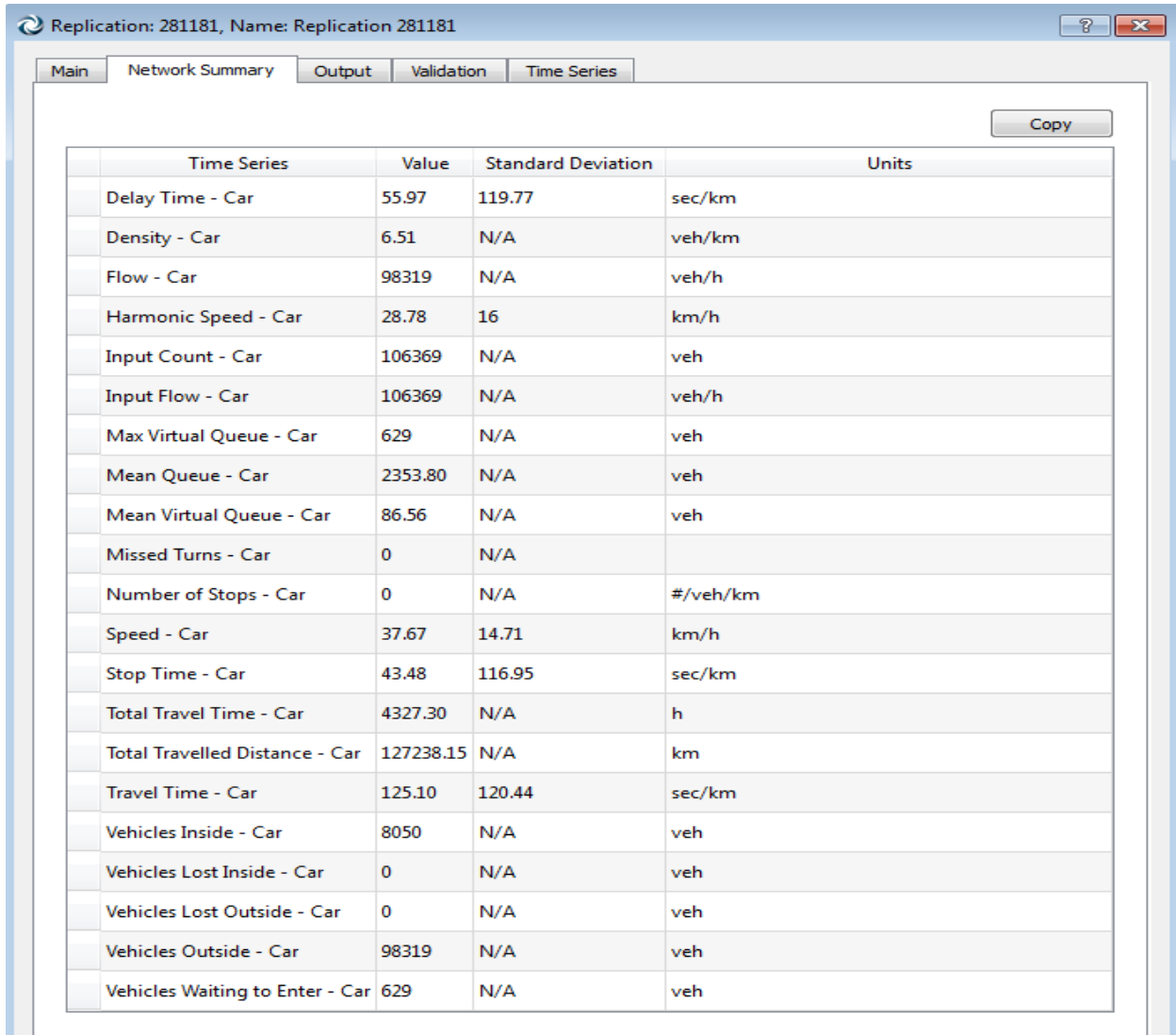
3.3 Output from AIMSUN Micro-simulator

Data produced by the micro-simulator is in the following forms:

1. It is possible to obtain data concerning each individual vehicle, such as its position, speed, distance covered and path followed. This can be seen as a virtual sensor collecting GPS data for each vehicle.

2. It is also possible to have readings similar to those of a sensor placed on the road infrastructure, that is collecting traffic data (i.e., traffic densities and flows) at a fixed position. Moreover, other grouped quantities can be obtained, such as number of vehicles entering or exiting a zone, or number of vehicles following a particular path.
3. Based on the output of the simulator it is possible to calculate other indicators: pollution, consumption of fuel, emission of carbon dioxide, etc.

The output of the micro-simulator can be visualized thanks to the AIMSUN tool, both in the form of a flow of vehicles moving in the city (a simulated video) or in a table form, as it is shown in Figures 9 and 10, or as relational database SQLite. The exported data can be loaded into Matlab by partners using the Matlab's `csvread` functions or it can be easily processed using the SQLite module of Matlab or alternatively using `sqlitebrowser` (<http://sourceforge.net/projects/sqlitebrowser>). Examples of exported data are shown in Figures 11 and 12.



Time Series	Value	Standard Deviation	Units
Delay Time - Car	55.97	119.77	sec/km
Density - Car	6.51	N/A	veh/km
Flow - Car	98319	N/A	veh/h
Harmonic Speed - Car	28.78	16	km/h
Input Count - Car	106369	N/A	veh
Input Flow - Car	106369	N/A	veh/h
Max Virtual Queue - Car	629	N/A	veh
Mean Queue - Car	2353.80	N/A	veh
Mean Virtual Queue - Car	86.56	N/A	veh
Missed Turns - Car	0	N/A	
Number of Stops - Car	0	N/A	#/veh/km
Speed - Car	37.67	14.71	km/h
Stop Time - Car	43.48	116.95	sec/km
Total Travel Time - Car	4327.30	N/A	h
Total Travelled Distance - Car	127238.15	N/A	km
Travel Time - Car	125.10	120.44	sec/km
Vehicles Inside - Car	8050	N/A	veh
Vehicles Lost Inside - Car	0	N/A	veh
Vehicles Lost Outside - Car	0	N/A	veh
Vehicles Outside - Car	98319	N/A	veh
Vehicles Waiting to Enter - Car	629	N/A	veh

Figure 8 : Example of exported data

Simulation Vehicle: 10018235 (Temporary) (Layer: Network)		
Static Attributes		
Attribute	Value	Units
Aimsun ID	16963	
Vehicle Type	Car	
Arrival Time	00:09:31	
Has Been Enrouted	No	
Tracked	No	
Equipped	No	
Guidance Acceptance	100	%
Cooperation Degree	24.047	%
Length	3.6781	m
Width	2	m
Maximum Acceleration Desired	3.20349	m/s ²
Deceleration Desired	-4.32354	m/s ²
Maximum Deceleration Desired	-6.86245	m/s ²
Mean Speed Desired	112.692	km/h
Speed Limit Acceptance	1.11234	
Minimum Distance Between Vehicles	0.72059	m
Maximum Give Way Time	12.7247	
Reaction Time	0.8	
Reaction Time at Stop	1.2	
Reaction Time at Traffic Light	1.6	
Vehicle Class	None	
Stay in Fast Lane	No	
Minimum Headway	0	Secs
Sensitivity Factor	1	

Figure 9 : Example of exported data

Simulation Vehicle: 10018235 (Temporary) (Layer: Network)		
Dynamic Attributes		
<input type="checkbox"/> Follow <input type="checkbox"/> Get Floating Vehicle Data <input type="checkbox"/> Collect Time Series Data		
<input type="checkbox"/> Store Path in a Google Earth File		
Attribute	Value	Units
Current Speed	50.8139	km/h
Previous Speed	47.7456	km/h
Mean Speed Desired	55.617	km/h
Position	714834 5005602.21	m
Number of Missed Turns	0	
Zone	1	

Figure 10 : Example of exported data.

1	did	oid	ent	sectionId	xCoord	yCoord	timeSta	speed	travelledDistance	acceleration
2	12079	23	1	12017	713128.2	5006530.59	28811.2	55.87	8.82	0
3	12079	23	2	12017	713140.58	5006531.43	28812	55.87	21.24	0
4	12079	23	3	12017	713152.97	5006532.27	28812.8	55.87	33.66	0
5	12079	23	4	12017	713165.36	5006533.1	28813.6	55.87	46.07	0
6	12079	23	5	12017	713177.75	5006533.94	28814.4	55.87	58.49	0
7	12079	23	6	12017	713190.13	5006534.78	28815.2	55.87	70.9	0
8	12079	23	7	12017	713202.52	5006535.61	28816	55.87	83.32	0
9	12079	23	8	12017	713214.91	5006536.45	28816.8	55.87	95.73	0
10	12079	23	9	12017	713227.29	5006537.29	28817.6	55.87	108.15	0
11	12079	23	10	12017	713239.68	5006538.13	28818.4	55.87	120.56	0
12	12079	23	11	12017	713252.07	5006538.96	28819.2	55.87	132.98	0
13	12079	23	12	12017	713264.45	5006539.83	28820	55.87	145.39	0
14	12079	23	13	12017	713276.8	5006541.11	28820.8	55.87	157.81	0
15	12079	23	14	12017	713289.13	5006542.55	28821.6	55.87	170.22	0
16	12079	23	15	12017	713301.46	5006544	28822.4	55.87	182.64	0
17	12079	23	16	12017	713313.79	5006545.45	28823.2	55.87	195.06	0
18	12079	23	17	12017	713326.12	5006546.9	28824	55.87	207.47	0
19	12079	23	18	12017	713338.48	5006548.01	28824.8	55.83	219.88	-0.01
20	12079	23	19	4808	713350.85	5006548.99	28825.6	55.83	232.29	0
21	12079	23	20	4808	713362.6	5006549.96	28826.4	50.32	244.08	-1.91
22	12079	23	21	4808	713373.14	5006550.83	28827.2	44.8	254.65	-1.91
23	12079	23	22	4808	713382.45	5006551.59	28828	39.29	263.99	-1.91
24	12079	23	23	4808	713390.54	5006552.26	28828.8	33.78	272.11	-1.91
25	12079	23	24	4808	713396.96	5006550.15	28829.6	29.34	279.13	-1.54
26	12079	23	25	4808	713402.67	5006547	28830.4	29.34	285.65	0
27	12079	23	26	5558	713406.55	5006536.05	28832	29.34	298.68	0
28	12079	23	27	5558	713404.98	5006527.84	28832.8	37.64	307.05	2.88
29	12079	23	28	5558	713403.13	5006518.22	28833.6	44.07	316.84	2.23
30	12079	23	29	5558	713400.95	5006507.65	28834.4	48.56	327.63	1.56
31	12079	23	30	5558	713398.63	5006496.45	28835.2	51.48	339.07	1.01

Figure 11: Example of exported data

Figure 11 shows an example of data exported from AIMSUN for the individual vehicles locations, i.e., data that can be interpreted as a virtual sensor simulating a GPS placed in the car. The table gives information about location of vehicles, speed, travelled distance and acceleration of each vehicle for a particular timestamp and section of the road. Columns “xCoord” and “yCoord” represent the X and Y co-ordinates of each vehicle. These co-ordinates are in standard UTM format. Each vehicle has its own identifier in column “oid”. A road in AIMSUN consists of sections. Each section has an index (column “ent”) related to the vehicle path and an exact identifier (column “sectionid”). “Acceleration” column can be negative and in this case it is deceleration.

1	did	oid	eid	sid	ent	countv	countveh	flow	flow_D	speed	speed_D	occupancy	occupancy_D	density	density_D	headway	headway_D
2	12079	12700	Sensor-2	0	1	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
3	12079	12701	Sensor-3	0	1	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
4	12079	12703	Sensor-1	0	1	1	-1	240	-1	55.41582	-1	3.525961	-1	4.273908	-1	-1	-1
5	12079	12700	Sensor-2	1	1	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
6	12079	12701	Sensor-3	1	1	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
7	12079	12703	Sensor-1	1	1	1	-1	240	-1	55.41582	-1	3.525961	-1	4.273908	-1	-1	-1
8	12079	12700	Sensor-2	2	1	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
9	12079	12701	Sensor-3	2	1	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
10	12079	12703	Sensor-1	2	1	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
11	12079	12700	Sensor-2	0	2	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
12	12079	12701	Sensor-3	0	2	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
13	12079	12703	Sensor-1	0	2	3	-1	720	-1	53.8901	-1	10.69514	-1	12.33688	-1	4.414449	-1
14	12079	12700	Sensor-2	1	2	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
15	12079	12701	Sensor-3	1	2	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
16	12079	12703	Sensor-1	1	2	3	-1	720	-1	53.8901	-1	10.69514	-1	12.33688	-1	4.414449	-1
17	12079	12700	Sensor-2	2	2	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
18	12079	12701	Sensor-3	2	2	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
19	12079	12703	Sensor-1	2	2	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
20	12079	12700	Sensor-2	0	3	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
21	12079	12701	Sensor-3	0	3	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
22	12079	12703	Sensor-1	0	3	2	-1	480	-1	54.25145	-1	7.180911	-1	8.736336	-1	2.951555	-1
23	12079	12700	Sensor-2	1	3	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
24	12079	12701	Sensor-3	1	3	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
25	12079	12703	Sensor-1	1	3	2	-1	480	-1	54.25145	-1	7.180911	-1	8.736336	-1	2.951555	-1
26	12079	12700	Sensor-2	2	3	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
27	12079	12701	Sensor-3	2	3	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
28	12079	12703	Sensor-1	2	3	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
29	12079	12700	Sensor-2	0	4	0	-1	0	-1	-1	-1	0	-1	0	-1	-1	-1
30	12079	12701	Sensor-3	0	4	3	-1	720	-1	56.91246	-1	11.08334	-1	12.99711	-1	0.779375	-1
31	12079	12703	Sensor-1	0	4	4	-1	960	-1	52.87468	-1	14.3734	-1	16.89318	-1	3.280399	-1

Figure 12: Example of exported data from the simulator virtual sensors.

Figure 12 shows the output data exported from AIMSUN for the virtual sensors placed in the simulator, representing a sensor placed on the road, similar to the real sensors that Grenoble Traffic Lab has on the South Ring. There are many attributes which we are receiving in this type of output, but mainly: vehicle count (column “countveh”), occupancy, density, headway, etc. The suffix “D” for each field shows the Standard deviation for that particular field. This table also gives a list of sensors names (column “eid”) and types of vehicles (column “sid”) used in the simulation. Types can be: all, only cars and only trucks. Column “ent” is for time intervals.

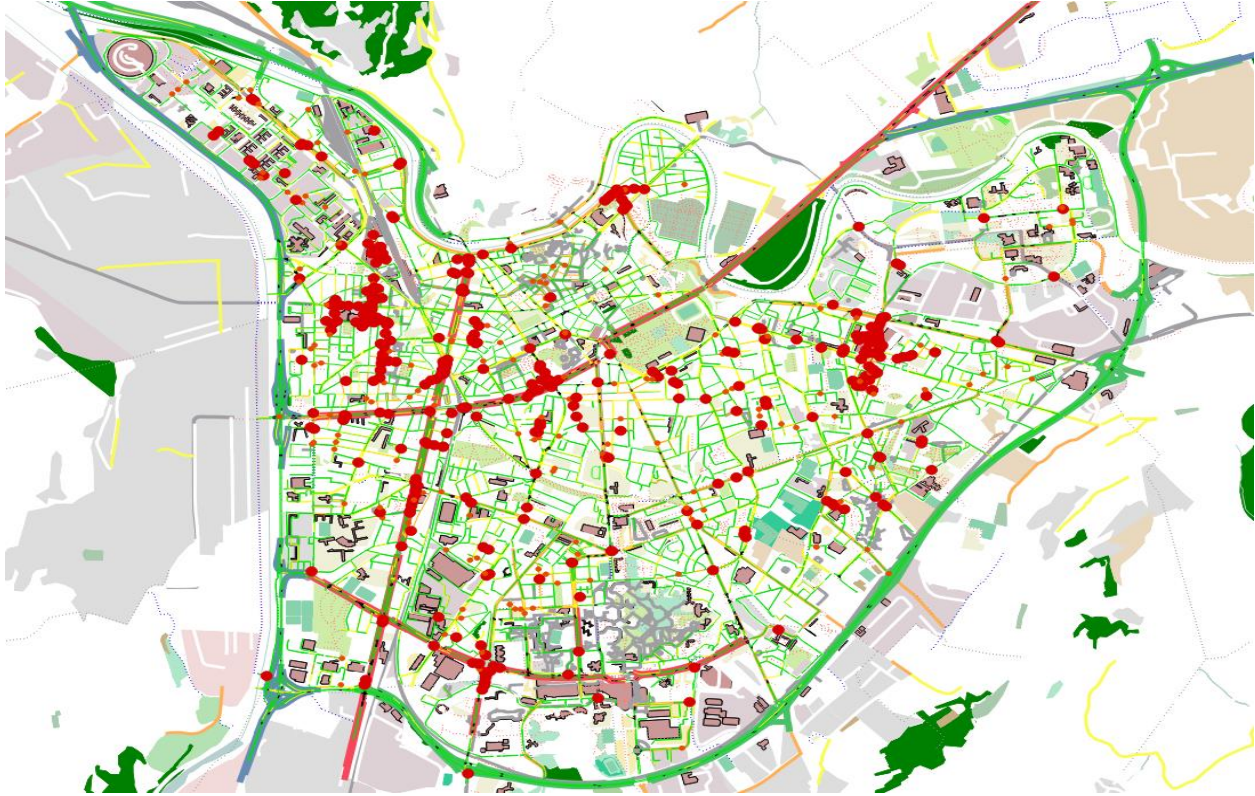


Figure 13: Visualization of traffic congestion at different locations.

The Figure 13 above displays one of the visual types of output that we can extract from the Aimsun software, where you can select different parameter like speed, density, number of vehicle on each section. The color of a section changes according to the parameter values. The figure above gives information about traffic congestion at different locations in the Aimsun model, which is indicated by a red circle. As we can see there are several parts of the Grenoble city model, where we can detect congestion during simulation (mainly in the left and right side).

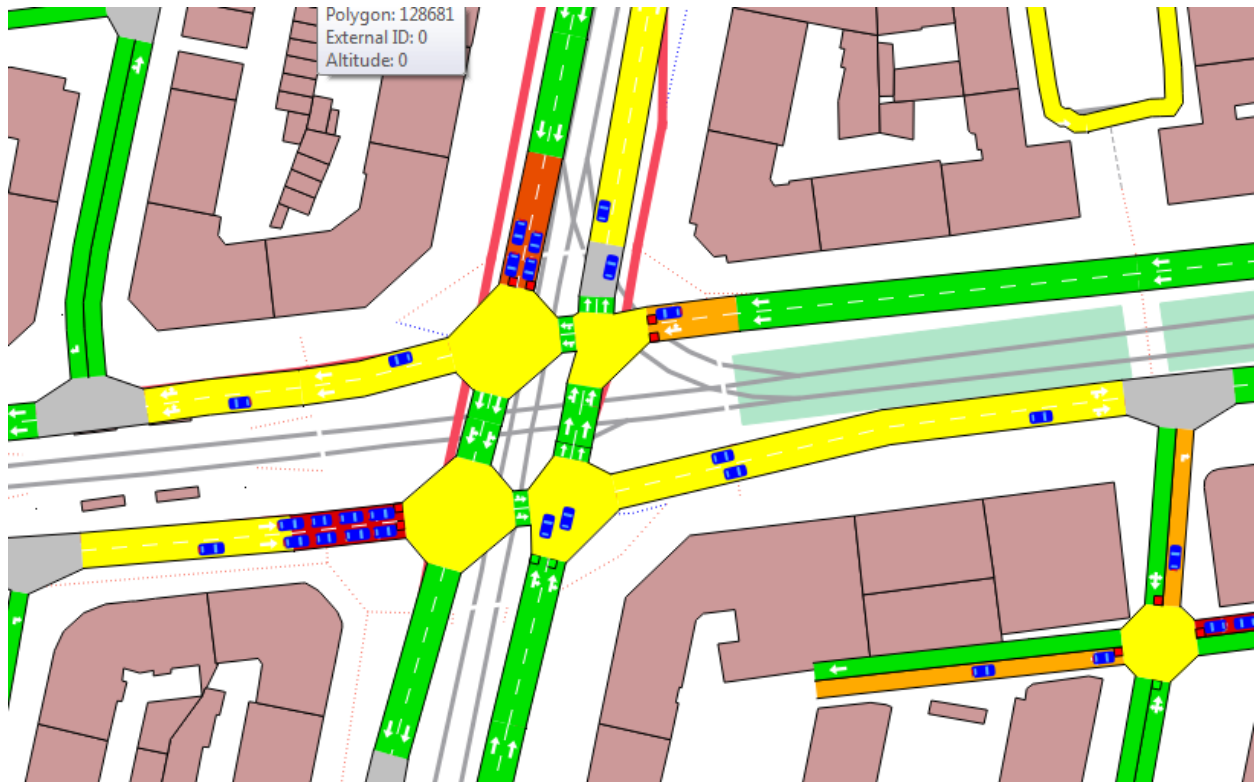


Figure 14: Colormap concentration of vehicles at given sections of the traffic network.

Figure 14 above is an example of Aimsun output visualization, where a density parameter has been set to display the concentration of vehicle at individual section in accordance with different colors. We can clearly see that there are several sections which are red because their density is higher than a given threshold while other sections are colored in orange, yellow and green.

4. Conclusion

This document describes the final version the micro-simulator, based on AIMSUN commercial software, and the stumbling blocks that needed to be overcome in order to develop the simulator. This final version includes the overall model for Grenoble city and several APIs to connect Aimsun with SPEEDD Protocol and close the loop for controlling. There are more than 1000 intersections present in the model and more than 50,000 flow of vehicles.