UNIVERSITÀ
DI PAVIA

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL, COMPUTER AND BIOMEDICAL ENGINEERING

INDUSTRIAL AUTOMATION ENGINEERING - ROBOTICS AND MECHATRONICS

Master's Degree Thesis

Traffic-Simulated Driving Assistance in a Rehabilitative Framework

---

Supervisors:

Prof. Antonella Ferrara

PhD. Nikolas Sacchi

Candidate:

Alessandro Galassi

Matr. n. 556546

Academic Year 2025/2026

Contents

List of Figures

List of Tables

List of Equations

Abstract

The idea behind this thesis is to deliver a comprehensive controller able to offer for a patient enrolled in a rehabilitation program a solution to drive again. The person in question is affected by certain pathologies that preclude him full ability to drive. The reference framework is the Annex III of the European Directive 2006/126/EC [1] (and its later updates 2009/113/EC and 2014/85/EU), transposed into Italian law by Legislative Decree 59/2011 [2] and anchored to Article 119 (Codice della Strada) and Articles 319–329 of Presidential Decree 495/1992. From this, Annex III lists the categories of relevant pathologies.

Along these conditions we have the *locomotor disabilities* (musculoskeletal) for a disabled person (for example after an incident that the patient needs a rehabilitation), *cardiovascular conditions* when they entail a risk of a sudden event (e.g. arrhythmias, *heart failure* with reduced ejection fraction, ischaemic heart disease), *neurological diseases* with central or peripheral nervous system conditions with sensory, motor, or cognitive effects that affect safety (for instance Parkinson's, multiple sclerosis, stroke sequelae), *medicinal prescriptions* that modifies the reflex time or attention of the person.

The intention is to give a questionnaire user-friendly, asking for the physical conditions (and not only) at the beginning of the day, knowing the path that is intended to follow with the car for that date, then deliver the best possible solution for each specific case.

This thesis has not the claim to remove the person from driving but helping him to recover and return fully to the control, centrality and responsibility, without trying to substitute him.

So, the work will be developed following this ideas, starting defining the State of the Art of the tools and concepts needed to define correctly the problem. Then, will move to defining the way to simulate a real traffic scenarios with different type of vehicles and an EGO car (our patient in study) that needs to go from a point A to a point B. After that, it will explain a method to reduce this tasks into a single Python file and try alternative paths along the same

destination to avoid traffic. Furthermore it enters in the questionnaire problem where I try to put in numbers and questions all the possible components that each patient affected with different pathologies wants to avoid (complex streets, sharp curves, safety distance from the car in front, velocity limits).

After all the evolutions it enters into the final part developing all the complex tasks described and condensed into a Model Predictive Control working in the macroscopic frame able to predict the future traffic, linked with the questionnaire and the adaptive routing, and then deliver a velocity reference with an appropriate path to follow.

This method tries to suggest "the best possible solution" for different cases, clearly with some approximation that will be deeply analysed, concluding that does not distance itself from real scenarios.

Thus, the proposed framework is validated on simulation traffic scenarios reproducing realistic urban and extra-urban conditions, showing that the controller is able to integrate the patient's declared limitations into both the routing and the velocity reference while keeping the resulting trajectories feasible and comfortable. The main contribution of this work is therefore a unified, patient-centred approach that couples all the features that will be described in the following chapters and deliver an optimal reference and guide to follow.

Sommario

L'idea proposta alla base di questa tesi è quella di realizzare un controllore adattivo in grado di offrire, a un paziente inserito in un programma di riabilitazione, una possibile soluzione per tornare alla guida. La persona in analisi è affetta da patologie che ne compromettono la piena capacità; il quadro normativo di riferimento a tal riguardo è l'Allegato III della Direttiva Europea 2006/126/CE [1] (e i suoi successivi aggiornamenti 2009/113/CE e 2014/85/UE), recepita nell'ordinamento italiano dal Decreto Legislativo 59/2011 [2] e correlata all'Articolo 119 del Codice della Strada e agli Articoli 319–329 del D.P.R. 495/1992.

Seguendo le linee guida introdotte, l'Allegato III elenca alcune delle categorie di patologie rilevanti, tra queste le *disabilità locomotorie*, anche dette muscolo-scheletriche, (per esempio a seguito di un incidente dopo il quale il paziente necessita un percorso per ritrovare il funzionamento degli arti coinvolti nello scontro), le *patologie cardiovascolari* quando comportano il rischio per il guidatore di imbattersi in un evento improvviso (ad es. aritmie, scompenso cardiaco con frazione di eiezione ridotta, cardiopatia ischemica), le *malattie neurologiche* che ricoprono i malfunzionamenti del sistema nervoso centrale o periferico con effetti sensoriali, motori o cognitivi che incidono sulla sicurezza (per esempio morbo di Parkinson, sclerosi multipla, esiti di ictus), e infine le *prescrizioni farmacologiche* diagnosticate ad un paziente in cura che alterano i tempi di reazione o l'attenzione alla guida.

Si propone quindi al paziente un questionario con domande e risposte, su scala da 1-5, sulla situazione fisica e cognitiva in cui verte. Tali informazioni vengono collezionate e, conoscendo il tragitto che intende percorrere per tale giorno, si fornisce la soluzione *ottima* per ogni esigenza fisica e temporale.

L'obbiettivo è quindi quello di fornire uno strumento di supporto per il paziente, che possa aiutarlo a ritornare alla guida e alla piena coscienza del mezzo, senza avere l'intenzione di destituirlo dalla figura responsabile di pilota (per esempio fornendo un sistema di guida autonoma completa).

Il lavoro svolto sarà quindi presentato seguendo queste idee cardini, descrivendo inizialmente lo Stato dell'Arte dei concetti teorici per definire correttamente il problema, e a seguire dei software simulativi utilizzati, con scenari di traffico reali e l'auto del paziente in esame (denominata EGO) dove se ne controlla il tragitto. Dopo averne ottimizzato e descritto tale processo in un unico file Python, si introduce un metodo adattivo per la ricerca della strada da percorrere evitando tratti ad alta condensazione di veicoli. Dunque verrà introdotta la parte centrale del lavoro con l'introduzione del questionario e l'algoritmo che ne sta dietro con analisi delle domande, pesi e equazioni che provano a modellizzare la situazione fisica del paziente e a tramutarla in richieste diverse di strade da percorrere (arterie complesse, curve strette, distanza di sicurezza dall'auto che precede e limiti di velocità).

Infine si entra nell'ultima parte dove viene discusso il modello predittivo che condensa tutte quelle funzioni in un unico controllore ottimo, Model Predictive Control (MPC), in grado di prevedere il traffico che l'auto in analisi affronterà tramite un modello di traffico Macroscopico, unito al questionario e il metodo di ricerca della strada adattivo, in grado di fornire la velocità ottima al paziente da seguire e la corrispettiva strada ottima.

Tale metodo si propone di offrire la "soluzione migliore" per differenti situazioni, chiaramente presentando alcune approssimazioni che saranno analizzate in seguito che però non ci portano a conclusioni lontane da scenari reali studiati in precedenza e proposti in letteratura.

In conclusione il lavoro svolto si pone in un ambiente simulativo riproducendo scene urbane e condizioni extra-urbane verosimili, dimostrando che il controllore è in grado di assistere il paziente fornendo una reference e una strada adatta a diversi scenari risultando sempre fattibile e confortevole, seguendo il più possibile le esigenze fisiche e psichiche. Il contributo illustrato da questa tesi è quindi costruito attorno alla figura del paziente e nella sua evoluzione proporrà diverse implementazioni che verranno analizzate nei seguenti capitoli.

Chapter 1

Patient in Analysis: Return to Drive

The whole framework proposed in this thesis is built around a patient who wishes to drive again, and every later choice is namely to give back to a convalescent driver as much autonomy as it is safe to return. This chapter therefore describes who this patient is, which conditions may have limited his or her ability to drive, and how contemporary rehabilitation medicine accompanies such a person back behind the wheel.

1.1 The patient and the conditions that limit driving

The patient in analysis is not defined by a single disease but by a *family* of conditions, the same categories that Annex III of the European Directive 2006/126/EC [1] recognises as relevant to fitness to drive. What they share is that each of them degrades one or more of the functions on which safe driving rests, *vision and perception*, *cognition* (attention, executive control, reaction time) and *motor control*, sometimes permanently and sometimes fluctuating during the day.

Locomotor (musculoskeletal) disabilities. After a trauma, an amputation, a spinal or joint injury, or a long immobilisation, the patient may present reduced strength, a limited range of motion, pain, or the loss of a limb. For driving this translates into difficulty in operating the pedals or the steering wheel, a reduced ability to rotate the head and trunk to check blind spots, and *slower*

emergency manoeuvres. The residual capacity is often stable in time, so the limitation is essentially mechanical, and it is the category best addressed by vehicle adaptations.

Cardiovascular conditions. Arrhythmias, ischaemic heart disease and heart failure with reduced ejection fraction matter not for a continuous deficit but for the risk of a *sudden incapacitating* event (syncope, malignant arrhythmia) at the wheel. Here fitness to drive is a probabilistic judgement on the likelihood of an abrupt loss of control, and it is tightly regulated, especially for professional drivers.

Neurological diseases. This is the broadest and most driving-relevant group, because it can involve different branches of safe driving. Stroke sequelae may leave hemiparesis, visual field losses (hemianopia or quadrantanopia), unilateral spatial neglect, aphasia and a general slowing of information processing. Also, Parkinson's disease adds bradykinesia, rigidity, tremor, longer reaction times and freezing episodes, compounded by the effect of its own medication. Multiple sclerosis brings fatigue, spasticity, visual disturbances and cognitive slowing; and vestibular or peripheral disorders cause vertigo, dizziness and imbalance. The limit that is recurrent in these pathologies is *fatigue*, which erodes attention long before it is perceived.

Pharmacological effects. Many prescribed treatments, sedatives, anxiolytics, antiepileptics, opioids and some antihypertensives, prolong reaction time, blunt attention or induce drowsiness. This limitation is *time-varying*, typically strongest in the hours following a dose, which is precisely why an assessment tied to the day and time of the intended trip is meaningful.

The degree of impairment differs enormously between patients, and even for the same patient it changes from one day to the next and within hours. It is this variability, more than any single diagnosis, that motivates a questionnaire.

1.2 The legal and medical framework

Annex III of Directive 2006/126/EC [1] (with its updates 2009/113/EC and 2014/85/EU) fixes the minimum medical standards for driving, distinguishing *Group 1* (ordinary) from *Group 2* (professional) drivers and allowing licences of reduced validity subject to periodic medical review, or licences restricted

to vehicles fitted with adaptations. In Italy the Directive is transposed by Legislative Decree 59/2011 [2], anchored to Article 119 of the "Codice della Strada" and to Articles 319–329 of Presidential Decree 495/1992, with the Commissione Medica Locale entrusted with deciding any restrictions and the review interval.

A person may be fit under conditions with vehicle adaptations, or with restrictions in time or area and with periodic re-evaluation. This logic is the space in which a supportive system is meant to operate, guaranteeing an assist for the patient and safeguarding health and safety of other drivers and pedestrians on the street.

1.3 Rehabilitation pathways in current medical practice

Returning to driving after an illness or injury is a structured and multidisciplinary pathway, today recognised as a sub-field of occupational therapy known as *driver rehabilitation* [3], [4]. Its stages can be summarised as follows.

1. **Medical stabilisation and screening.** The treating physician first establishes that the condition is stable and that no absolute contra-indication exists, then refers the patient for a formal driving evaluation.

2. **Off-road clinical assessment.** Usually led by an occupational therapist, it evaluates the driving-relevant functions, visual acuity and fields, cognition and perception (attention, visuospatial and executive abilities, processing speed, neglect) and physical function (strength, range of motion, coordination, reaction). Standardised instruments, from the Trail Making Test and road-sign recognition tasks to dedicated batteries such as the Occupational Therapy Driver Off-Road Assessment Battery [5] declare who is ready for an on-road test.

3. **On-road evaluation.** Considered the gold standard, it is conducted in a dual-control car by a driver rehabilitation specialist (in many countries an occupational therapist with additional certification, such as the Certified Driver Rehabilitation Specialist of ADED [4]). Its outcome can be:

   (a) fit;

   (b) fit with adaptations;

(c) needs of retraining;

(d) not (yet) fit.

4. **Retraining and driving simulators.** Where remediation is required, contextual practice is delivered (including driving simulators) which offer a safe and repeatable exposure to complex scenarios. In case of several deficits this contextual training is more effective than non-contextual cognitive exercises [3].

5. **Vehicle adaptation.** For stable motor deficits, *adaptive controls*, hand controls, left-foot accelerators, steering knobs or spinners and relocated secondary controls, let the patient drive safely [4]. For a micro-level approach of the problem posed in this thesis, this aspect could be interesting, treating them as inputs for the driving active assistance MIMO system.

6. **Monitored return.** The return is often progressive, starting from what is known such as familiar routes, or simpler cases e.g. daylight and low traffic cases. This procedure is accompanied by periodic medical review.

When such a pathway is followed the results are encouraging; stroke survivors trained in a driving simulator passed the formal driving test far more often, 73 % compared to 42 % at three months [3].

1.4 Positioning of this thesis within the pathway

The system proposed complements the clinical pathway, living inside the vehicle once the patient has already been deemed fit to drive, possibly with restrictions. Its role is to analyse the patient's health state and collect this influence matrix through the questionnaire of Chapter 5 into two concrete supports:

- a *route* that avoids the features that the clinical profile flags as risky (e.g. sharp curves, cognitively demanding streets), with the traffic-aware feature,
- and a comfortable, optimal *speed reference*.

The aim of the thesis is to develop these tools that make this process possible, beginning with the theoretical background of the next chapter.

Chapter 2

Traffic Dynamics and the Simulation Framework - State of the Art

This chapter is used to describe all the fundamental concepts about traffic necessary for this thesis. It's based mainly on the work done by Ferrara, Sacone and Siri, *Freeway Traffic Modelling and Control* [6], whose early chapters provide a rigorous picture of traffic flow theory: from the basic variables to the first- and second-order dynamic models. It starts with the clarification about the Microscopic and Macroscopic definition problems, how and when are defined in the work. In order to correctly represent all the results achieved, the plots that draw the fundamental variables will be discussed and then recalled during the developing part of the thesis, in Chapter 3 and following. So, it's useful to get used to space–time diagrams, time series or plots representing the relations between flow and density or between speed and density.

Then it will introduce the Simulation Platform SUMO describing how it works and how it is implemented for the task required in this thesis.

2.1 Fundamental variables

In order to describe the vehicles behaviour and their interaction on the road, different models were developed; starting with the first one in 1955 by Lighthill and Whitham [7]. After this proposal, several models were studied with different levels of details. In this huge variety of models, the most appropriate to be chosen for a given real case basically depends on the type and scope of the

considered application.

In these articles [8], [9] different methods to classify these models were defined, but the most common one is related to the *level of details*, with Microscopic, Mesoscopic and Macroscopic models analysed later. Also the space-time is distinguished from the continuous or discrete classification.

The following figure, that takes inspiration from [6], describes these classes:

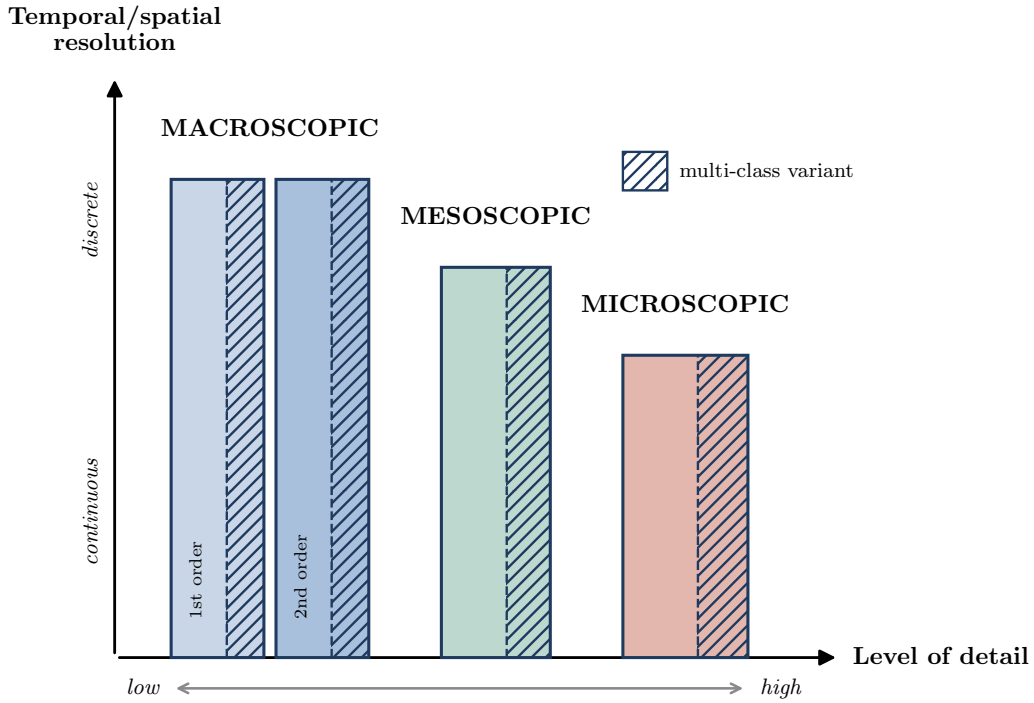


Figure 2.1: Classification in Space and Time with Micro, Meso, Macroscopic classes.

- *Microscopic models*, in which the dynamics of all vehicles and their interactions are represented in detail. Normally, each vehicle is described with a dynamic model, which includes different parameters (e.g. desired speed or acceleration capabilities of the vehicle, aggressiveness, and reaction times of the driver). These types of models are very detailed and, consequently, can be computationally intensive for representing large road networks. This model is the one used by SUMO (Section 2.4).

- *Macroscopic models*, in which the traffic dynamics are represented at an aggregate level. Specifically, the flow of vehicles is seen as a unique stream (following the analogy with the flow of fluids) and its dynamics are

described by means of density, mean speed and flow. These models are less computationally intensive than microscopic models and can allow fast simulations also for large-scale traffic networks. Macroscopic models are further classified according to the number of state variables, in first-order or second-order (Section 2.3), and according to the number of represented vehicle types, in one-class or multi-class models.

- *Mesoscopic models*, which present an intermediate level of detail. They do not distinguish individual vehicles but represent the heterogeneity of the drivers choices in probabilistic terms.

In order to outline the traffic flow, Table 2.1 explains the variables needed.

Table 2.1: The basic variables of macroscopic traffic flow theory.

Variable	**Description**	**Unit**
Speed v	distance travelled per unit time (micro: one vehicle; macro: the stream)	km/h
Density ρ	number of vehicles per unit length of road	veh/km
Flow q	number of vehicles passing a fixed point per unit time	veh/h
Space headway	front-to-front distance between successive vehicles	m/veh
Time headway	time lag between successive vehicles passing a point	s/veh
Occupancy	fraction of time a point is covered by a vehicle	%
Demand	number of vehicles wishing to enter the road per unit time	veh/h

The most important factors are density, speed and flow. Of the three, density is the one that most directly captures how "full" a road is. Written ρ and measured in veh/km (sometimes called *concentration*), it counts how many vehicles occupy a given stretch of road at a fixed instant. An empty road has $\rho = 0$, and as traffic builds up the density rises until, at the *jam density* ρ_{jam}, the vehicles sit still and motion stops altogether. *Space headway* indicates the distance between two vehicles, given by the distance between the front and the rear bumpers of the vehicles, plus the length of the leading vehicle (measured in [m/veh]). It is a microscopic variable, referred to one single vehicle. Since the density is a difficult variable to be measured, *occupancy* is then measured and used to estimate it, defined as the time in which a given location is occupied by vehicles, and is usually expressed as a percentage.

For a single vehicle, the speed v is straightforward, but as soon as we describe a whole stream it has to be an average, and there are two inequivalent ways

of taking that average [6]. One can average the *instantaneous speeds* of the vehicles passing a fixed location over a time interval, obtaining the *time-mean* (arithmetic) speed

$$v_t = \frac{1}{n}\sum_{i=1}^{n} v_i, \tag{2.1}$$

or one can average the speeds of the vehicles present on a stretch at a fixed instant, obtaining the *space-mean* (harmonic) speed

$$v_s = \frac{n}{\sum_{i=1}^{n} 1/v_i}. \tag{2.2}$$

Macroscopic theory works with the space-mean speed, because it's related to the movement of the traffic stream.

Traffic Flow q [veh/h], also called *flow* closes the picture defined as the rate at which vehicles cross a given point, the same unit is often adopted to represent the *demand*. An important variable associated with traffic flow is *time headway*, expressed in [s/veh] and as explained in Table 2.1 has a microscopic meaning. The average time headway can be computed as the inverse of traffic flow.

The most important quantities are not independent but bounded together by the fundamental identity

$$q = \rho \cdot v, \tag{2.3}$$

This is the backbone of the whole macroscopic theory, since knowing any two of the variables, the remaining one is immediately obtainable.

2.2 Traffic Diagrams

This section explores the most used traffic diagrams.

Flow–Density and Speed–Density Diagrams. As the name suggests, the Flow-Density and Speed-Density describe respectively the flow and speed, for a given time period and a given road segment, over density. These type of data are normally used for heterogeneous traffic conditions [6]. This two plots can be found in Figures 3.2 of Chapter 3.

Time Series. They are generally used to plot the profiles of aggregated quantities, referred to a given location, over a given time period. These graphs

are useful to understand the evolution in time of the plotted variables, e.g. understand the behaviour during simulation of a busy edge. The Time-Series for Flow, Density and Avg. Speed are shown in Figure 3.5 of Chapter 3.

Space-Time Diagrams. They are used to plot the trajectories of vehicles travelling in a given road segment for a certain time interval, reporting time on the x axis and space on the y axis. Taking the example from [6],

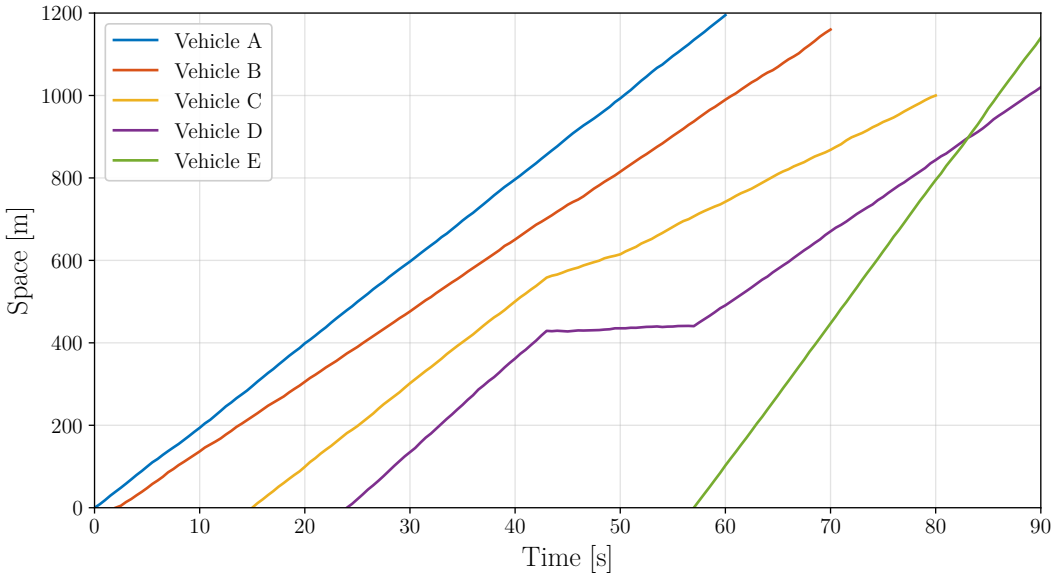


Figure 2.2: Example of trajectories of five vehicles in a Space-Time Diagram [6].

we can appreciate some features:

- the number of trajectories crossing a horizontal segment associated with a given location and a certain time interval corresponds to the *traffic flow*;
- the number of trajectories crossing a vertical segment associated with a given time instant and a certain space interval corresponds to the *traffic density*;
- the slope of the trajectory represents the local speed of the vehicle in a given location in a determined time instant. If in the plot the trajectory is horizontal (the slope is equal to zero), the corresponding vehicle is stationary so blocked in traffic;
- the vertical distance between trajectories represents the *distance headway*, whereas the horizontal distance between two trajectories indicates the *time headway*;

- two crossing trajectories in a certain time correspond to an overtaking between the two vehicles.

Looking at Figure 3.6 of Chapter 3 we can observe some congested areas during the final iterations of the simulation, when SUMO has already injected a consistent number of vehicles for the dimension of the map, resulting as said in a jam. The slopes are equal, if the road is free, meaning that SUMO pushes the background traffic at the road limit speed.

2.2.1 The Fundamental Diagram

A central concept in traffic flow theory is the *Fundamental Diagram* (FD), the steady-state curve that encodes the characteristic behaviour of a road section in the flow-density plane. It traces back to Greenshields, who as early as 1935 fitted a linear speed-density relationship to field data [10]. The Greenshields model postulates the linear law

$$v(\rho) = v_f \left(1 - \frac{\rho}{\rho_{\text{jam}}}\right), \tag{2.4}$$

where v_f is the *free-flow speed*, the speed of drivers on an essentially empty road. Substituting (2.4) into (2.3) gives the parabolic flow-density curve

$$q(\rho) = v_f \, \rho \left(1 - \frac{\rho}{\rho_{\text{jam}}}\right), \tag{2.5}$$

which reaches its maximum, the capacity $q_{\max}$, at the *critical density*:

$$\rho_c = \rho_{\text{jam}}/2. \tag{2.6}$$

The critical density ρ_c is the density at which the flow reaches its maximum, the capacity $q_{\max}$, as clear from Figure 2.3.

In the *free-flow* regime ($\rho < \rho_c$) density is low, vehicles interact little, and flow increases almost linearly with density as more vehicles use the available capacity. In the *congested* regime ($\rho > \rho_c$) density is high, interactions dominate, and flow actually decreases as more vehicles are added. The speed-density curve, by contrast, is monotonically decreasing throughout because a vehicle always travels slower in a denser stream, whichever regime it is in.

The important features are:

- When traffic density $\to 0$, the flow reduces $\to 0$ consequently.

- The slope of the tangent of the curve in zero represents the free-flow speed, i.e. the speed of vehicles in absence of traffic or with a very low flow.
- If the density grows, the flow follows the same trend with ≈ linear trend (due to the limited interactions among vehicles) that is maintained to a certain value of density, called critical density, evaluated in the central picture (b) of Figure 2.3. When the density is equal to the critical value, the flow reaches its maximum point, called capacity (in general is a measure of the maximum throughput of a road). Beyond the critical value, a further increase in the density corresponds to a decrease in the flow, due to the increasing interactions among vehicles resulting in a reduction of the average speed.
- The traffic flow becomes zero when the density reaches its maximum value, called jam density.

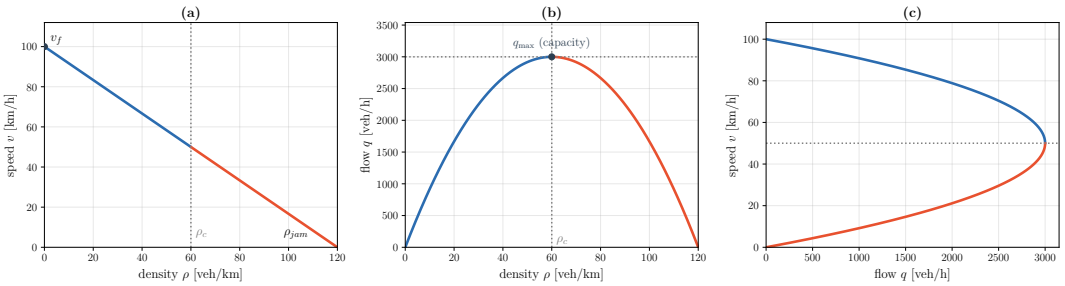


Figure 2.3: The Greenshields fundamental relationships, (a) the linear speed-density law of (2.4); (b) the resulting flow-density parabola of (2.5), peaking at capacity $q_{\max}$ at the critical density Equation 2.6; (c) the speed-flow curve. The free-flow branch ($\rho < \rho_c$, blue) and the congested branch ($\rho > \rho_c$, red) are highlighted. The values are illustrative ($v_f = 100\,\text{km/h}$, $\rho_{\text{jam}} = 120\,\text{veh/km}$).

Real measurements deviate from Figure 2.3 because of heterogeneous vehicle types, non-stationary demand, hysteresis during transitions between regimes, and the spatial and temporal averaging inherent in any measurement process. The nonlinearity of (2.5) gives rise to several phenomena that Chapter 2 of [6] discusses and that reappear in the simulation outputs:

- Capacity drops near bottlenecks.
- Shock waves (sharp discontinuities in the traffic state that propagate upstream, against the direction of travel).

- Phantom traffic jams (stop-and-go waves that emerge with no physical obstacle at all, purely deriving from the finite reaction time of drivers) appear as abrupt changes in the edge-level time series of speed and density.

2.3 Dynamic Models of Traffic Flow

To predict the build-up and the dissipation of congestion one needs a dynamic model belonging to the Macroscopic family. For the *First-Order Models* it is worth mentioning the two big families of continuous and discrete models with independent variable respectively continuous and discrete. We will analyse one for each type, which are the most used in the literature, and both descending directly from the variables of the previous Section 2.1.

Then will be described the *Second-Order Models* in detail, which one of them is adopted in Chapter 6.

2.3.1 First-order models: the LWR conservation law

The first and most influential macroscopic model was proposed independently by Lighthill and Whitham [7] in 1955 and by Richards [11], known as the *LWR* model. Its single assumption is that drivers adjust their speed instantaneously to the equilibrium value dictated by the local density through the fundamental diagram:

$$v(x,t) = V\big(\rho(x,t)\big) \tag{2.7}$$

where $V(\cdot)$ is the equilibrium speed-density function of the fundamental diagram (e.g. the Greenshields law (2.4)), i.e. the steady-state speed a driver adopts at the local density ρ, and hence $q = Q(\rho)$. The LWR model presents some approximations that can lead to unrealistic results, namely it does not contain any inertial effects, since it assumes that vehicles adjust their speeds instantaneously, so we achieve inaccurate accelerations with high jerk values. Yet it systematically predicts the output flow of a congested area equal to the capacity flow (even if the portion is not fully congested), in contrast with the capacity drop phenomenon observed in real-world traffic networks [6].

The physical principle that vehicles are neither created nor destroyed along a road without entering or exiting is applied in the conservation equation

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial Q\big(\rho(x,t)\big)}{\partial x} = 0, \tag{2.8}$$

a first-order, hyperbolic partial differential equation in the single state variable ρ. It's a first-order model in the sense that it captures the dynamics of a single variable e.g. the traffic density.

Because the flux $Q(\rho)$ is non-linear (the parabola of the fundamental diagram), the LWR equation spontaneously develops discontinuities. These are the *shock waves* that travel along a congested road ([7], [11] where the wave theory is applied and the propagation of kinematic waves is discussed in detail), with the analytical counterpart of the stop-and-go bands visible in a space-time diagram.

2.3.2 The Cell Transmission Model

The most widely used discretisation of LWR is the Cell Transmission Model (CTM) of Daganzo [12], sketched in Figure 2.4. The road is divided into cells, each carrying a density $\rho_i(k)$, and the flow between two cells is the minimum of the upstream *demand* D_i and the downstream *supply* S_{i+1}. Thus the CTM computes the flow between two adjacent cells as the minimum of:

- $D_i(k)$: it represents how much cell i would like to send,
- $S_{i+1}(k)$: telling how much the downstream cell can accept.

Therefore the flow is gained as:

$$\Phi_i(k) = \min\{\, D_i(k),\ S_{i+1}(k) \,\}. \tag{2.9}$$

When a downstream cell saturates, its supply S collapses, throttling the upstream flow and pushing a queue against the direction of travel. This behaviour is what makes congestion propagate backwards, following the $min\,\{\, demand, supply \,\}$ rule.

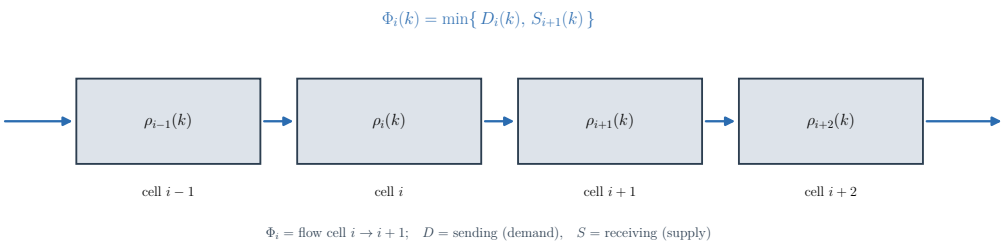


Figure 2.4: The Cell Transmission Model. The road is divided into cells, each carrying a density $\rho_i(k)$, and the flow between two cells is the minimum of the upstream demand D_i and the downstream supply S_{i+1} [12].

The CTM is provably consistent with the LWR hydrodynamic theory and never produces unphysical states such as negative densities or densities above the jam value [12]. That's why it is the prediction model of choice in many freeway MPC schemes, including the event-triggered scheme in [6].

2.3.3 Second-order models

Following the analogy with the previous section, in their continuous form, second-order models are built by resemblance with the equations of fluid dynamics, treating the vehicle stream as a compressible one-dimensional fluid [6]. They maintain the conservation law Equation 2.8 of the first-order theory and couple it with a second partial differential equation that governs the evolution of the mean speed. In the classical formulation of Payne and Whitham [13], [14] the two equations read

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho v)}{\partial x} = 0, \tag{2.10}$$

$$\frac{\partial v}{\partial t} + \underbrace{v \frac{\partial v}{\partial x}}_{\text{convection}} = \underbrace{\frac{1}{\tau}\big(V_e(\rho) - v\big)}_{\text{relaxation}} - \underbrace{\frac{c_0^2}{\rho} \frac{\partial \rho}{\partial x}}_{\text{anticipation}}, \tag{2.11}$$

composed by:

- a *relaxation* term that drives the speed toward the equilibrium value $V_e(\rho)$ over a characteristic time τ;
- a *convection* term by which each vehicle carries its own speed downstream;
- and an *anticipation* (or traffic-pressure) term, with c_0 the anticipation constant, that makes drivers slow down when the density grows ahead of them.

The first-order LWR model of Section 2.3.1 is recovered imposing $\tau \to 0$, which forces Equation 2.7 instantaneously and collapses the speed back onto the fundamental diagram.

Because (2.11) can lead to non physical solution, the Payne–Whitham (PW) model may in some regimes predict vehicles being pushed backwards. This observation was raised by Daganzo [15], where the anisotropic models of Aw, Rascle and Zhang (ARZ) [16], [17] were later introduced to remove this artefact.

For the control-oriented use made of the model, these refinements are secondary. So following [6] we adopt the discretised Payne–Whitham model in

the form that has become standard in freeway traffic control, providing the continuous equations into their discretisation in space and time.

The partial derivatives are replaced by finite differences:

$$\partial v/\partial t \approx \big(v_i(k+1) - v_i(k)\big)/T, \tag{2.12}$$

and

$$v\,\partial v/\partial x \approx (v_i/L_i)\,(v_{i-1} - v_i). \tag{2.13}$$

Applying this scheme to (2.10)–(2.11) produces the model used throughout this thesis, the METANET [18].

The road is discretised into segments $i = 1, \dots, M$ of length L_i $[km]$ with λ_i lanes, each carrying a density ρ_i and a mean speed v_i, updated with sampling time T. The first equation is again the conservation law of Section 2.3.1, written per segment,

$$\rho_i(k+1) = \rho_i(k) + \frac{T}{L_i\lambda_i}\big(q_{i-1}(k) - q_i(k)\big), \qquad q_i = \lambda_i\,\rho_i\,v_i, \tag{2.14}$$

where q_i is the segment flow. Following the part described in Equation 2.11, the dynamic law for the speed is

$$v_i(k+1) = v_i + \underbrace{\frac{T}{\tau}\big(V_e(\rho_i) - v_i\big)}_{\text{relaxation}} + \underbrace{\frac{T}{L_i}\,v_i\,\big(v_{i-1} - v_i\big)}_{\text{convection}} - \underbrace{\frac{\nu\,T}{\tau\,L_i}\,\frac{\rho_{i+1} - \rho_i}{\rho_i + \kappa}}_{\text{anticipation}}. \tag{2.15}$$

The equilibrium speed is the static speed-density law (here the exponential form of Papageorgiou [19], [20] rather than the linear Greenshields one),

$$V_e(\rho) = v_{\text{free}}\,\exp\Big(-\tfrac{1}{a}\,(\rho/\rho_{\text{crit}})^a\Big), \tag{2.16}$$

with v_{free} the free-flow speed, ρ_{crit} the critical density and a a shape exponent. Compared with the first-order LWR model, METANET reproduces phenomena that LWR cannot, such as the capacity drop at active bottlenecks and finite accelerations [6].

The parameters used are collected in Table 2.2. Table 2.3 places the three modelling families side by side.

The background traffic in this thesis is generated by the microscopic simulator SUMO (Section 2.4), the macroscopic models are used as *predictive* models inside the velocity references. The first-order LWR and CTM framework is the

Table 2.2: METANET parameters used in this thesis (Papageorgiou's standard values, lightly adapted to the urban Nave network of Chapter 3).

Parameter	**Symbol**	**Value**	**Meaning**
Critical density	ρ_{crit}	33.5 veh/km/lane	density at which flow is maximal
Jam density	$\rho_{\max}$	180 veh/km/lane	bumper-to-bumper density
Relaxation time	τ	18 s	how fast speed relaxes to V_e
Anticipation const.	ν	35 km^2/h	strength of the downstream-density reaction
Density offset	κ	13 veh/km/lane	regularises the anticipation term
Speed exponent	a	1.867	shape of the equilibrium law V_e
Vehicle occupancy	δ	7 m	effective space taken by one vehicle

Table 2.3: The three modelling families of traffic flow theory.

Family	**State variable(s)**	**What it captures**	**Example**
Microscopic	per-vehicle position and speed	individual car-following and lane-changing	Krauss (SUMO)
First-order macroscopic	density $\rho(x,t)$	shock waves, backward queue propagation	LWR, CTM
Second-order macroscopic	density ρ and speed v	also capacity drop, finite acceleration	METANET

language in which the freeway MPC of [6] expresses the traffic it anticipates. The upper-layer planner of [21] instead relies on a store-and-forward urban model in the tradition of the TUC strategy [22], in which each link carries a vehicle count updated by a conservation equation.

What is left is the analysis of the Microscopic environment SUMO, developed in the following Section.

2.4 SUMO as a Simulation Platform

All the experiments in this thesis are built on SUMO (Simulation of Urban MObility), the open-source microscopic traffic simulator developed by the German Aerospace Center and now maintained under the Eclipse Foundation [23], [24].

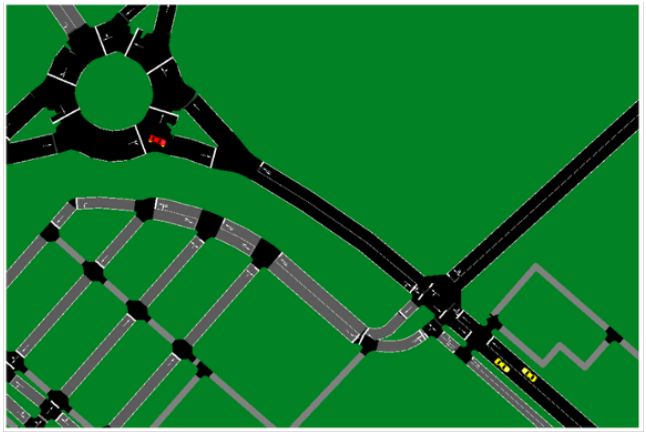


Figure 2.5: The SUMO simulation environment used in this work. The platform provides a microscopic view of the road network, vehicles and traffic interactions, while exposing the same simulated world to external Python controllers through TraCI.

The simulation advances in fixed time steps (1.0 s in the validation phases, and 0.5 s once real-time control is introduced) and at each step every vehicle is moved independently through a *car-following* model [25]. SUMO is an ecosystem of tools, and only a subset of them is exercised here.

Building the network. Road networks are specified in SUMO's native `.net.xml` format, which can be generated directly from OpenStreetMap [26] data with the bundled `netconvert` utility. Vehicle demand is expressed through route files (`.rou.xml`) and the simulation is configured through a `.sumocfg` file.

Generating the demand. The background flow is produced with the helper script `randomTrips.py`, which samples random origin → destination pairs and emits one vehicle every few seconds, creating the congestion against which the controlled vehicle must operate.

The controlled (EGO) vehicle and any special vehicles such as the emergency vehicles equipped with SUMO's `device.bluelight` (considered only in the

early phases of this work) are instead defined explicitly with parameters such as maximum speed, acceleration and so on.

The car-following model. The default longitudinal behaviour in SUMO is the collision-free car-following model of Krauss [25], which at each step every vehicle computes a *safe* speed that still allows it to stop safely,

$$v_{\text{safe}} = v_l + \frac{g - v_l\,\tau}{\dfrac{v_l + v}{2b} + \tau}, \tag{2.17}$$

with v_l the leader speed, v the follower speed, g the gap, τ the reaction time and b the comfortable deceleration. The applied speed is then bounded by the desired and the maximum speed and perturbed by a controlled amount of randomness,

$$v(t + \Delta t) = \max\Big\{0,\ \min\big(v_{\text{safe}},\, v + a\,\Delta t,\, v_{\max}\big) - \varepsilon\, a\, \eta\Big\}, \qquad \eta \sim \mathcal{U}[0,1],$$

where ε is the *driver imperfection* (the σ parameter used later) and η a uniform random number. Aggregating these individual trajectories over an edge recovers exactly the macroscopic q, ρ and v of Section 2.1.

Recording the outputs. In order to analyse the output values of the simulation, SUMO provides the user some files with different levels of detail:

- the *floating-car data* (FCD) output logs record the position, speed and acceleration of every vehicle at each step;
- the *edge-data* output provides the aggregated statistics (flow, density, mean speed) over configurable time intervals that are compared with the fundamental diagram.

The TraCI Interface. By default SUMO run is open-loop, so routes and speeds are fixed before the simulation starts. Using the SUMO feature TraCI, the Traffic Control Interface [27] opens a link between the running simulation and an external Python script, as described in Figure 2.6, so that at every step the script can both *read* the live state of the network (for example the mean speed of an edge via `traci.edge.getLastStepMeanSpeed()` or the gap to the leader via `traci.vehicle.getLeader()`, Chapter 3).

Also *write* commands that take effect immediately (e.g. re-computing a route with `traci.simulation.findRoute()`, Section 4.2, or overriding the EGO speed with `traci.vehicle.setSpeed()`, Chapter 4). This "read → decide → act" cycle is later used for the guidance system developed in Chapter 4.

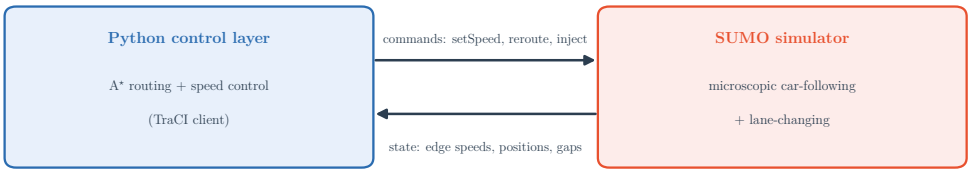


Figure 2.6: The closed-loop architecture used throughout the thesis. At each step the Python control layer reads the traffic state from SUMO through TraCI and writes back routing and speed commands, turning an open-loop simulator into a feedback control system. The image represents the work presented in Chapter 4.

Chapter 3

From Map to Simulation

This chapter represents the early stage of the work when Gianluca Broglia, Davide Faiola, and I started approaching this problem and familiarising ourselves with the applications explained in Chapter 2.

These early stages began by choosing the map and distributing the traffic, as explained before, using the SUMO platform; so we downloaded from the Internet the `.osm` file by OpenStreetMap [26], which contains the road network of the Nave, the area around our Engineering University's Campus. The decision to focus on this specific area was taken because it is the kind of medium-density urban environment (a mix of residential streets, a main arterial, access to the A54 motorway and the presence of the San Matteo Hospital) in which a person would realistically need to travel. So, it's complex enough to produce interesting traffic dynamics, yet compact enough for the simulation to run in a reasonable time and for the outputs to remain interpretable. Also, let me say, for a romantic choice, representing the ending of this beautiful journey of studies.

The purpose of the two phases described in this chapter is to validate the simulation environment and to establish a reproducible, well-structured pipeline that the later phases can be worked on. So the emphasis is on what was set up and then used as a base to develop the full work, starting by obtaining the output diagrams described in Section 2.2.

Then it moves describing the topological validation of a graph constructed over the Nave map and certifies the correctly converted network, a necessary feature for the adaptive routing implementation of Algorithm 1.

The more complex control-driven results (edge time series, space-time diagrams, rerouting histories) begin in Chapter 4, where the infrastructure built here is exploited with a closed-loop controller.

3.1 The First Scenario

The Nave network was obtained, as already said, by exporting it from OpenStreetMap[26] and converting the `.osm` file to SUMO's native `.net.xml` format with `netconvert`:

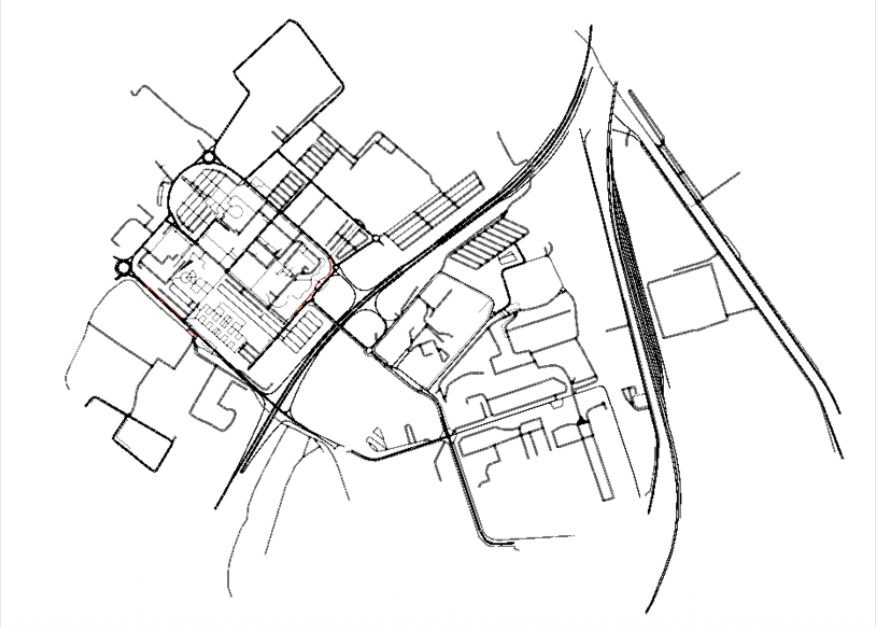


Figure 3.1: Aerial view of the Nave area selected for the first simulation scenario. The map corresponds to the surroundings of our University and was chosen because it offers a compact but realistic urban network with intersections, curved roads and multiple feasible routes for the vehicles.

```
netconvert --osm-files nave.osm \
           --output-file nave_cars.net.xml \
           --geometry.remove --roundabouts.guess \
           --ramps.guess --junctions.join \
           --remove-edges.by-vclass pedestrian,bicycle
```

Listing 3.1: Network conversion command.

Each flag removes a specific artefact from the raw OSM data, as summarised in Table 3.1.

The resulting network comprises **979 nodes** and **1,827 directed edges**, therefore a graph of manageable size but sufficient topological diversity, as promised sufficient for our purposes.

Table 3.1: The `netconvert` flags used to clean the raw OpenStreetMap export into a car-only SUMO network.

Flag	Effect
`-geometry.remove`	collapses redundant intermediate nodes, keeping the graph compact
`-roundabouts.guess`	detects and reconstructs roundabouts from the geometry
`-ramps.guess`	rebuilds motorway on- and off-ramps
`-junctions.join`	merges closely spaced nodes that represent one physical junction
`-remove-edges.by-vclass pedestrian,bicycle`	drops footways and cycleways, leaving a car-only network

Multi-class traffic and the EGO vehicle. The simulation was run for one hour (3600 s) with a step length of 1.0 s. Three vehicle classes populate the scenario, summarised in Table 3.2. The *background flow* (roughly 1,800 passenger cars generated with a departure period of 2 s and a fringe factor of 5) provides an approximation of the real traffic in the area. The *EGO vehicle* is a single controlled car with a reduced maximum speed (13.9 m/s), a value chosen to approximate a conservative kinematic profile of a patient who cannot tolerate abrupt motion. A small driver imperfection $\sigma = 0.3$ introduces a mild stochastic component in the Krauss car-following model of Section 2.4. The last are twelve *emergency vehicles* equipped with SUMO's blue-light device (`device.bluelight`) depart every 300 s from fixed locations, introducing the high-priority interactions with the traffic, simulating the activity of the near Hospital San Matteo.

Table 3.2: The three vehicle classes of the scenario.

Class	Count	$v_{\max}$[m/s]	Role and notes
Background	∼1,800	road limit	random origin-destination flow, period 2 s, fringe factor 5
EGO	1	13.9	controlled patient car, $a_{\max} = 1.2\,\mathrm{m/s^2}$, $\sigma = 0.3$
Emergency	12	road limit	`device.bluelight`, departs every 300 s, high priority

These three classes represent a decent starting point to extract some outputs and display them in the following plots.

From microscopic trajectories to macroscopic diagrams. Floating Car Data (FCD) and 60 s-aggregated edge statistics (`edgeData` output) were collected for post-processing in Python. Over a measurement region of length L (one edge) and duration ΔT (one aggregation interval), the macroscopic variables are recovered from the individual trajectories through the generalised definitions

$$q = \frac{\sum_i d_i}{L\,\Delta T}, \qquad \rho = \frac{\sum_i t_i}{L\,\Delta T}, \qquad v = \frac{\sum_i d_i}{\sum_i t_i} = \frac{q}{\rho}, \tag{3.1}$$

where d_i and t_i are the distance travelled and the time spent inside the region by vehicle i. The identity of (2.3) is therefore satisfied by construction, and the space-mean speed of (2.2) recalls here. The EGO vehicle's acceleration, which SUMO does not export directly, was recovered from its speed trace by finite difference,

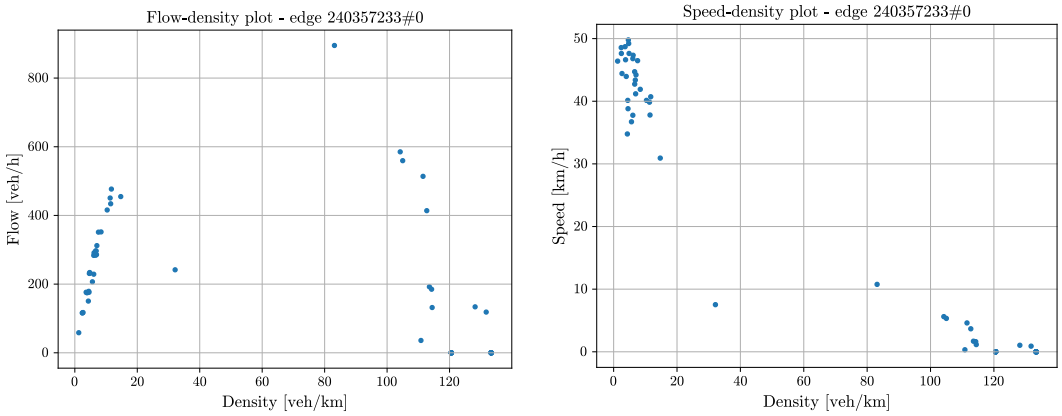
$$a(t_k) = \frac{v(t_k) - v(t_{k-1})}{\Delta t}, \qquad \Delta t = 1\,\text{s}. \tag{3.2}$$

Figure 3.2 plots the flow-density and speed-density diagrams for the busiest edges over the full hour. Figure 3.3 compares these results with the Greenshields parabola of Chapter 2, rising to a capacity peak and then falling away, while the speed-density relationship decreases monotonically as predicted by (2.4). Enlarging the map results in a more distributed density over the x-axes, guaranteeing more available edges for the vehicles (Section 5.4).

The EGO vehicle's own kinematics (Figure 3.4) provide that the speed profile stays within the conservative envelope imposed by its vehicle type. Also the finite-difference acceleration of (3.2) remains small, since cannot be controlled, for this stage the only possible imposition is on velocity as said. The same edge can also be read in time rather than in the flow-density plane.

Figure 3.5 tracks its flow, density and mean speed over the full hour just as described in Section 2.2. Density grows as the network starts to be populated by cars and the mean speed obviously decreases in the exact moment in time. The space-time diagram (Figure 3.6) shows the two periods of free-flow and congestion during the hour.

Every marker in Figure 3.3 corresponds to one `edgeData` aggregation interval (one measurement every 60 s) on the busy edge, so each point is an independent sample of the (density, flow, speed) state of the link.



(a) Flow-density diagram. (b) Speed-density diagram.

Figure 3.2: Fundamental traffic diagrams for a busy edge of the Nave network (1 h simulation). The flow-density curve reproduces the Greenshields parabola and the speed-density relationship decreases monotonically, matching the macroscopic theory of Chapter 2.

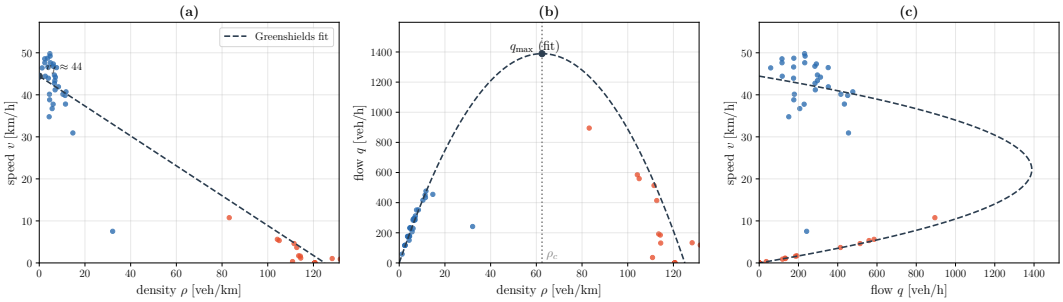
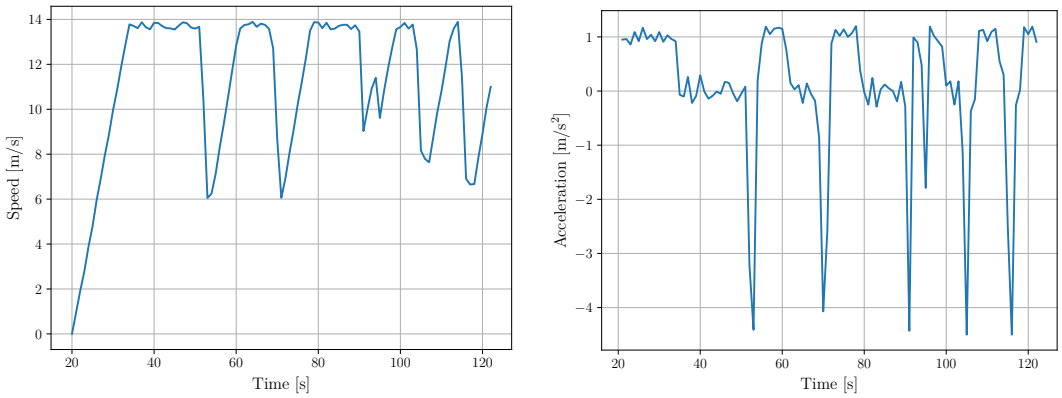
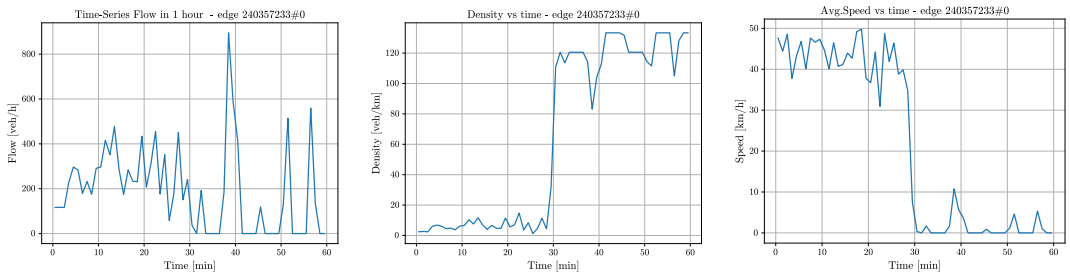


Figure 3.3: Measured fundamental diagram of the simulation (busy edge): (a) speed-density, (b) flow-density, (c) speed-flow. Blue: free-flow branch; red: congested branch.



(a) EGO speed profile [m/s]. (b) EGO acceleration [m/s^2].

Figure 3.4: EGO vehicle kinematic profiles, baseline simulation.



(a) Flow [veh/h]. (b) Density [veh/km]. (c) Mean speed [m/s].

Figure 3.5: Edge-level traffic time series for a busy edge (1 h simulation).

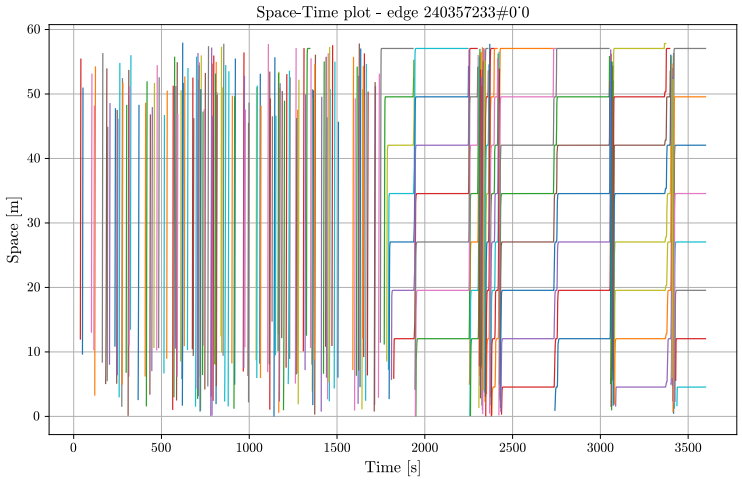


Figure 3.6: Space-time diagram of a busy edge. The diagonal bands are vehicle platoons arising from the periodic departure pattern.

Unlike the theoretical Greenshields diagram, the plot is built from the simulation output, so a fundamental diagram obtained from measurements is therefore intrinsically a cloud of points. The scatter reflects the discrete, time-varying nature of the recorded traffic state.

The two regimes of the model are clearly recognisable in the data, with a low-density, high-speed free-flow branch (blue) and a high-density, low-speed congested branch (red).

Topological validation. An independent Python script (`graph_nave.py`) takes as input the compiled network with `sumolib` and construct over a NetworkX [28] `MultiDiGraph`, verifying a perfect 1:1 correspondence between SUMO's internal representation and the graph object. Both show 979 nodes and all 1,827 edges, giving an exact match. This is a necessary check in order to create solid bases for the rest of the work, because it confirms that `netconvert`

introduced no alternative elements and that the graph used for routing computations is an accurate model of the actual SUMO network Section 4.2.

A mismatch here would mean that a route computed by an external algorithm might reference edges that SUMO does not recognise, causing simulation errors.

3.2 Scripting the Workflow

Until now we produced correct results so it's ready to be optimised, because it relies on manually edited XML files and a fixed sequence of command-line invocations.

So, we replaced that manual procedure with a single Python entry point (`main.py`) that drives the whole simulation pipeline end to end, shown in Figure 3.7.

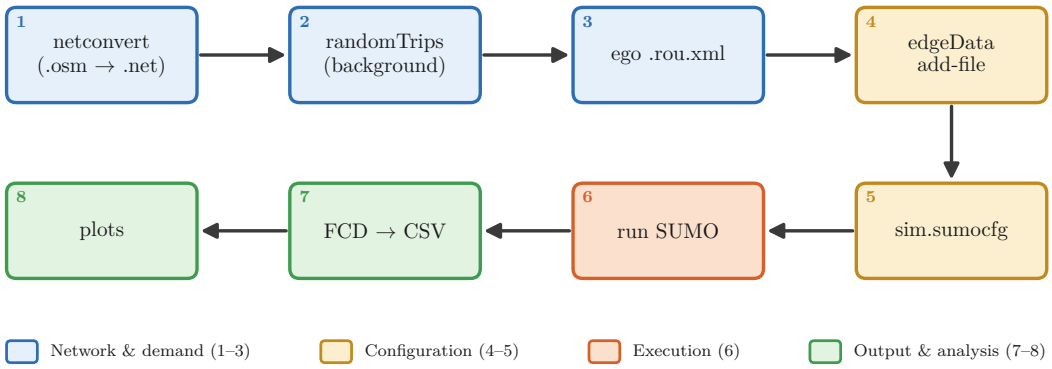


Figure 3.7: The automated eight-step pipeline, each step with its own role, driven end-to-end by a single `main.py` entry point.

On each run the script first calls `netconvert` to build `nave_cars.net.xml`, then generates the background demand with `randomTrips.py` (with a configurable period and fringe factor) and writes the EGO definition (`.rou.xml`) programmatically. It assembles the `edgeData` output configuration and the `sim.sumocfg` that references all the input files, launches SUMO as a subprocess and waits for it to finish, and finally parses the FCD output into a clean CSV and produces every plot in the output directory.

Some parameter choices changed with respect to the previous one used, collected in Table 3.3, e.g. the simulation horizon was reduced to 360 s to focus on the transient traffic build-up phase, the regime most relevant for the following progresses. Also the emergency vehicles were removed to isolate the EGO behaviour in the regular background traffic, and the fringe factor was

raised to 10 which increases the proportion of through-traffic originating at the network boundaries and produces a slightly more congested loading pattern.

Table 3.3: Key simulation parameters in these two phases.

Parameter	**First scenario**	**Automated Pipeline**
Simulation horizon	3600 s	360 s
Step length	1.0 s	1.0 s
Background departure period	2 s	2 s
Fringe factor	5	10
Emergency vehicles	12	0
`edgeData` interval	60 s	60 s
Directed edges	1,827	1,825

By dividing the entire pipeline in a parametric script, every subsequent experiment becomes reproducible with a single command. For example changing the fringe factor or the horizon or the EGO vehicle's parameters no longer requires passing through several XML files, but only changing one or two constants at the top of a Python script. This discipline becomes essential in Chapter 4, where the simulation runs inside a larger TraCI loop and the parameter space to be explored grows considerably.

The topological validation was repeated, again confirming the 1:1 correspondence, this time with 1,825 directed edges (two fewer than the previous simulation, due to the fact that SUMO instantiates a couple of junction connector edges only when `device.bluelight` vehicles are present in the scenario, so removing the emergency vehicles removes those two connectors).

The solid `main.py` file created and explained here is ready to be developed, implemented with all new features introduced in the following Chapters.

Chapter 4

TraCI C.L. approach, A* routing and PID

As anticipated in the previous Chapter, now will be covered the closed-loop approach introducing the TraCI API [27]. It opens a connection between the running SUMO instance and the external Python script seen before, allowing it to query the current state of any edge or vehicle and issue commands that take effect at the very next simulation step. With this *feedback* structure we are able to develop the work done, knowing that the EGO vehicle can react to its environment, then re-compute a better route and let it follow the new one, all within the same simulation tick.

The codebase developed is structured as a Python package with eight modules, where the entry point `main_traci3.py` sequences the SUMO initialisation → background traffic injection → EGO departure with A* routing → simulation loop → post-processing and graph analysis.

Let's cover these implementations step by step.

4.1 Background Traffic and EGO Injection

Following a similar logic of Chapter 3 but with slight differences in order to achieve different results, about 360 background vehicles are injected through TraCI at one vehicle per second over the first 360 s of simulation time. For each vehicle, a random valid origin-destination pair is selected from the set of passenger-accessible edges, and the route is computed on-the-fly by `traci.simulation.findRoute()`. Vehicles that cannot be routed within 1,000 attempts are discarded. Injecting traffic dynamically through TraCI, rather

than loading a pre-generated route file, gives finer control over the loading profile and avoids the need to regenerate the route file every time a parameter changes.

The EGO vehicle departs at $t = 350\,\mathrm{s}$, just as the background traffic is reaching a quasi-stationary congested regime to ensure that the vehicle enters a realistically loaded network, making the routing problem non-trivial. The start edge (`60931876`) and goal edge (`-183789774#0`) are the spatially most-separated feasible pair in the network, chosen to maximise the route length and topological diversity of the trajectory.

4.2 Traffic-Aware A* Routing

After NetworkX [28] `MultiDiGraph` explained before, our Nave map is now represented by a *graph*. A graph is made by **n** nodes $V_1, \ldots, V_n \quad ((V)ertex)$ which are connected by **m** edges $E_1, \ldots, E_m$. In the following example a graph with the set of nodes $\{A, B, C, D, E, F\}$:

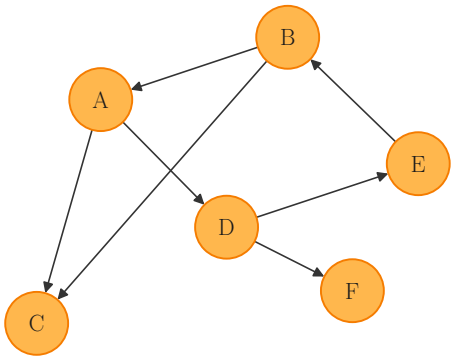
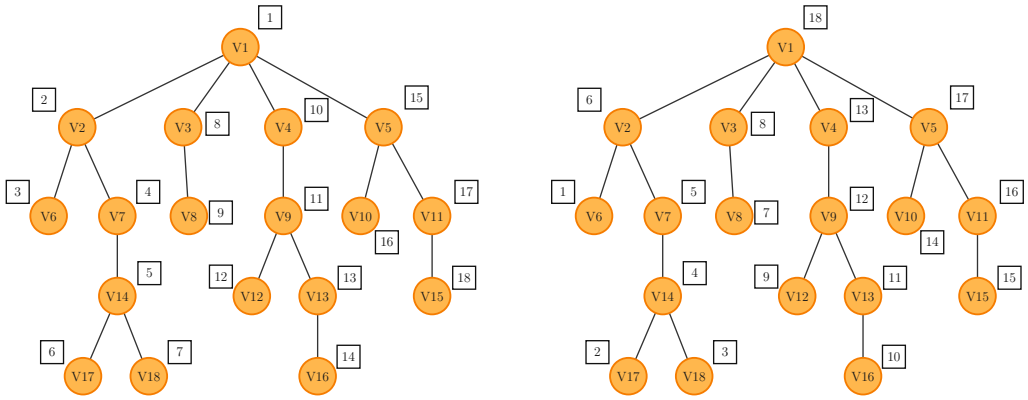


Figure 4.1: Generic graph with 6 nodes and 7 oriented edges.

in this case with arrows, so with *oriented* edges. It is defined also as *cyclic* graph because it presents a path starting from and getting back to the same node going through distinct nodes $\{A \to D \to E \to B \to A\}$. Clearly this is a much simpler example compared with the Nave's graph, but they share exactly the same properties cited.

With the graph we need to perform a visiting task, consisting of examining the nodes of a graph to search for a node associated with the desired information, in order to plan a path over it. The literature presents different methodologies to explore them [29], represented in the following figures as trees for simplicity:

- **Depth-First Search (DFS)** dives as deep as possible along one branch before backtracking, managing the frontier with a LIFO stack. It is memory-light but does *not* guarantee the shortest path and can wander into dead ends.



(a) DFS pre-order.

(b) DFS post-order.

Figure 4.2: DFS representation with the numbers in the squares representing the order of exploration.

- **Breadth-First Search (BFS)** explores the graph level by level with a FIFO queue. It returns the shortest path on an *unweighted* graph, but it ignores edge weights and is therefore blind to congestion.

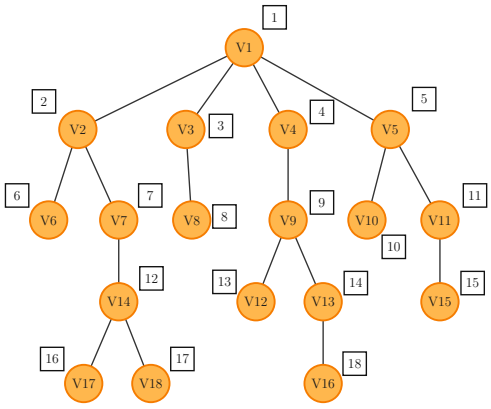


Figure 4.3: BFS represented with the same logic as the Figure 4.2.

- **A*** keeps the optimality of BFS while adding a goal-directed heuristic, which is exactly what is needed here, where every edge is weighted by its length and its live congestion.

A* ranks the open nodes by a score f that splits into a part already paid and a part still estimated:

- $g(v)$: the cost of the best path found so far from the start to v;
- $h(v)$: the heuristic, an estimate of the cost still to go from v to the goal;

getting $f(v) = g(v) + h(v)$ as the estimated total cost of a route that passes through v. At each step the open node with the smallest f is expanded; if the heuristic never overestimates the true remaining cost (it is *admissible*), the returned path is optimal. Figure 4.4 illustrates the bookkeeping on a small weighted graph:

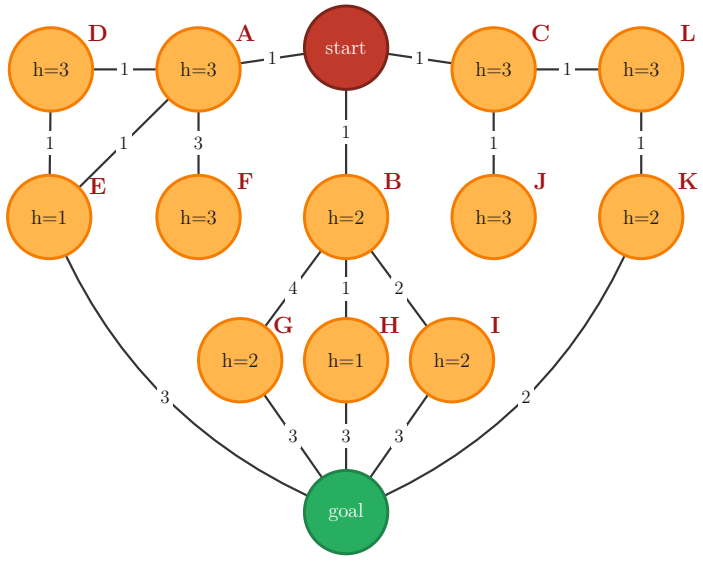


Figure 4.4: A* on a small weighted graph. Each node carries its heuristic value h (the estimated cost to the green goal) and each edge its traversal cost; the search always expands the open node with the smallest $f = g + h$. For node E the cost already paid from the red start is $g(E) = 2$ and the heuristic is $h(E) = 1$, so its priority is $f(E) = 3$.

For the reasons explained, the routing module implements A* that searches over the drivable edges of length $\geq 20\,\mathrm{m}$. In this implementation the heuristic h is the Euclidean distance from the centre of each candidate edge to the goal edge, plausible because a straight line never overestimates the true road distance. The edge traversal cost $c(e)$ penalises congested segments:

$$c(e) = \begin{cases} \lambda(e) & \bar{v}(e) \geq v_{\min}, \\ \lambda(e)\left(1 + \alpha \, \dfrac{v_{\min} - \bar{v}(e)}{v_{\min}}\right) & \bar{v}(e) < v_{\min}, \end{cases} \tag{4.1}$$

where:

- $\lambda(e)$ is the edge length [m],
- $\bar{v}(e)$ is the live mean speed [m/s] acquired from TraCI via `traci.edge.getLastStepMeanSpeed()`,
- $v_{\min} = 4.0\,\mathrm{m/s}$ (about $14\,\mathrm{km/h}$) is the mean speed below which an edge is regarded as congested,
- $\alpha = 4.0$ is the penalty factor, setting how severely such an edge is penalised, since at a near standstill ($\bar{v} \to 0$) the bracket in (4.1) tends to $1 + \alpha = 5$.

Algorithm 1 Traffic-aware A* routing on the edge graph.

Require: start edge e_s, goal edge e_g, edge set $\mathcal{E}$, live mean speeds $\bar{v}(\cdot)$
Ensure: least-cost edge path $\mathcal{P}^\star(e_s, e_g)$

$g(e_s) \leftarrow c(e_s)$; $\mathit{cameFrom} \leftarrow \emptyset$; $\mathit{visited} \leftarrow \emptyset$
$\mathit{open} \leftarrow$ empty priority queue; push $\big(g(e_s) + h(e_s, e_g),\, e_s\big)$ ▷ key $f = g + h$
while $\mathit{open} \neq \emptyset$ **do**
  $(\,\cdot\,, e) \leftarrow$ pop the entry of *open* with the smallest f
  if $e \in \mathit{visited}$ **then**
    continue
  end if
  $\mathit{visited} \leftarrow \mathit{visited} \cup \{e\}$
  if $e = e_g$ **then**
    return RECONSTRUCT(*cameFrom*, e)
  end if
  for all successor edges e' of e **do**
    $\tilde{g} \leftarrow g(e) + c(e')$ ▷ $c(\cdot)$: traffic-aware edge cost, Eq. (4.1)
    if $\tilde{g} < g(e')$ **then** ▷ $g(e') = +\infty$ if unseen
      $\mathit{cameFrom}(e') \leftarrow e$; $g(e') \leftarrow \tilde{g}$
      push $\big(\tilde{g} + h(e', e_g),\, e'\big)$ into *open* ▷ h: Euclidean distance to e_g
    end if
  end for
end while
fail: no route exists from e_s to e_g

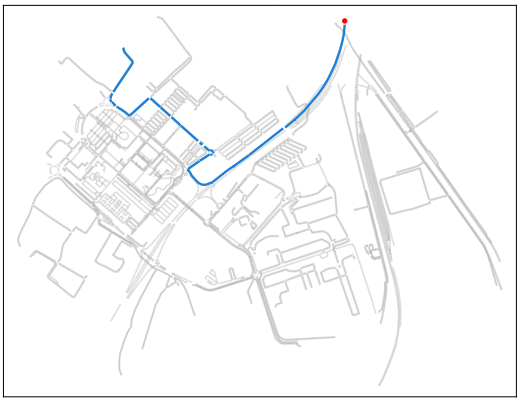
So a jammed edge is treated as five times longer than it really is and the router will accept a replan up to five times its length to avoid it. Thus an edge

operating in the congested branch of the Fundamental Diagram (Section 2.2.1) will have a mean speed well below its free-flow value.

Algorithm 1 represents the logic implemented, with every $T_{\text{reroute}} = 5.0\,\text{s}$, live edge speeds are sampled and the remaining route is recomputed.

Figure 4.5: 1:1 topological graph of the Nave road network reconstructed with NetworkX. Grey: the full network; blue: the edges actually traversed by the EGO vehicle; red: the discarded paths.

During the EGO trip we perform 5 snapshots (Figure 4.6) that show the evolution in time of the evaluated routes (visible in Figure 4.5). They show the EGO vehicle's path (blue), sampled at successive instants of the journey from departure to arrival. The planned trajectory is continually trimmed and, where necessary, redirected as the vehicle advances and the live traffic state changes ahead of it. Thus an implementation on event-driven replanning and not time-driven based is performed in Chapter 5.



(a) Departure, snapshot 1.

(b) Early journey, snapshot 2.

(c) Mid journey, snapshot 3.

(d) Approaching destin., snapshot 4.

(e) Arrival, snapshot 5.

Figure 4.6: Five snapshots at time 350.5 s, 384 s, 422 s, 464 s, 506.5 s of the EGO vehicle's planned route (blue) on the Nave network, from departure to arrival. The route is progressively shortened and re-shaped as the vehicle advances and reacts to the live traffic state.

4.3 PID Speed Control

The speed of the EGO vehicle is regulated by a *Proportional-Integral-Derivative* (PID) controller, by far the most widespread feedback control law in engineering practice thanks to its simplicity, interpretability, and the fact that it requires no explicit model of the plant [30]. A PID controller acts on the *tracking error*

$$e(t) = r(t) - y(t), \tag{4.2}$$

the difference between the reference $r(t)$ (here the desired speed) and the measured output $y(t)$ (the current vehicle speed), and combines three elementary actions into the control signal [30]

$$u(t) = \underbrace{K_p\, e(t)}_{\text{proportional}} + \underbrace{K_i \int_0^t e(\tau)\, d\tau}_{\text{integral}} + \underbrace{K_d\, \frac{de(t)}{dt}}_{\text{derivative}}, \tag{4.3}$$

where K_p, K_i and K_d are the proportional, integral and derivative gains:

- The *proportional* action reacts to the error present at the current instant, providing a correction proportional to how far the vehicle is from the target speed, but on its own it leaves a non-zero steady-state error.
- The *integral* action accumulates the past history of the error and keeps growing as long as any error remains, driving the steady-state error to zero in the presence of constant references or disturbances.
- The *derivative* action responds to the rate of change of the error and thus anticipates its future evolution, adding damping and improving stability. Since differentiation amplifies high-frequency measurement noise, in practice it is applied to a filtered signal and kept small.

Because the controller runs inside the SUMO simulation loop and is evaluated once per simulation step ($\Delta t = 0.5\,\text{s}$), the continuous law (4.3) is implemented in discrete time overriding SUMO's built-in car-following model issuing an explicit speed command via `traci.vehicle.setSpeed()`:

$$u(t_k) = v(t_k) + K_p\, e(t_k) + K_i \sum_{j \le k} e(t_j)\, \Delta t + K_d\, \frac{e(t_k) - e(t_{k-1})}{\Delta t}, \tag{4.4}$$

The gains were selected starting from the classical tuning tables of Ziegler and Nichols [31] and then refined empirically; a complete collection of tuning rules can be found in the work done by Åström and Hägglund [32].

The resulting scheme, together with the gains adopted in this work ($K_p = 0.07$, $K_i = 0.03$, $K_d = 0.01$), is shown in Figure 4.7.

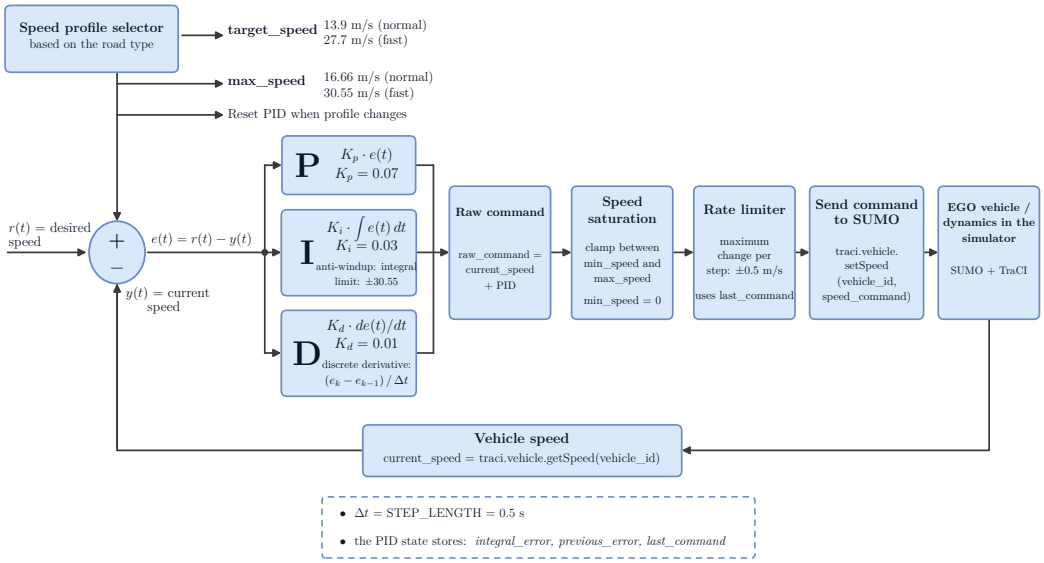


Figure 4.7: PID speed-control loop; it compares the desired speed with the current EGO-vehicle speed, combines proportional, integral and derivative corrections, and sends the resulting command back to SUMO at each simulation step.

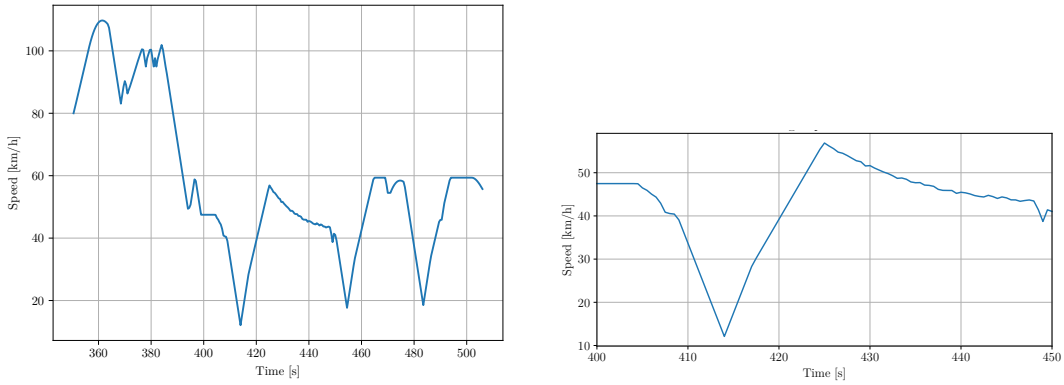
These gains were tuned by hand for a smooth, non-oscillatory response without stressing the hypothetical patient driving with strong accelerations and high variations in time. The *Anti-windup* [30] clamps the integrator so it can never go beyond the threshold, and a rate limiter of $\pm 0.5\,\mathrm{m/s}$ per step bounds the change of the command between two steps, equivalent to adding a limitation on the acceleration, recalling the comfort requirement said earlier.

Two speed profiles are selected automatically based on the road type of the current edge, with the reset of the integrator to prevent overshoot at road-type transitions:

- *normal* on urban roads ($v^* = 13.9\,\mathrm{m/s} \approx 50\,\mathrm{km/h}$, $v_{\max} = 16.66\,\mathrm{m/s} \approx 60\,\mathrm{km/h}$);
- *fast* on motorway and trunk segments ($v^* = 27.7\,\mathrm{m/s} \approx 100\,\mathrm{km/h}$, $v_{\max} = 30.55\,\mathrm{m/s} \approx 110\,\mathrm{km/h}$).

4.4 Results

After the simulation, the EGO speed profile (Figure 4.8) demonstrates that the PID controller tracks the target speed with limited oscillation. The detail panel over a 50 s window shows the regulation behaviour more clearly.



(a) Full EGO speed profile [km/h].

(b) Detail: $t \in [400, 450]$ s.

Figure 4.8: EGO vehicle speed under PID control. The controller correctly switches target speed at road-type boundaries and maintains relatively smooth behaviour throughout.

This chapter has demonstrated the function of the adaptive routing using A* over the graph-matching Nave map and the EGO car controlling method via PID. The vehicle reaches its destination while maintaining a smooth speed profile and avoiding congested edges.

Although this feature can be further developed, the router doesn't know anything about the patient sitting inside. The routing cost function (4.1) treats all roads equally as long as they are not congested. The following Chapter will address road curvature, distance to the following car, cognitive demand of the patient, his health situation, and so on. So the speed controller tracks a fixed target without any awareness of whether that target is comfortable or stressful for the patient on a given day. Addressing precisely these limitations is the subject of Chapter 5.

Chapter 5

Health, Safety, Complexity and Robustness analysis

The following is the first real approach aimed at understanding the health situation of the patient willing to drive. Every patient with different pathologies (e.g. those described in Chapter 1) will have different requests and different symptoms, with consequently different levels of impairment.

Through the *questionnaire*, before the simulation starts, the patient will be asked to answer some clinical questions, and the answers are translated directly into guidance parameters. For example, a patient reporting severe dizziness gets a route that avoids winding roads or who is highly fatigued has motorway access restricted. All these different cases will be discussed and covered below. Someone who is alert and comfortable drives essentially the same route that would have been computed before. The major evolutions introduced in this Chapter are the health-adaptive parameter derivation, patient-mode routing with topology-based edge penalties, and event-driven replanning that reduces computational complexity compared to the time-based replanning. Thus in the end a separate validation run on a larger map confirms that the system's behaviour is consistent when the road network is scaled up.

5.1 The Health Questionnaire and Risk Model

A nine-question assessment is presented either through a graphical `tkinter` window or a terminal prompt; each one targets a specific clinical dimension relevant to safe driving for a patient recovering, i.e. from a debilitating condition, fatigue, dizziness, pain and so on. Answers are given on a 1→5 rank scale.

Table 5.1: Health questionnaire: questions, scoring direction, and weight. *Direct*: higher answer → higher impairment. *Inverse*: higher answer → lower impairment. Q7 (urgency) is handled separately and does not enter in the risk score.

	Question	**Scale (1 → 5)**	**Dir.**	**W.**
Q1	How physically comfortable do you feel?	uncomfortable → comfortable	Inv.	1.5
Q2	How tired do you feel?	not tired → extremely tired	Dir.	2.0
Q3	Nausea, dizziness, or malaise?	none → severe	Dir.	2.0
Q4	How anxious about driving?	calm → very anxious	Dir.	1.5
Q5	How clear-headed and focused?	confused → fully focused	Inv.	2.5
Q6	How ready to start driving?	not ready → fully ready	Inv.	2.0
Q7	How urgent is your trip?	no rush → extremely urgent	Urgency	–
Q8	How much physical pain?	no pain → severe pain	Dir.	1.5
Q9	Medication impairment?	not affected → strongly	Dir.	2.0

Risk score. A weighted clinical risk score $r \in [0, 1]$ aggregates the eight scored questions (Q7 is treated separately as said in Table 5.1):

$$r = \frac{\sum_i w_i \cdot \mathrm{imp}(a_i, d_i)}{\sum_i w_i}, \tag{5.1}$$

where $a_i \in \{1, \ldots, 5\}$ is the answer to question i, d_i the direction of the answer and w_i is its weight (last column of Table 5.1). The weights encode the clinical importance of each dimension for safe driving with cognitive clarity (Q5) carrying the largest weight (2.5), followed by fatigue, nausea, readiness and medication (2.0), in the end comfort, anxiety and pain weighted slightly less (1.5). The impairment function maps each answer to a normalised value $\mathrm{imp}(a, d) \in [0, 1]$ (0 for the healthiest, 1 for the most impaired one) through a weighted average form:

$$\mathrm{imp}(a, d) = \begin{cases} (a-1)/4 & \text{direction is } \textit{direct}, \\ (5-a)/4 & \text{direction is } \textit{inverse}, \end{cases} \tag{5.2}$$

where the direction (Table 5.1) records whether a high answer means *more* impairment (direct, e.g. pain) or *less* (inverse, e.g. comfort). Dividing by $\sum_i w_i$

in (5.1) keeps r in $[0, 1]$ behaving as a normalisation. The score is then mapped to one of five impairment levels via fixed thresholds $(0.20, 0.40, 0.60, 0.80)$ that split the $[0, 1]$ range into five equal bands.

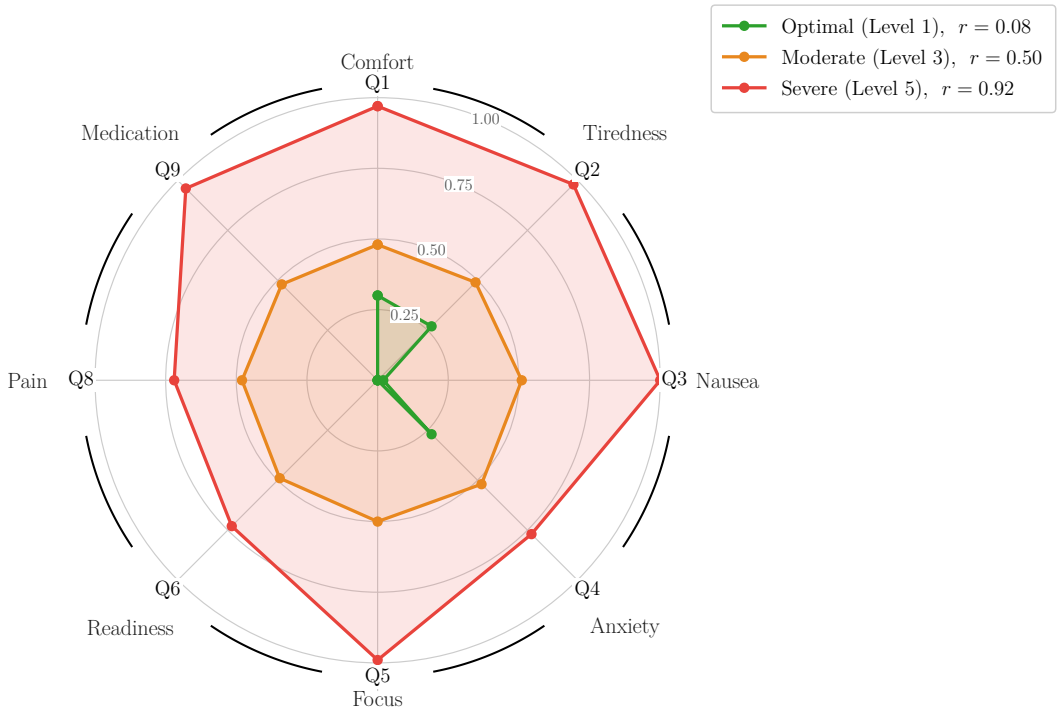


Figure 5.1: Clinical impairment profile derived from the questionnaire, for three example patients. Each axis is one of the eight scored questions; the radius is its impairment imp $\in [0, 1]$. The weighted mean of each polygon, following the risk score (5.1), is shown in the legend together with the resulting impairment level.

A compact way to read a patient is the radar plot of Figure 5.1, which shows the per-question impairment $\text{imp}(a_i, d_i)$ over the eight scored dimensions for three example patients. Each polygon is a clinical profile, and the weighted mean of its vertices, following the Equation (5.1), is the risk score r reported in the legend. The shape, not only the single number r, is what matters, e.g. two patients with the same risk score but different profiles (say, one dominated by nausea, another by fatigue) are routed and driven differently. Table 5.2 reports the baseline values associated with each level.

What actually drives the simulation is the way each individual answer pushes the four guidance parameters in its own direction; Table 5.3 shows, question by question, whether an impairment lengthens the safety distance, tightens the curvature tolerance, raises the minimum acceptable speed limit, excludes a road type, flips the patient-mode switch (explained later, related to

Table 5.2: Impairment levels: risk thresholds and baseline parameter values.

Level	Name	Risk	Safety dist. [m]	Curvature max	Min speed limit
1	Optimal	< 0.20	15	0.50	–
2	Mild	0.20–0.40	25	0.40	–
3	Moderate	0.40–0.60	40	0.25	20 km/h
4	High	0.60–0.80	65	0.15	30 km/h
5	Severe	≥ 0.80	100	0.10	40 km/h

A* routing Section 5.2), or scales the urgency factor. The pattern is clinically traceable because nausea (Q3) is the single strongest lever, acting on almost every parameter at once.

Table 5.3: Parameter influence matrix: the direction and relative strength with which each question's impairment modifies each guidance parameter ("+" increase, "−" decrease; intensity shown by repetition; arrows act on the urgency factor).

Question	Safety dist.	Curvature max	Min speed	Road excl.	Patient mode	Urgency factor
Q1 comfort	+	−	+	–	–	–
Q2 tiredness	+	−	+	motorway	–	↓
Q3 nausea	++	−−	++	urban	✓	↓↓
Q4 anxiety	+	−	+	residential	–	↓
Q5 focus	−−	−	+	all types	✓	↓
Q6 readiness	+	–	–	–	✓	–
Q7 urgency	−	+	−	–	–	↑↑
Q8 pain	+	−	+	–	–	↓
Q9 medication	+	−	+	urban	✓	↓

Question Q7 (urgency) initialises a base urgency factor $u \in [1.0, 4.0]$ (via the look-up $1 \to 1.0$, $2 \to 1.5$, $3 \to 2.0$, $4 \to 3.0$, $5 \to 4.0$) that is subsequently reduced by the impairments from Q2, Q3, Q4, Q8, and Q9, because rushing while clinically impaired is unsafe. When Q7 ≥ 3, the patient may also specify an arrival deadline, which activates a time-urgency mechanism in the router (Section 5.2).

5.2 Patient-Adaptive Routing and Speed Control

The purely traffic-aware cost of A* planner introduced in Section 4.2 (Algorithm 1) is extended with a set of clinical penalties, so that a road which is both congested and medically unsuitable is doubly discouraged. The rest of this section describes those penalties, the safety layer added on top of the speed controller, and the event-driven replanning that decides when the planner is re-run.

Edge penalties. When the *patient mode* flag is activated (impairment level ≥ 3, or notable dizziness, medication, or focus impairment), three categories of edge penalties are pre-computed once at startup and stored in a lookup dictionary:

- **Curvature penalty.** For each edge, a geometric curvature index is computed from the polyline shape:

$$\kappa(e) = \frac{L_{\text{poly}}(e)}{L_{\text{straight}}(e)} - 1,$$

  where L_{poly} is the total arc length and L_{straight} is the straight-line distance between endpoints. A value of 0 is a perfectly straight road; values above 0.20 indicate substantially curved segments. Edges with $\kappa > \kappa_{\max}$ (the maximum tolerated curvature, derived mainly from Q3 and Q5 and tabulated per level in Table 5.2) receive a multiplicative cost factor of 8, that is, for routing purposes, an edge treated as eight times longer than it physically is, strongly discouraging it.

- **Low speed-limit penalty.** Roads whose posted limit falls below a threshold $v_{\text{lim,min}}$ (i.e. derived from Q3, Q5, Q4) receive a factor of 6 (treated as six times longer), directing the router away from slow residential streets where cognitive demand is highest.

- **Road-type exclusion.** Motorways are rendered effectively impassable by a near-infinite factor of 999 when the patient is both very tired and poorly focused. On the other living zones are excluded under severe dizziness or strong medication effects.

These penalties are multiplicative with the dynamic congestion cost of (4.1),

extended to include a patient factor $p(e)$:

$$c(e) = \begin{cases} \lambda(e) \cdot p(e) & \bar{v}(e) \geq v_{\min}, \\ \lambda(e)\left(1 + \alpha \, \dfrac{v_{\min} - \bar{v}(e)}{v_{\min}}\right) \cdot p(e) & \bar{v}(e) < v_{\min}. \end{cases} \tag{5.3}$$

The value $p(e)$ is pre computed, so this "term" related to the congestion cost is paid once at startup. Table 5.4 summarises the three penalty families and the questionnaire items from which each threshold is derived.

Table 5.4: Summary of the patient mode penalties, enabled when the impairment level is ≥ 3, or when dizziness, medication or focus impairment are notable.

Penalty	**Trigger condition**	**Cost factor**	**Driven by**
Curvature	$\kappa(e) > \kappa_{\max}$	$\times 8$	Q3, Q5
Low speed limit	$v_{\lim}(e) < v_{\lim,\min}$	$\times 6$	Q3, Q4, Q5
Motorway exclusion	high tiredness and poor focus	$\times 999$	Q2, Q5
Living-zone exclusion	severe dizziness or medication	$\times 999$	Q3, Q9

Braking Law and Time-Urgency in routing. The PID controller introduced in the previous Chapter is modified with a new safety layer; at each step, the gap to the leading vehicle within a 150 m scan radius is measured via `traci.vehicle.getLeader()`. If the gap g falls below the questionnaire-derived *safe following distance* d_{safe}, the PID output is overridden by a proportional braking law:

$$v_{\text{cmd}} = v_{\text{leader}} \cdot \frac{g}{d_{\text{safe}}},$$

which reduces the vehicle's speed as the gap closes.

When an arrival deadline T_d is set, a *time-urgency* factor $\gamma(t)$ multiplies the travel times passed to SUMO:

$$\gamma(t) = 1 + (u_{\max} - 1) \, \min\left(1, \, \frac{t - T_{\text{dep}}}{T_d - T_{\text{dep}}}\right).$$

The factor ramps linearly from 1.0 at departure toward $u_{\max} = 3.0$ at the deadline, causing the router to increasingly penalise slow edges as time passes.

5.3 Event-Driven Replanning and Results

This new section introduces a complementary strategy that reroutes only when it is actually necessary, compared to the one proposed before every 5.0 s unconditionally. Every 2.5 s, the EGO vehicle's remaining route is scanned for congested edges, defined as edges where $\bar{v}(e) < 0.5\, v_{\text{lim}}(e)$. A `findRoute()` call is issued only when at least one such edge is detected. Algorithm 2 formalises this logic with the cheap scan runs on every cycle, but the expensive planner (Algorithm 1) is invoked only when a jam actually appears on the road ahead.

Algorithm 2 Event-driven replanning.

Require: remaining route $\mathcal{R}$ of the EGO, live mean speeds $\bar{v}(\cdot)$, posted limits $v_{\text{lim}}(\cdot)$, scan period $T_{\text{scan}} = 2.5\,\text{s}$
Ensure: a new route is computed only when the current one runs into congestion

if $t \bmod T_{\text{scan}} \neq 0$ **then**
  return ▷ the scan itself runs only every T_{scan}
end if
congested $\leftarrow \emptyset$
for all edges $e \in \mathcal{R}$ ahead of the EGO **do**
  if $\bar{v}(e) < 0.5\, v_{\text{lim}}(e)$ **then** ▷ edge jammed: speed below half its limit
    congested $\leftarrow$ *congested* $\cup \{e\}$
  end if
end for
if *congested* $= \emptyset$ **then**
  return ▷ no jam ahead: keep the route, no `findRoute` call
end if
$\mathcal{R} \leftarrow$ AstarRoute(current edge, goal, $\bar{v}$) ▷ Algorithm 1
assign the new route $\mathcal{R}$ to the EGO

Table 5.5 shows the outcome over the 145 s EGO trip. The event-driven strategy invoked `findRoute()` exactly 5 times against 30 for the periodic baseline, an approximately 83% reduction in routing calls (considering that the function weight grows proportionally with the number of calls), with no degradation in route quality.

Table 5.5: Periodic vs. event-driven replanning over the EGO vehicle trip.

Method	Check interval	Total checks	findRoute() calls	CPU savings
Periodic	5.0 s	30	30	– (baseline)
Event-driven	2.5 s	64	5	≈ 83%

The five rerouting events occurred at $t = 353.0$, 365.5, 368.0, 380.5, and

460.5 s. The most significant was at $t = 368.0$ s, when the A* router extended the route from 28 to 39 edges to circumvent the exit ramp of the A54 that leads to our Nave University. The mentioned route updates that took place over the journey are collected in Table 5.6, including the progressive shortening as the vehicle approached its destination.

Table 5.6: EGO vehicle rerouting history with Event-Driven approach.

Time [s]	**Edges**	**Note**
350.0	28	Initial A* route
353.0	28	First event reroute
365.5	28	Congestion event: returning to the original A* route
368.0	39	**Major reroute: congestion avoidance (+11 edges)**
375.5	37	Progressive shortening
380.5	36	Event-driven: congestion on remaining route
385.5	33	Still following the same route
390.5	32	
395.5	31	
400.5	30	
420.5	29	Approaching destination
425.5	28	
440.5	20	
460.5	15	Event-driven trigger but no feasible paths available
480.5	8	Final segment
495.5	1	Destination reached

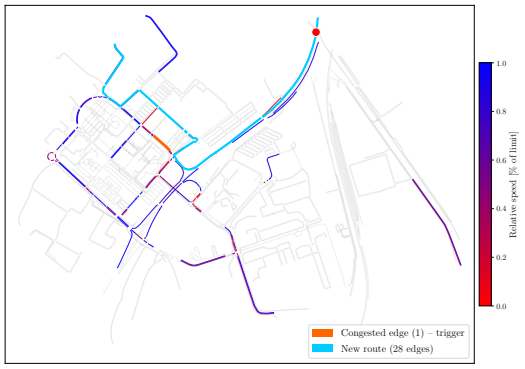
Figure 5.2 captures each of the five event-driven replanning moments on the network itself. In every panel the road graph is coloured by the relative speed of each edge (red where traffic has collapsed to a standstill, blue where it flows freely), the congested edge that *triggered* the replan is highlighted in orange, and the freshly computed route is overlaid in cyan from the EGO position (red dot) to the destination. For each subfigure:

a) For the Event 1 we can appreciate that the straight path of the two red ones in Figure 4.5 (the shortest path to the destination) is congested so a reroute is proposed.

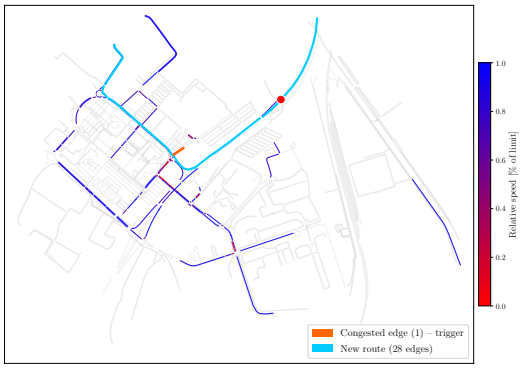
b) In Event 2 the first edge of the change done in Event 1 is congested so is reproposed the original shortest path.

c) In Event 3 we appreciate the **Major reroute: congestion avoidance (+11 edges)** highlighted in Table 5.6.

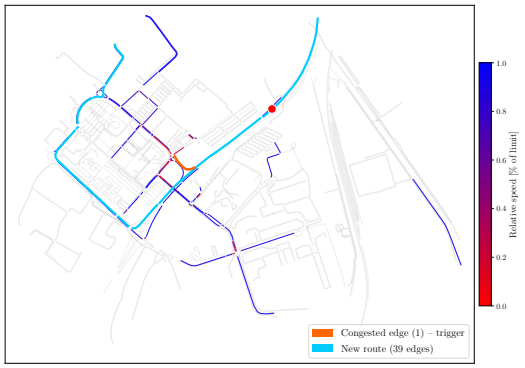
d) In Event 4 something new happens, because in the previous iteration planned path we have a congested link but the EGO car is not replanned



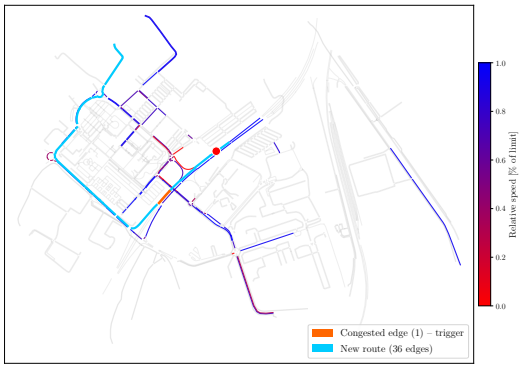
(a) Event 1, $t = 353.0\,\mathrm{s}$.



(b) Event 2, $t = 365.5\,\mathrm{s}$.



(c) Event 3, $t = 368.0\,\mathrm{s}$ (major reroute).



(d) Event 4, $t = 380.5\,\mathrm{s}$.

(e) Event 5, $t = 460.5\,\mathrm{s}$.

Figure 5.2: The five event-driven replanning events over the EGO trip. Edge colour encodes relative speed (red: stopped; blue: free flow); the orange edge is the congestion trigger; the cyan polyline is the newly computed route from the EGO position (red dot) to the destination. These correspond one-to-one to the event-driven rows of Table 5.6.

to turn and exit through the ramp (only possible alternative route in that position of the EGO) because forward from there we have a "more" congested edge (red color) compared to the one in the selected path, so continue in the same route.

e) In Event 5 the last edge is congested, but since is the only possible edge to be routed to arrive at the end point (the end is exactly that edge), so no different path is possible.

Working with a controlled path, in which are known all the possible routes, and with limited roads is the "academic" best way to show the functioning of the algorithms.

The two strategies proposed during this thesis differ not in the cost of a single routing query, which is the same for both, but in how often that query is paid. To see why the event-driven advantage grows with the size of the map, it is worth analysing the asymptotic cost first and then checking it against the measured timings. This all is what justifies the curves of Figures 5.3 and 5.4.

Theoretical cost. A* is Dijkstra's algorithm equipped with an admissible and consistent heuristic, every node is dequeued at most once, so the two algorithms share the worst-case complexity, degenerating to it when the heuristic is uninformative. Thus the heuristic improves the constant factor and the number of expanded nodes, not the asymptotic order. On a graph with N nodes and M edges, and since a road network is sparse (bounded node degree, so $M = O(N)$), a single query therefore costs

$$C_{\text{route}} = O\big((N + M)\log N\big) = O(N \log N). \tag{5.4}$$

The congestion check, by contrast, only scans the edges still on the remaining route. In a planar road network a route spanning a region of N nodes has $O(\sqrt{N})$ edges, so a check costs $C_{\text{check}} = O(\sqrt{N})$, which is asymptotically negligible against a full query. Over an EGO trip of S control steps the two strategies accumulate

$$\begin{aligned} T_{\text{periodic}} &= \frac{S}{k}\, C_{\text{route}} = O\Big(\tfrac{S}{k}\, N \log N\Big), \\ T_{\text{event}} &= \frac{S}{T_e}\, C_{\text{check}} + n_{\text{trig}}\, C_{\text{route}} = O\Big(\tfrac{S}{T_e}\sqrt{N} + n_{\text{trig}}\, N \log N\Big), \end{aligned} \tag{5.5}$$

where k and T_e are the periodic-reroute and event-check intervals (in steps) and n_{trig} is the number of congestion triggers.

Because $n_{\text{trig}} \ll S/k$ and $\sqrt{N} \ll N \log N$, the event-driven cost is dominated by its cheap checks, $T_{\text{event}} = O(\sqrt{N})$ in the sparse-trigger regime. The resulting speed-up therefore *grows* with the network,

$$\frac{T_{\text{periodic}}}{T_{\text{event}}} \propto \frac{T_e}{k}\sqrt{N}\,\log N \qquad (n_{\text{trig}} \ll S/k), \tag{5.6}$$

which is exactly the divergence between the two curves in Figure 5.4.

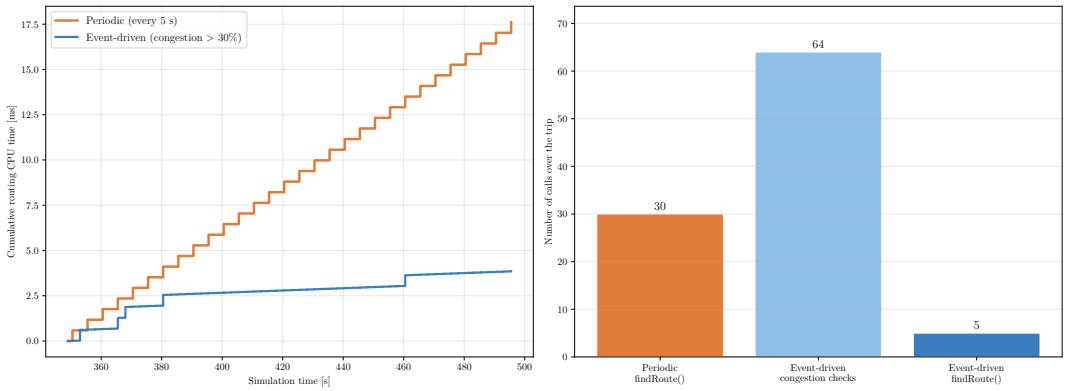


Figure 5.3: Cumulative CPU time: periodic vs. event-driven replanning. The event-driven method reduces routing overhead by $\approx 77.8\%$ with no loss in route quality.

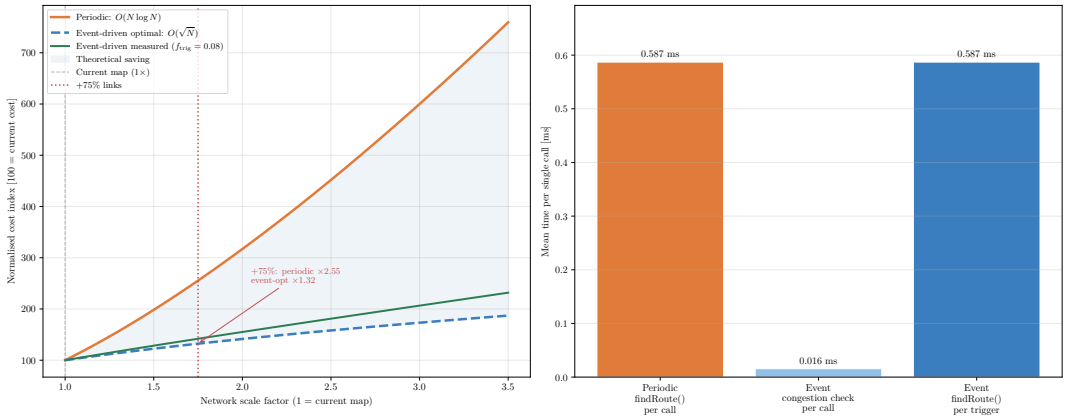


Figure 5.4: Big-O growth curves with 75% of network's increment highlighted.

Measured cost. Table 5.7 compares the model with the timings logged over the 145.5 s EGO trip (`reroute_timing.csv`). A single `findRoute()` call took on average 0.59 ms, while a congestion check is more than an order of magnitude

cheaper. With 30 unconditional periodic calls against only 5 event-driven calls, since the checks are cheap, the saving in routing CPU time is of the same order (around $\approx 77.8\%$ in the measured results, Figure 5.3), with no loss of route quality.

Table 5.7: Per-operation complexity and the cost measured over the 145.5 s EGO trip (`reroute_timing.csv`). N is the network size; on a sparse road graph a Dijkstra query costs $O(N \log N)$ while a route scan costs only $O(\sqrt{N})$.

Operation	**Per-call cost**	**Calls over trip**	**Total CPU**
`findRoute()`, periodic	$O(N \log N)$	30	≈ 17.6 ms
`findRoute()`, event-driven	$O(N \log N)$	5	≈ 2.9 ms
congestion check, event-driven	$O(\sqrt{N})$	64	≈ 1 ms
Routing CPU, periodic			≈ 17.6 ms
Routing CPU, event-driven			≈ 3.9 ms

Projection to a larger map. Extrapolating the normalised model of Figure 5.4 to a network 75% larger ($x = 1.75$), the periodic cost rises by a factor $1.75 \log_2(2.75) \approx 2.55$, whereas the check-dominated event-driven cost rises only by $\sqrt{1.75} \approx 1.32$. The event-driven advantage therefore widens by about $2.55/1.32 \approx 1.9\times$ on the larger map. As expected this new Event-Routing strategy pays off more as the network grows, guaranteeing the best strategy for larger applications, e.g. the one presented in Section 5.4.

5.4 Robustness on a Larger Network

The entire system was re-run on a larger map (`nave_2.osm`), which extends the original area by including the surrounding territory and a greater number of edges, in order to verify the generalisability of the results.

The outputs, fundamental diagrams, EGO speed profiles, rerouting history, and event-driven vs. periodic comparison are qualitatively consistent with the results obtained in the original map. The event-driven strategy again achieves a significant reduction in `findRoute()` calls relative to the periodic baseline, as expected.

The four phases presented in this thesis constitute a coherent progression from a static validation baseline to a closed-loop, health-aware autonomous guidance system:

Figure 5.5: Extended region of our University.

- Phase 1 validated the simulation environment.
- Phase 2 established a reproducible pipeline.
- Phase 3 introduced real-time routing and speed control.
- Phase 4 connected the guidance system to the patient's clinical state and demonstrated that the resulting architecture is both computationally efficient and robust to network changes.

The presented work is now ready for its final implementation, which will provide the optimal reference for the patient, as described in the following Chapter.

Chapter 6

Model Predictive Control with METANET

This is the central Chapter and it enters into the main part of the thesis, describing its final form, focusing on a *Model Predictive Control* (MPC) implementation. The idea is to create a "big" optimisation problem that will provide a consistent reference to our representation of the patient, the EGO car, and let it drive towards it by the internal controller of SUMO [23]. As a final implementation, we need to extract all the concepts analysed in the previous chapters and then condense them in the MPC. So, we will deliver an optimal controller that will choose the shortest path and change it when congestion ahead appears (Section 4.3). Also, that consider the patient's health situation and urgency (Section 5.1), moderating the acceleration, the jerk, the min and max velocity, the min distance, the road exclusion and the max velocity accepted; following the questionnaire described in Chapter 5.

But it goes forward, because it's implemented a METANET (already discussed in Section 2.3.3) model and correctly weight it inside the cost function in order to predict, in the MPC sample, the traffic that the EGO car will approach during its trip.

Clearly, the constraints are not fixed but they are written as functions of the patient's clinical state, so that the "shape" of the optimisation problem changes with the different person sitting in the car.

The Chapter opens with a fairly extended theoretical introduction about the MPC given by Magni and Scattolini in their book on Advanced Control [33] (the same material taught in the Industrial Control course). Then, it will move from a theoretical point of view to a traffic, more realistic scenario with

network-level freeway control of Ferrara, Sacone and Siri. With their work done in Chapter 9, they develop implementation-oriented and event-triggered MPC for freeway systems [6]. Follows it the theoretical background about the *vehicle-level*, speed oriented MPC developed by Sacchi, Basile, Siri, Minetti, Bonfiglio and Ferrara [21], a work carried out within the E-COSMOS project, *"Electric Charging Optimization for Sustainable Mobility Systems"*, and referred to in what follows as the "E-COSMOS scheme".

Thus it describes in detail how that template was adapted, re-weighted, constrained to fit the rehabilitation-driving problem of this thesis. Furthermore, the velocity profiles obtained are analysed and discussed across two traffic scenarios under different conditions.

The Chapter closes with an application outlook, considering the technology available with different solutions; under different protocols.

6.1 Model Predictive Control: State of the Art

Model Predictive Control is not a single algorithm but a family of them, united by one idea: at each sampling instant it uses a model of the process to predict its future behaviour. Then, choose the sequence of control actions that optimises some objective over a finite shifting horizon window; apply only the first component of the control sequence and then repeat the whole computation one step later with new measurements.

This is the strategy that, over the last three decades, has become the most widely used advanced control methodology in the process industries [33].

Its success is due to its principal properties that no classical closed-loop controller offers simultaneously (designed in the Laplace Domain): with the MPC, the control problem is posed as an *optimisation* problem, so that several (possibly competing) goals can be encoded in a single cost function and traded off explicitly. Furthermore, on constraints, the inputs, states, and outputs can be included directly in the problem formulation rather than handled after using different tools such as saturations blocks, or ad-hoc logics. The controller is synthesised from a process model, so we retain all the information without losing any type of generality. For example, if the system is not fully reachable or fully observable, if we perform a Laplace transform and we obtain a transfer function of the general form, we are losing details about the system because the system is no more fully described [33].

The pair (A, B) is *reachable* if and only if the reachability matrix has full

row rank,

$$\mathcal{R} = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix} \in \mathbb{R}^{n \times nm}, \qquad \operatorname{rank} \mathcal{R} = n. \tag{6.1}$$

Dually, the pair (A, C) is *observable* if and only if the observability matrix has full column rank,

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \in \mathbb{R}^{np \times n}, \qquad \operatorname{rank} \mathcal{O} = n. \tag{6.2}$$

In classical control this loss can already be seen by writing the transfer function in canonical forms that emphasise only poles, zeros, and gains. In the **factorised pole-zero form**, the numerator and denominator are decomposed into simple monomial or binomial factors,

$$G(s) = K \frac{\prod_{i=1}^{m} (1 + s\tau_i)}{\prod_{j=1}^{n} (1 + sT_j)}, \tag{6.3}$$

where K is the system gain, τ_i are the system zeros (lead frequencies), and T_j are the system poles (lag frequencies).

Equivalently, the **amplifier or time-constant canonical form** isolates the static gain from the dynamic part by normalising the constant term of both polynomials to one,

$$G(s) = K_p \frac{1 + a_1 s + a_2 s^2 + \cdots}{1 + b_1 s + b_2 s^2 + \cdots}, \tag{6.4}$$

where K_p is the steady-state gain, namely the gain evaluated at $s = 0$.

For a vehicle that follows a speed reference, that if tracked correctly the acceleration and jerk will stay in comfort bounds (and still arriving at its destination), retaining the ability to state all of these requirements as one optimisation problem is the strong part about this theory.

The ingredients. Following our theory book [33], the essential components of any MPC algorithm are the following:

- The process model, usually in discrete time,
- a set of input, output, and state constraints,
- a cost function J defined at the current instant k over a finite prediction horizon $[k,\, k+N)$,
- an optimisation algorithm that computes the future optimal control sequence,
- and the *Receding Horizon* (RH) principle that ties them together.

The RH principle is the core of the method because *at time k, given the available information, one solves the optimisation problem with respect to the future control sequence $[u(k),\, \ldots,\, u(k+N-1)]$ and applies only its first element $u^\circ(k)$. At the next instant $k+1$, a **new** problem is solved over the shifted horizon $[k+1,\, k+N]$, using the information now available, and again only the first element is applied.* The remarkable consequence is that (although a finite-horizon *open-loop* optimisation is solved at every step) the repeated re-solving with new measurements turns the scheme into a **time-invariant feedback control law**.

6.1.1 Unconstrained MPC of linear systems in state space form

Consider a discrete-time linear system

$$x(k+1) = A\,x(k) + B\,u(k), \qquad y(k) = C\,x(k), \tag{6.5}$$

with state $x \in \mathbb{R}^n$ (assumed measurable), control $u \in \mathbb{R}^m$, and output $y \in \mathbb{R}^p$. At each instant k the controller computes the sequence $u(k),\, u(k+1),\, \ldots,\, u(k+N-1)$ minimising the finite-horizon quadratic cost:

$$J\big(x(k), u(\cdot), k\big) = \sum_{i=0}^{N-1} \Big(\|x(k+i)\|_Q^2 + \|u(k+i)\|_R^2 \Big) + \|x(k+N)\|_S^2, \tag{6.6}$$

where $Q = Q^\top \succ 0$, $R = R^\top \succ 0$, and $S = S^\top \succ 0$ weight, respectively, the state deviation, the control effort, and the terminal state (the positive integer

N is the *prediction horizon*). The notation $\|z\|_M^2 = z^\top M z$ denotes the weighted squared norm.

Two solution routes lead to the same controller. In the *closed-loop* route, the problem is recognised as a finite-horizon linear-quadratic (LQ) problem and solved backwards in time via a Riccati recursion, yielding the time-varying gain law $u(k+i) = -K(i)\,x(k+i)$. In the *open-loop* route (the one that generalises to constraints) the predicted states are stacked into a single vector and written as an affine function of the stacked control sequence,

$$X(k) = \bar{A}\,x(k) + \bar{B}\,U(k), \tag{6.7}$$

where $X(k)$ collects $x(k+1), \ldots, x(k+N)$ and $U(k)$ collects $u(k), \ldots, u(k+N-1)$. Substituting (6.7) into the cost (6.6) turns J into a positive definite quadratic form in the single decision vector $U(k)$:

$$U^\circ(k) = -\big(\bar{B}^\top \bar{Q}\,\bar{B} + \bar{R}\big)^{-1} \bar{B}^\top \bar{Q}\,\bar{A}\,x(k), \tag{6.8}$$

with $\bar{Q}$ and $\bar{R}$ the block-diagonal weight matrices. Applying the receding-horizon principle, so keeping only the first block of $U^\circ(k)$, gives the state-feedback law $u^{\mathrm{MPC}}(k) = -K(0)\,x(k)$. In the nominal case, with no disturbances or modelling error, this open-loop construction coincides with the closed-loop LQ solution. They diverge only when uncertainty is present, and even then their *first* elements remain equal [33].

Handling constraints. The whole point of preferring the open-loop construction is to define the constraints in the optimisation problem. Suppose the inputs, states, and outputs must satisfy bounds

$$u_m \le u(k+i) \le u_M, \quad x_m \le x(k+i) \le x_M, \quad y_m \le y(k+i) \le y_M.$$

Stacking these over the horizon turns the optimisation into a constrained quadratic program (QP) [34],

$$\min_{U(k)} \; J\big(x(k), U(k), k\big) \quad \text{s.t.} \quad X_m \le \bar{A}x(k) + \bar{B}U(k) \le X_M, \;\; U_m \le U(k) \le U_M, \tag{6.9}$$

which no longer admits an explicit solution but is routinely solved on-line by quadratic-programming methods. In the real world physical actuators saturate, roads have speed limits, and passengers tolerate only a range of acceleration

and a defined variation in time (described in Section 6.4.2, following [35], [36]). All these are linear inequalities that are represented in the form of (6.9).

Tracking, disturbances, and the control horizon. When the goal is to follow a reference y°, the cost weights the predicted tracking error $y^\circ(k+i) - y(k+i)$ instead of the state. When a constant disturbance acts on the plant, an integral action can be recovered either by estimating the disturbance with an observer or by rewriting the system in *velocity form*, penalising control *increments* δu so that the cost function (6.6)$\to 0$ when the error vanishes and the input settles. So we achieve *tracking and zero steady-state error and disturbances rejection*.

A different approach used in practice is when the control is allowed to vary only over the first N_u steps and is held constant thereafter, reducing the number of optimisation variables and producing a less aggressive action. This method is known as the *control horizon* $N_u \leq N$. The extreme choice $N_u = 1$ is the classic *mean-level control*.

6.2 MPC in Traffic Control: the Freeway-Network Perspective

The previous section presented the MPC as a general-purpose control methodology. Before specialising it to a single car, it is worth recalling that MPC also has a long and specific history within traffic control.

This branch is developed in Chapter 9 "Implementation-Oriented Freeway Traffic Control Strategies" in *Freeway Traffic Modelling and Control* by Ferrara, Sacone and Siri [6]. They dedicated their chapter to try to ask how the MPC for freeway can actually run in real time. The problem regarding its control is comparable to this thesis, but in the book the "controlled" object is not a vehicle; is the *infrastructure*. Although the way they proposed predictive control for a macroscopic traffic-flow model (built on exactly the density, flow and speed variables of Chapter 2) is very similar to the one developed in this Chapter.

They identified some obstacles to implementing the MPC on a real network:

- the Finite-Horizon Optimal Control Problem (FHOCP) is non-linear and high dimensional;
- it must be solved at *every* controller step;

- it requires the full system state to be transmitted to the controller each step.

They then laid out the directions that make MPC tractable with different methodologies for different technologies avaiable:

- Decomposing the problem into local sub-problems (distributed and decentralised MPC).
- Simplifying or reformulating the model (for instance casting the Cell Transmission Model as a mixed-integer linear program that efficient solvers can handle).
- Controlling only when and where necessary: an implementation philosophy that isn't new for this thesis. Also, in the book is cited the "Transmitting" part which supposes the use of sensors and transmitters. Clearly, it is a part that is not touched due to the theoretical and simulation-based approach of this work.

Different methodologies lead to different solutions. *Decentralized and Distributed Model Predictive Control* (DMPC) addresses the problem of controlling a multivariable dynamical process, composed by several interconnected subsystems and subject to constraints. Compared to a centralized MPC setup,

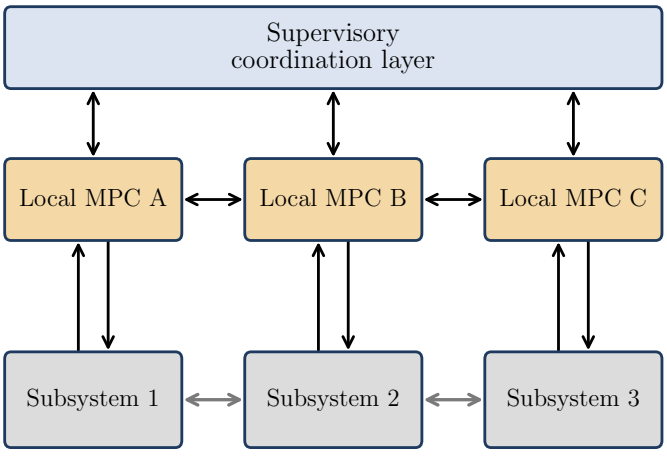


Figure 6.1: Hierarchical and Decentralized/Distributed Model Predictive Control of a large-scale process.

where a global optimal control problem must be solved on-line with respect to all actuator commands (given the entire set of states) [37].

From the picture is clear to understand that in the DMPC technique, the control problem is divided in sub-local MPCs interconnected to each other, communicating informations such as local optimal predictions, local decisions, etc. So, each controller is based on a partial (local) model of the overall dynamics, possibly neglecting existing dynamical interactions.

Generally speaking, distributed control schemes are based on the synergy of local controllers, designed to solve small-scale control problems, which contribute to regulating the overall system, by relying on specific cooperation patterns (from [6], page 236).

Another important part, that can lead to a big issue, which needs to be discussed is *the computational effort*, both in terms of computational complexity of the FHOCP and in terms of number of times in which the control law is computed. A solution adopted is the event-triggered MPC, the subject of Section 9.3 of the book. Rather than re-solving the optimal control problem

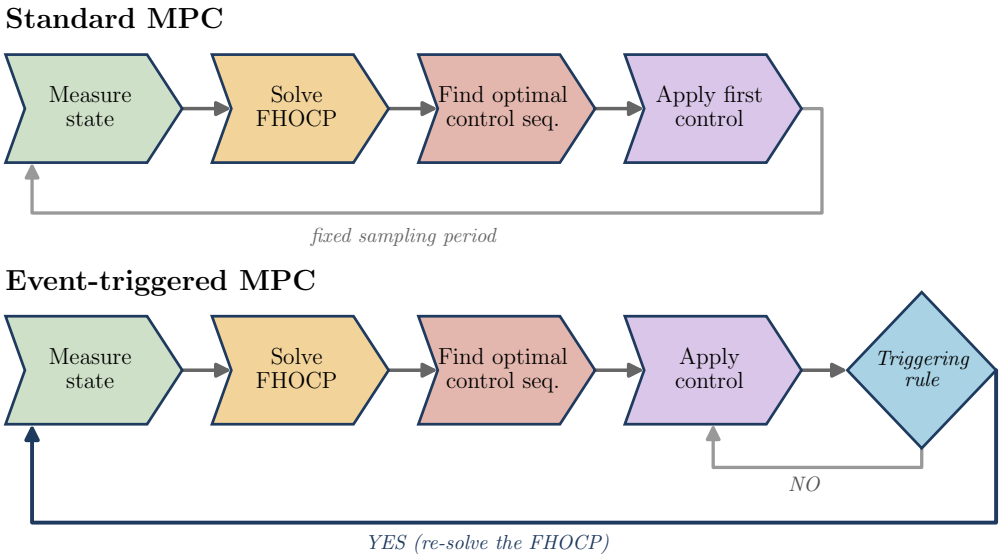


Figure 6.2: Comparison between the standard MPC scheme and the event-triggered MPC scheme.

on a fixed clock, a *triggering rule* is evaluated on the measured state at each step, so the control law is recomputed only at the steps where that rule fires. The parallel with the Event-Driven Replanning of Chapter 5 is straightforward; there, `findRoute()` is invoked only when a congested edge actually materialises on the remaining route, which is nothing other than a triggering rule evaluated on the measured traffic state.

There is also a deeper connection with Chapter 9 of [6] and Chapter 2 because the Macroscopic relations introduced there as descriptive theory reappear *inside* the controller (Section 6.4.1), as the prediction model the optimiser follows.

6.3 From Theory to the Road

The work done by Nikolas Sacchi and the other authors [21], already introduced in Chapter 2 as the methodological reference for this thesis, delivers the application of an MPC with a *shrinking horizon* (the prediction window always ends at the fixed deadline, so it shortens at every solve), applied according to a receding-horizon logic (only the first command is applied, after which the problem is re-optimized). Their architecture is two-layered: an upper layer plans, for each connected autonomous vehicle, a traffic-aware path through the network by solving a resource-constrained shortest-path problem that is periodically refreshed against a macroscopic traffic model. A lower layer then regulates the *speed* of each independent vehicle along that fixed path with a decentralised MPC.

In this thesis the traffic-aware A* router is the upper-layer planner, and the speed reference developed by the MPC is the lower level.

What makes their lower-layer formulation so directly reusable is its choice of variables, where the route is treated as a one-dimensional track, and the vehicle's progress is described by a *normalised position* $p \in [0, 1]$, equal to the distance travelled along the route divided by the route's total length ($p = 0$ at the start and (6.12) at the destination). Over a prediction horizon of N_{MPC} steps, the speed profile u_c is then obtained by solving an open-loop optimal control problem:

$$u_c^\star = \arg\min_{u_c} \sum_{r=0}^{N_{\mathrm{MPC}}-1} \big(1 - p_c(r)\big). \tag{6.10}$$

It's subject to the position dynamics

$$p_c(r+1) = p_c(r) + u_c(r)\, T_{\mathrm{MPC}}/\lambda_c, \tag{6.11}$$

the mobility-energy dynamics $E_{c,\mathrm{mob}}(r+1) = E_{c,\mathrm{mob}}(r) - (\eta_1 u_c(r) + \eta_2 u_c(r)^2)\, T_{\mathrm{MPC}}$ with the residual-charge requirement $E_{c,\mathrm{mob}}(N_{\mathrm{MPC}}) > 0$, the terminal arrival condition

$$p_c(N_{\mathrm{MPC}}) = 1, \tag{6.12}$$

and the speed bound

$$0 \leq u_c(r) \leq \bar{v}_{c,i}^{h}(r) \tag{6.13}$$

set by the admissible speed of the currently occupied link.

The single term of (6.10) penalises the remaining distance pulling the vehicle towards its destination.

Here the progress is the only objective, while the deadline and the battery enter as constraints, also, and more importantly for this thesis, traffic never enters the cost function. Congestion is accounted for entirely upstream, in the resource-constrained shortest path, through the travel times τ_i^h and, at the speed level, only as the upper bound $\bar{v}_c$ of (6.13), itself a decreasing function of the link occupancy.

Focus here is on the patient's situation and not grid services, therefore we will have a different cost, with comfort terms (on acceleration and on jerk) added and every weight and bound is a function of the clinical questionnaire of Chapter 5. In my scenario I cannot empty a macroscopic queue acting on a single vehicle, so the congestion term must be redesigned around its own marginal contribution. Thus the implemented term is a normalised, speed-coupled local penalty (6.28) that acts only when congestion is predicted.

6.4 The MPC Formulation

In the controller, acceleration and jerk become the *augmented states* (derived from the input, the velocity itself (6.16)) and they are used to "shape" and "bound" the patient's comfort. Traffic enters in the cost through a METANET model of the links on the EGO's path, so as said before the speed reference automatically slows down where congestion is predicted. Because the METANET dynamics are nonlinear the optimisation is a nonlinear program (NLP) problem, solved by single-shooting and Sequential Least-Squares Quadratic Programming (SLSQP) [38].

The MPC will produce a speed-reference profile where only its first value is sent to SUMO (the RH principle [33] used also in the E-COSMOS template [21]) via `setSpeed` each step. So that the *SUMO's own car-following model is the inner loop* that tries to reach it with its own equations described in Section 2.4, Eq. (2.17); it will cap it to the collision-safe speed when the traffic does not allow it.

At each control instant the MPC optimises the EGO speed sequence over a

horizon of N steps of length T,

$$V = \big[\, v(0),\, v(1),\, \ldots,\, v(N-1) \,\big]. \tag{6.14}$$

Acceleration and jerk are the first and second differences of this profile (with the measured boundary $v(-1) = v_{\text{prev}}$, $a(-1) = a_{\text{prev}}$),

$$a(r) = \frac{v(r) - v(r-1)}{T}, \tag{6.15}$$

$$j(r) = \frac{a(r) - a(r-1)}{T} = \frac{v(r) - 2v(r-1) + v(r-2)}{T^2}. \tag{6.16}$$

As said before they are augmented states, so they are not chosen independently, but determined by the speed profile, and penalised in the cost and bounded by the constraints exactly as the questionnaire requires.

Progress along the path. The normalised progress along the planned path of total length λ_{tot} evolves, as in the E-COSMOS template (Eq. (6.11)), by

$$p(r+1) = p(r) + \frac{v(r)\,T}{\lambda_{\text{tot}}}, \qquad p \in [0, 1], \tag{6.17}$$

where $p = 1$ means the destination has been reached. It will be explained that the remaining-distance term $(1-p)^2$ in the cost rewards progress, thus it pushes the EGO to move as fast as possible (considering the physical condition of the patient given by the health-situation, the road limits, the traffic and so on).

The METANET prediction model inside the loop. The links of the planned path are discretised into segments and advanced with the second-order METANET model introduced in Section 2.3 (Equations (2.14), (2.16)). Clearly it needs to be described in discrete form (as in the Equations (6.18), ...).

The model works in the classic macroscopic units, so the coarse sampling step M_T enters as $T_h = M_T/3600$ expressed in hours. The variables used are:

- lengths in km, time in h;
- density ρ_i in veh/km/lane;
- mean speed v_i in km/h;
- flow q_i in veh/h.

Segments and state. Each drivable edge of the route becomes one segment i, characterised by:

- length L_i $[km]$ (clamped to a minimum of 0.05 km for tiny edges);
- number of lanes λ_i;
- free-flow speed $v_{\text{free},i}$, read from the SUMO edge;
- cumulative bounds $[s_i, s_i + L_i]$ that map the EGO progress p to the segment it currently occupies.

The state of segment i is the pair $[\rho_i, v_i]$, re-initialised at every solve from the live SUMO measurements. For the state pair we need the per-edge vehicle count n_i that gives the density $\rho_i = n_i/(L_i\lambda_i)$ and the per-edge mean speed that gives v_i (default equal to $v_{\text{free},i}$ on empty edges).

Flow and conservation of vehicles. The flow leaving segment i is the fundamental relation

$$q_i(k) = \lambda_i \, \rho_i(k) \, v_i(k), \tag{6.18}$$

while the density evolves according to the vehicle conservation law, which balances the inflow $q_{i-1}(k)$ from the upstream segment against the outflow $q_i(k)$,

$$\rho_i(k+1) = \rho_i(k) + \frac{T_h}{L_i \, \lambda_i} \big(q_{i-1}(k) - q_i(k)\big). \tag{6.19}$$

Equilibrium speed and dynamics. A traffic stream of density ρ_i relaxes toward an equilibrium speed described by the Papageorgiou law (2.16). Starting from this equilibrium, the mean speed of the segment is updated at each time step:

$$v_i(k+1) = v_i + \underbrace{\frac{T_h}{\tau}\big(V_e(\rho_i) - v_i\big)}_{\text{relaxation}} + \underbrace{\frac{T_h}{L_i} \, v_i \, (v_{i-1} - v_i)}_{\text{convection}} - \underbrace{\frac{\nu \, T_h}{\tau \, L_i} \frac{\rho_{i+1} - \rho_i}{\rho_i + \kappa}}_{\text{anticipation}}. \tag{6.20}$$

The three contributions have distinct physical meanings described in Section 2.3.3.

Boundary conditions. The recursion needs a cell upstream of the first segment and a density downstream of the last. So here raise a problem for the first and last cases of the recursion: these two entities need to be determined and they both use a self-consistent boundary that creates no artificial queue.

- The upstream cell mirrors the first segment, $\rho_{-1} = \rho_0$ and $v_{-1} = v_0$, so the inflow is $q_{-1} = \lambda_0 \rho_0 v_0$.
- The downstream density mirrors the last segment, $\rho_M = \rho_{M-1}$.

After each update the state is clipped to the physical box $\rho_i \in [0, \rho_{\max}]$ and $v_i \in [0, v_{\text{free},i}]$.

EGO evaluation. The coupling that makes the model controllable is that the EGO acts as a *moving bottleneck*. On the segment $i^\star$ it currently occupies, the equilibrium speed is capped by the EGO speed:

$$V_e^{\text{EGO}}(\rho_{i^\star}, r) = \min\big(V_e(\rho_{i^\star}),\, v(r)\big). \tag{6.21}$$

So a slower EGO drags the local stream down.

The per-segment occupancy is

$$x_i(k) = \rho_i(k)\, L_i\, \lambda_i, \tag{6.22}$$

which plays here the role that the link vehicle count x_i^h plays in the E-COSMOS traffic model, although there it never appears in the cost. The simplest way to bring the congestion term into the objective would be to penalise the absolute occupancy of the path links, but, as said before, for a single EGO car this drives the speed to zero (verified offline: with the absolute term the optimal mean speed collapsed from 48 km/h to 0 km/h). The meaningful "mitigate congestion" signal for one car will be developed considering the single entity itself (do not drive "fast" into a congested link), so the implemented term acts as a penalty but with a normalisation dependent on the velocity of the EGO car, guaranteeing the logic expected:

$$c_{\text{traffic}}(r) = w_2 \left(\frac{v(r)}{\bar{v}_i}\right)^2 \Big(\frac{1}{|\mathcal{W}|} \sum_{i \in \mathcal{W}} \frac{\rho_i(r)}{\rho_{\max}}\Big)^2, \quad \mathcal{W} = \{\text{EGO segment} + \text{next two}\}. \tag{6.23}$$

Being gated by the normalised density $\rho_i/\rho_{\max}$ results in *no effect on free roads* (where $\rho \to 0$) and acts only when congestion appears. Setting $w_2 = 0$ covers

the *not-traffic-aware* case; in Section 6.6 the two cases ($w_2 = 0$ and $w_2 = 40$ nominal case) are compared.

All the calibration constants (ρ_{crit}, $\rho_{\max}$, τ, ν, κ, a) are those of Table 2.2 in Section 2.3, evaluated at the MPC and METANET sampling time $M_T = 5\,\text{s}$.

The cost function. Summed over the horizon, the cost is

$$J = \sum_{r=0}^{N-1} \Big[\underbrace{w_1\,(1 - p(r))^2}_{\text{progress}} + \underbrace{w_a\,a(r)^2 + w_j\,j(r)^2}_{\text{comfort}} + \underbrace{c_{\text{traffic}}(r)}_{\text{congestion}} + \underbrace{c_{\text{lim}}(r)}_{\text{speed limit}} \Big]. \quad (6.24)$$

Each term plays a distinct role:

- the *progress* term rewards arrival, pushing the EGO as fast as the limits and the traffic allow;
- the two *comfort* terms penalise acceleration and jerk, and are the channel through which the clinical state acts on the ride;
- the *speed-limit* penalty keeps the profile under the patient-scaled per-segment road limit;
- the *congestion* term slows the reference where a jam is predicted (developed just below).

The speed-limit penalty is the cap which keeps the profile under the patient-scaled per-segment road limit $\bar{v}_i$ (with ϕ the clinical speed factor), being evaluated at the position consistent with V in the same forward pass so that the reference follows the per-segment limits smoothly.

$$c_{\text{lim}}(r) = w_{\text{lim}}\,\max\big(0,\; v(r) - \bar{v}_i\big)^2, \qquad \bar{v}_i = v_{\text{free},i} \cdot \phi, \quad (6.25)$$

Constraints. The optimiser enforces, over the whole horizon,

$$0 \;\le v(r) \;\le\; v_{\text{global max}}, \quad (6.26)$$

$$a_{\min} \;\le a(r) \;\le\; a_{\max}, \quad (6.27)$$

$$-j_{\max} \;\le j(r) \;\le\; j_{\max}. \quad (6.28)$$

Since $a(r)$ and $j(r)$ are linear combinations of the speed (6.16), then the two comfort constraints are *linear* in V. Exact Jacobians are supplied to SLSQP, so

the bounds are enforced exactly over the whole horizon (Section 6.6 shows the planned a and j never leaving their bands). The per-segment limit is enforced by c_{lim} because the applied command is clamped to the current edge limit in the simulation loop, so the reference never exceeds the road limit on the segment it occupies, even if the patient is in hurry (rush case analyzed below) because as a reference we can't encourage the person driving (here simulated by SUMO) to exceed the road limits.

Two timescales and the SUMO inner loop. SUMO advances at STEP_LENGTH $= 0.5\,\text{s}$, whereas the MPC and the METANET model re-solve at the larger cadence $M_T = 5\,\text{s}$ over $N = 16$ steps (about 80 s of look-ahead).

The first speed $v(0)$ of the optimal profile is *held* and pushed to SUMO every 0.5 s step until the next solve. SUMO's default car-following (`speedMode 31`) keeps the commanded speed bounded by the collision-safe speed and the vehicle's accel/decel limits, which are different and independent from the one defined for the MPC. Since SUMO is dislocated from it and sees only the velocity reference, the EGO car cannot reach an over-optimistic reference, which is exactly the effect that we will appreciate later.

Figure 6.3 collects the whole Formulation in a single picture.

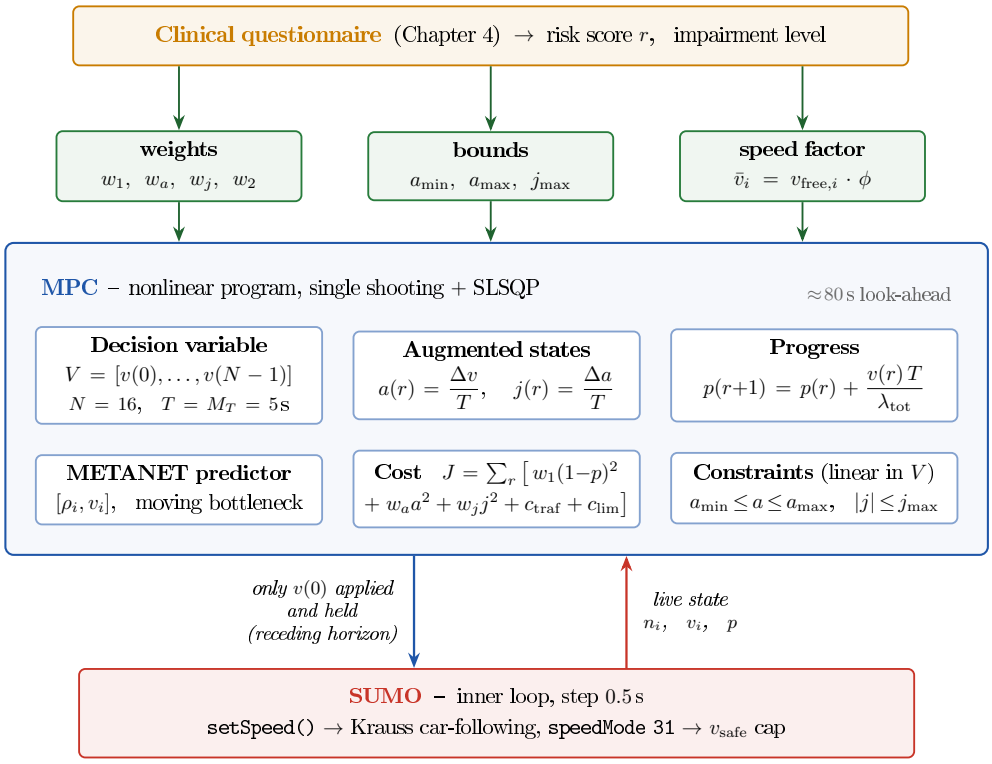


Figure 6.3: Composition of the traffic-aware patient MPC.

6.4.1 The final problem

Collecting all the aforementioned parts with the cost on top, dynamics and bounds as constraints, the problem reads

$$V^\star = \arg\min_{V} \quad \sum_{r=0}^{N-1} \Big[w_1 \,(1 - p(r))^2 + w_a \, a(r)^2 + w_j \, j(r)^2 + c_{\text{traffic}}(r) + c_{\text{lim}}(r) \Big] \tag{6.29a}$$

$$\text{s.t.} \quad p(r+1) = p(r) + \frac{v(r)\,T}{\lambda_{\text{tot}}}, \tag{6.29b}$$

$$\rho_i(r+1) = \rho_i(r) + \frac{T_h}{L_i\,\lambda_i}\big(q_{i-1}(r) - q_i(r)\big), \quad q_i = \lambda_i\,\rho_i\,v_i, \tag{6.29c}$$

$$v_i(r+1) = v_i + \frac{T_h}{\tau}\big(V_e(\rho_i) - v_i\big) + \frac{T_h}{L_i}\,v_i\,(v_{i-1} - v_i) - \frac{\nu\,T_h}{\tau\,L_i}\,\frac{\rho_{i+1} - \rho_i}{\rho_i + \kappa}, \tag{6.29d}$$

$$V_e(\rho_i) = v_{\text{free},i}\,\exp\Big[-\tfrac{1}{a}\big(\rho_i/\rho_{\text{crit}}\big)^a\Big], \quad V_e^{\,i^\star} = \min\big(V_e(\rho_{i^\star}),\, v(r)\big), \tag{6.29e}$$

$$a(r) = \frac{v(r) - v(r-1)}{T}, \qquad j(r) = \frac{a(r) - a(r-1)}{T}, \tag{6.29f}$$

$$p(0) = \hat{p}(t_j), \quad \rho_i(0) = \hat{\rho}_i(t_j), \quad v_i(0) = \hat{v}_i(t_j), \tag{6.29g}$$

$$0 \le v(r) \le v_{\max}, \tag{6.29h}$$

$$a_{\min} \le a(r) \le a_{\max}, \tag{6.29i}$$

$$-j_{\max} \le j(r) \le j_{\max}. \tag{6.29j}$$

The objective (6.29a) mixes the four goals already discussed, the constraints carry the dynamics and the limits.

The control algorithm in pseudocode. The whole controller can be summarised by two procedures. The inner one (Algorithm 3) is a single receding-horizon solve:

1. scores a candidate speed profile V by forward-simulating the EGO kinematics and the METANET traffic to evaluate the cost J of (6.24),
2. lets SLSQP search over V subject to the linear comfort constraints (again, only the first speed $V_0^\star$ is actually sent to SUMO).

Algorithm 3 Traffic-aware MPC speed-profile solve (one receding-horizon step).

Require: horizon N, step T; clinical weights and bounds $(w_1, w_a, w_j, w_2, a_{\min}, a_{\max}, j_{\max}, \phi)$; measured EGO state $(d_0, v_0, v_{\text{prev}}, a_{\text{prev}})$; measured METANET state (ρ^0, v^0); flag *trafficAware*

Ensure: speed command v_{cmd} for SUMO and the planned profile $V^\star$

function COST(V) ▷ forward simulation of one candidate speed profile
  $d \leftarrow d_0$; $v_- \leftarrow v_{\text{prev}}$; $a_- \leftarrow a_{\text{prev}}$; $(\rho, v) \leftarrow (\rho^0, v^0)$; $J \leftarrow 0$
  for $r = 0$ to $N - 1$ **do**
    $a \leftarrow (V_r - v_-)/T$; $j \leftarrow (a - a_-)/T$ ▷ accel, jerk as differences of V
    $d \leftarrow d + V_r\, T$; $p \leftarrow \min(d/\lambda_{\text{tot}},\, 1)$
    locate EGO segment $i^\star$; $\bar{v} \leftarrow v_{\text{free},i^\star}\, \phi$; $c_{\text{lim}} \leftarrow w_{\text{lim}} \max(0, V_r - \bar{v})^2$
    advance METANET (ρ, v) one step, EGO bottleneck $V_e^{i^\star} \leftarrow \min(V_e, V_r)$ ▷ Eqs. (6.19), (6.20)
    $c_{\text{traffic}} \leftarrow$ *trafficAware*? $w_2 (V_r/\bar{v})^2 \big(\overline{\rho/\rho_{\max}}\big)^2 : 0$
    $J \leftarrow J + w_1(1-p)^2 + w_a a^2 + w_j j^2 + c_{\text{traffic}} + c_{\text{lim}}$
    $v_- \leftarrow V_r$; $a_- \leftarrow a$
  end for
  return J
end function
assemble the comfort constraints $a \in [a_{\min}, a_{\max}]$, $|j| \le j_{\max}$ (linear in V)
$V^\star \leftarrow$ SLSQP(COST, V_0, bounds $[0, v_{\max}]^N$, constraints)
$v_{\text{cmd}} \leftarrow \text{clamp}\big(V_0^\star,\, 0,\, \text{current edge limit}\big)$
return $(v_{\text{cmd}},\, V^\star)$
▷ This Algorithm can be found inside of the folder with the name of `mpc_controller.py`.

The outer one (Algorithm 4) is the co-simulation loop that ties the two timescales together:

1. SUMO advances every 0.5 s, the MPC re-solves every $M_T = 5$ s,
2. the held command is pushed through `setSpeed`,
3. when re-routing is enabled, the upper-layer A* planner (Algorithm 1) supplies a new path whenever congestion is detected ahead.

Algorithm 4 Co-simulation receding-horizon control loop (SUMO $\leftrightarrow$ MPC).

Require: SUMO step $\Delta t = 0.5\,\mathrm{s}$, MPC period $M_T = 5\,\mathrm{s}$, planned path $\mathcal{P}$, clinical configuration

Ensure: the EGO reaches its destination with a patient-comfortable, traffic-aware speed

build the MPC problem from $\mathcal{P}$ (segments, clinical weights and bounds)
$v_{\mathrm{cmd}} \leftarrow$ undefined
while vehicles are still expected in the network **do**
    SumoStep(); $t \leftarrow$ simulation time
    if EGO is active **and** $t \bmod M_T = 0$ **then** ▷ re-solve every M_T
        measure v_0, v_{prev}, a_{prev} and the distance d_0 along $\mathcal{P}$
        measure per-edge densities and speeds $\rightarrow (\rho^0, v^0)$ ▷ re-init METANET
        $(v_{\mathrm{cmd}}, V^\star) \leftarrow$ SolveMpc(...) ▷ Algorithm 3
    end if
    if EGO is active **and** v_{cmd} is defined **then**
        `setSpeed`(EGO, v_{cmd}) ▷ held for the next $M_T/\Delta t$ SUMO steps
    end if
    if re-routing enabled **and** congestion detected on the remaining path **then**
        $\mathcal{P} \leftarrow$ AstarRoute(current edge, goal, live speeds) ▷ Algorithm 1
        rebuild the MPC segments from the new $\mathcal{P}$
    end if
end while

6.4.2 Personalising the optimisation: clinical parameters

The mapping of the clinical situation of the patient uses three quantities derived from the answers:

- $\mathrm{im}_{Qk} \in [0, 1]$, the per-question impairment of Equation (5.2) (e.g. im_{Q3} for motion-induced nausea, im_{Q8} for pain, im_{Q1} for general discomfort);
- $r \in [0, 1]$, the aggregate risk score of Equation (5.1);
- $u_n = (a_{Q7} - 1)/4 \in [0, 1]$, the normalised urgency from the trip-urgency question Q7.

As before, the weights and bounds of the optimal control problem are continuous functions of the nine-question clinical questionnaire of Chapter 5, so the

optimisation problem changes smoothly following the clinical state. The full mapping is collected in Table 6.1.

Table 6.1: Clinical parametrisation of the traffic-aware MPC. Each weight and bound is a continuous function of the questionnaire answers; the second column gives the healthiest (nominal) value.

Quantity	**Nominal**	**Dependence on clinical state**
w_1 (progress)	1.0	$1.0\,(1 + 1.5\,u_n)$, urgency pushes harder to the destination
w_a (acceleration)	0.6	$0.6\,\big(1 + 2.5\,\frac{\mathrm{im}_{Q3}+\mathrm{im}_{Q8}}{2}\big)$, nausea/pain penalise acceleration
w_j (jerk)	1.5	$1.5\,\big(1 + 4.0\,\frac{\mathrm{im}_{Q3}+\mathrm{im}_{Q1}}{2}\big)$, nausea/discomfort penalise jerk
w_2 (traffic)	40	$40\,\max(0.3,\, 1 - 0.4\,u_n)$, urgency slightly lowers caution
$a_{\max}$ [m/s^2]	2.0	$2.0\,\max(0.20,\, 1 - 0.55\,\mathrm{im}_{Q3} - 0.28\,\mathrm{im}_{Q8})$
$a_{\min}$ [m/s^2]	-3.0	$-3.0\,\max(0.30,\, 1 - 0.40\,\mathrm{im}_{Q3})$
$j_{\max}$ [m/s^3]	1.5	$1.5\,\max(0.12,\, 1 - 0.60\,\mathrm{im}_{Q3} - 0.22\,\mathrm{im}_{Q1})$
ϕ (speed factor)	1.0	$\max(0.55,\, 1 - 0.38\,r)$, higher risk caps the speed below the limit

One design property worth to be stressed out is the urgency factor because it acts only through the progress weight w_1 and the traffic weight w_2, so a healthy patient ($r \approx 0 \Rightarrow \phi = 1$) with maximum urgency (Q7 $= 5$) is driven up to the road limit and held there, never above it, recalling the safety idea explained before (Paragraph **Constraints.** in 6.4, Eq. (6.26)). The mildly-impaired patient behind the results of Section 6.6 (Level 2/5, aggregate risk $r = 0.25$, moderate motion sensitivity) maps to

- progress and traffic: $w_1 = 1.375$, $w_2 = 36$;
- comfort weights: $w_a = 0.975$, $w_j = 4.5$;
- comfort bounds: $a \in [-2.70,\, 1.585]\,\mathrm{m/s^2}$, $j_{\max} = 1.03\,\mathrm{m/s^3}$;
- speed factor: $\phi = 0.905$.

One observation from the reader could be: "How the parameters and the algebraic equations were chosen and how this can map correctly a real case scenario with real patients?"

Clearly, the parameters were chosen empirically and fine tuned by me to try to copy as much as possible a realistic trend. Again, these are all "variables" that can be modified by the final users for each type of usages, just to re-confirm

the flexibility of this work. But we will see, in the next subsection, that these numbers are not far from real cases studied before.

Justifying the mapping: a five-patient study

The formulas in Table 6.1 as said are design choices, and the controller's parametrisation was evaluated for five representative "type" patients under a fixed random seed (regarding the background traffic):

- **P1, healthy**: best answer to every question ($r = 0.00$, Level 1);
- **P2, healthy but urgent**: identical impairments to P1 but maximum trip urgency (Q7 = 5), included to isolate the effect of urgency;
- **P3, mild**: light discomfort and nausea ($r = 0.23$, Level 2);
- **P4, vestibular**: moderate motion sensitivity, nausea and pain (equilibrium problems such as labyrinthitis simulated here) ($r = 0.59$, Level 3);
- **P5, severe**: strong impairment across the board ($r = 1.00$, Level 5).

For each case we map the questionnaire parameters (Table 6.2) and then exercise the resulting controller, so we can appreciate the acceleration from rest to the patient's achievable cruising speed (Eq. (6.25)) on an empty road.

The density-gated congestion term is nearly 0 on a free network (Section 6.6), so this experiment isolates the *clinical* effect of the mapping from the stochastic traffic, and the resulting speed, acceleration and jerk traces are the exact behaviour of each clinical profile.

Table 6.2: Five-patient study: questionnaire risk score and the resulting MPC parametrisation.

Patient	r	**Lvl**	w_1	w_a	w_j	w_2	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$j_{\max}$ [m/s^3]	ϕ
P1 healthy	0.00	1	1.00	0.60	1.50	40	2.00	−3.00	1.50	1.00
P2 healthy/urgent	0.00	1	2.50	0.60	1.50	24	2.00	−3.00	1.50	1.00
P3 mild	0.23	2	1.38	0.79	3.00	36	1.73	−2.70	1.19	0.91
P4 vestibular	0.59	3	1.38	1.54	6.00	36	0.89	−2.10	0.58	0.78
P5 severe	1.00	5	1.00	2.10	7.50	40	0.40	−1.80	0.27	0.62

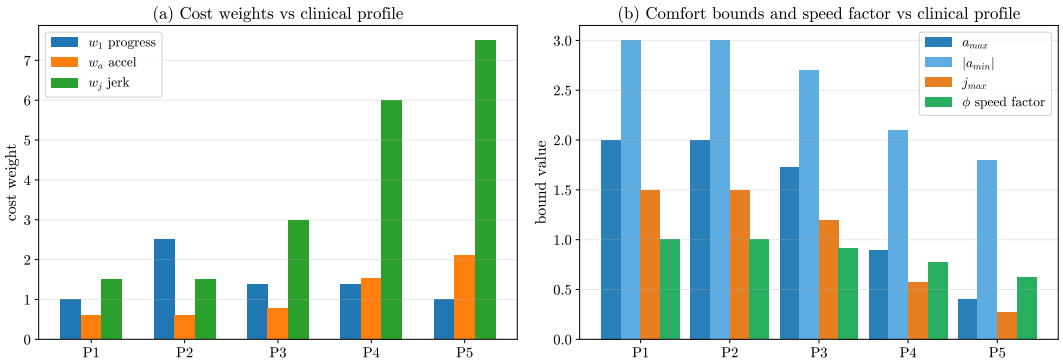


Figure 6.4: Clinical parametrisation across the five patients.

Figure 6.4 and Table 6.2 show that the mapping behaves as intended:

- the comfort weights w_a and w_j increase monotonically from the healthy P1 (0.60, 1.50) to the severe P5 (2.10, 7.50), because nausea (Q3), pain (Q8) and general discomfort (Q1) make the optimiser pay more for every unit of acceleration and jerk;
- the hard caps $a_{\max}$, $|a_{\min}|$ and $j_{\max}$ contract in the same direction (e.g. $a_{\max}$ from 2.0 to 0.40 $\mathrm{m/s^2}$), so even the most aggressive feasible manoeuvre stays gentle for a fragile patient;
- the speed factor ϕ falls from 1.0 to 0.62 with the aggregate risk, so higher-risk patients cruise further below the road limit;
- the progress weight w_1 is governed by urgency, not impairment: the urgent but healthy P2 has the largest $w_1 = 2.5$, while the most impaired P5 (not in a hurry) keeps $w_1 = 1.0$. This is the correct behaviour because urgency hastens the trip through w_1 and w_2, but never relaxes the comfort caps.

Table 6.3: Five-patient study: outcome of the traffic-free comfort manoeuvre.

Patient	v_{target} [km/h]	peak a [$\mathrm{m/s^2}$]	peak $\lvert j \rvert$ [$\mathrm{m/s^3}$]	t_{95} [s]
P1 healthy	50.0	2.00	1.50	7.5
P2 healthy/urgent	50.0	2.00	1.50	7.5
P3 mild	45.8	1.73	1.19	8.0
P4 vestibular	38.8	0.89	0.58	12.5
P5 severe	31.0	0.40	0.27	21.5

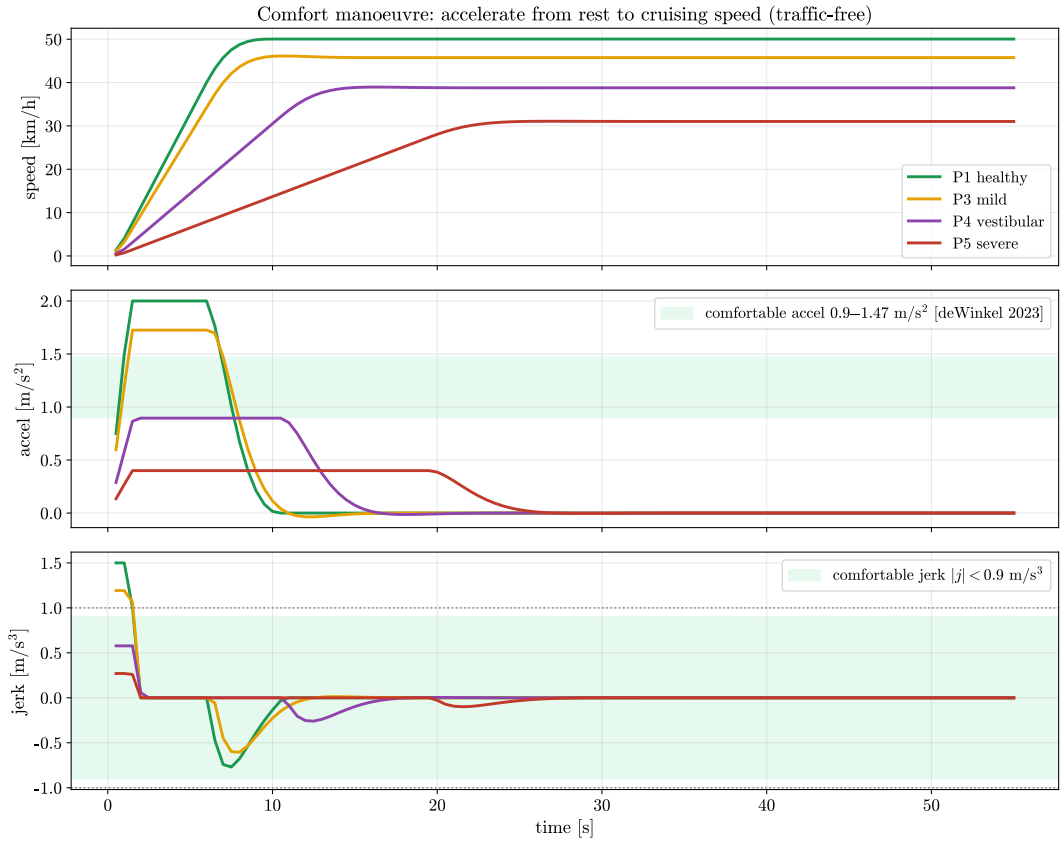


Figure 6.5: Comfort signature: the comfort manoeuvre accelerating each patient from rest to their cruising speed on a traffic-free road (P2 is omitted as it coincides with P1). The shaded bands are the design comfort envelope adopted in this work: an acceleration range of 0.9–1.47 $\mathrm{m/s^2}$, whose lower edge is the upper limit of the "Good" rating measured by de Winkel et al. [35] and whose upper edge is the acceptability limit for ground transportation reported by Hoberock [36], and a conservative jerk bound $|j| < 0.9\,\mathrm{m/s^3}$, well inside the $2.94\,\mathrm{m/s^3}$ acceptability threshold of [36].

The behavioural outcomes (Table 6.3, Figure 6.5) confirm the grading at the level of the ride itself. From P1 to P5 the peak acceleration falls from 2.0 to $0.40\,\mathrm{m/s^2}$, the peak jerk from 1.5 to $0.27\,\mathrm{m/s^3}$, and the 95 % settling time grows from 7.5 to 21.5 s. So, the greater the impairment, the gentler the acceleration and the longer the vehicle takes to settle at cruising speed: P1 and P2 coincide in this plot, which confirms the design property that urgency leaves the comfort weights and caps untouched (P2 differs from P1 only in how it weights progress and traffic).

6.4.3 Analysis from the literature

De Winkel et al. [35] placed 23 participants in a moving-base simulator and exposed them to motion pulses spanning peak accelerations of 0.4–2 $\mathrm{m/s^2}$ and peak jerks of 0.5–15 $\mathrm{m/s^3}$. Their central finding is that discomfort is governed by the acceleration amplitude: mapped onto verbal qualifiers, longitudinal accelerations up to 0.89 $\mathrm{m/s^2}$ are rated "Good" and the "So-so" point, which the authors read as the threshold of acceptability, falls at 1.23 $\mathrm{m/s^2}$. The effect of jerk, by contrast, was small and inconsistent in sign across participants, so that study provides no comfortable jerk band. The upper edge of the acceleration envelope adopted here, 1.47 $\mathrm{m/s^2}$, and the jerk figures come instead from the earlier survey of Hoberock [36], which places steady non-emergency longitudinal accelerations of 1.08–1.47 $\mathrm{m/s^2}$ in the acceptable range and considers jerks above 2.94 $\mathrm{m/s^3}$ unlikely to be accepted. The jerk bound $|j| < 0.9\,\mathrm{m/s^3}$ used is a conservative design choice, roughly three times stricter than the literature threshold. Mapping the five rides onto these envelopes:

- the healthy P1/P2 peak at 2.0 $\mathrm{m/s^2}$ and 1.5 $\mathrm{m/s^3}$, that is brisk and above the "comfortable" band but within everyday acceptable driving, which is the right trade for an unimpaired user who values progress.
- the mild P3 (1.73 $\mathrm{m/s^2}$, 1.19 $\mathrm{m/s^3}$) sits just above the comfortable bands of the jerk in the beginning, then stays always inside the two bands.
- the moderate P4 (0.90 $\mathrm{m/s^2}$, 0.58 $\mathrm{m/s^3}$) lands *inside* the comfortable band.
- the severe P5 (0.40 $\mathrm{m/s^2}$, 0.27 $\mathrm{m/s^3}$) is *below* even the comfortable lower bound for the acceleration case; the controller drives more gently than the nominal comfort standard.

The questionnaire-to-parameter maps follow a reasonable behaviour, not far from what we can consider real intervals. So, the most vulnerable patients are placed in very small ranges while letting healthier users trade a little comfort for progress towards destination.

6.5 Simulation Setup: Two Scenarios, Two Controllers

The experiment is a controlled comparison meant to isolate the effect of the METANET congestion term. It varies by levers, summarised in Table 6.4. The

first is the *traffic regime*:

- *congested* network (rush hour, dense injection),
- *free* network (night, almost empty).

The second is the *controller*:

- *not traffic aware* ($w_2 = 0$),
- *traffic aware* ($w_2 = 40$),

run in the *same* background traffic seed so that the only difference between them is the speed reference, on a *fixed* path.

The last one, used only in the congested scenario, is the *routing strategy*:

- fixed planned path,
- adaptive re-routing.

Table 6.4: The simulation setup: two traffic regimes, two controllers and, in congestion, two routing strategies. All runs use a Patient Level 2/5.

Lever	**Settings**
Traffic regime	congested (injection period 1 s, generation 420 s, EGO entry at 150 s, the loading phase) ; free (night, almost empty)
Controller	not traffic aware ($w_2 = 0$) ; traffic aware ($w_2 = 40$), same seed
Routing (congested only)	fixed planned path ; adaptive re-routing
MPC settings	$T = 5$ s, $N = 16$ (80 s horizon), 13 path segments

With re-routing active the EGO dodges every congested edge (it changed route about 8 times, different number compared to Section 5.3 due to different generation time and EGO injection; and had a leading vehicle only $\approx 5\%$ of the time), so it is never actually stuck and the traffic term has nothing to "work on" (only on a fixed path does, the speed reference reveal its effect).

Another important observation is that the congested network turns out to be essentially bimodal, either nearly free (injection period ≥ 2) or full gridlock (period = 1), with a very narrow "moderate" band. The congested scenario therefore uses the loading phase (EGO entry at 150 s) which produces both a strong reference over actual gap on the fixed path and a realistic escape route for the re-routing comparison.

6.6 Results

All results, as said before, are reported for a Patient Level 2/5 (risk $r = 0.25$); in every speed plot the solid line is the actual EGO speed, the dashed line is the MPC reference (its first value $v(0)$ applied each step), and the grey band is the road speed-limit envelope. The headline numbers are collected in Table 6.5.

Table 6.5: Final validated results (Patient Level 2/5). In congestion the traffic-aware term lowers the reference to the achievable speed and the gap is roughly half. On the contrary, in free traffic the two controllers nearly coincide. Please notice that the EGO is never encouraged to exceed road limits.

Scenario (routing)	**Controller**	$\overline{v}_{\mathbf{ref}}$ **[km/h]**	$\overline{v}$ **[km/h]**	**gap [km/h]**	**over limit**
Congested, fixed path	not aware	19.6	5.9	**13.7**	0 %
Congested, fixed path	aware	11.8	5.9	**5.9**	0 %
Free, fixed path	not aware	41.1	40.4	0.7	0 %
Free, fixed path	aware	40.7	40.4	0.3	0 %

6.6.1 Congestion case

Figure 6.6 shows the congested network on a fixed path. Here we appreciate on the left the *not-traffic-aware* controller produces a reference in the central part that oscillates around 14 km/h to 20 km/h, while the actual EGO is near the jam speed of ≈ 0 km/h (the reference is over-optimistic and physically unreachable). So the tracking gap explodes to 13.7 km/h. On the right, the *traffic-aware* controller lets the METANET congestion term (6.23) pull the reference down to the "more achievable" jam speed of about 12 km/h, which the car again cannot follow but the gap lowers to 5.9 km/h. In both panels the reference stays inside the road-limit envelope (the over-limit fraction is 0%). This is what we expect logically because the traffic term converts an unreachable speed reference into a more near realistic and trackable one.

We will nearly never obtain the same velocity profile for the *traffic-aware* and the EGO car, this is due to the fact that the MPC uses a different Macroscopic model (METANET) to simulate the traffic compared to the one used by SUMO (random traffic-Microscopic model).

The not-traffic-aware reference does not saturate the road limit because *only the first element of the profile is applied*: the optimal sequence is a ramp that

does reach the limit at the end of the horizon, but the EGO never leaves the queue, so every re-solve restarts from rest and re-applies the same first step.

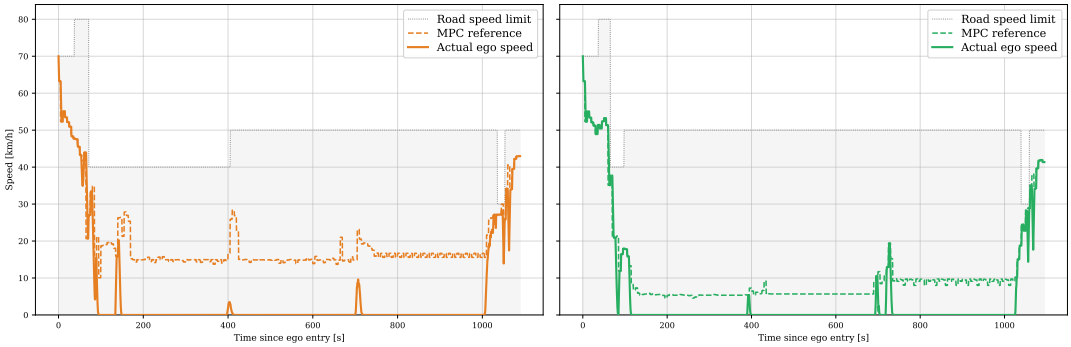


Figure 6.6: **Congested network** with a **fixed planned path**, Patient Level 2/5.

Does re-routing help the patient? Figure 6.7 fixes the traffic-aware controller and toggles only the routing (comparing not-aware against aware with re-routing would not be controlled, since the different reference would change the EGO's route and timing). On the fixed path the EGO moves in the jam for 1094 s. With adaptive re-routing it escapes onto the still-free side streets, cruising at 25 km/h to 45 km/h, and reaches the destination in 222 s, an important reduction in trip time. Also we appreciate the nearly same velocity profile between the reference and the EGO one.

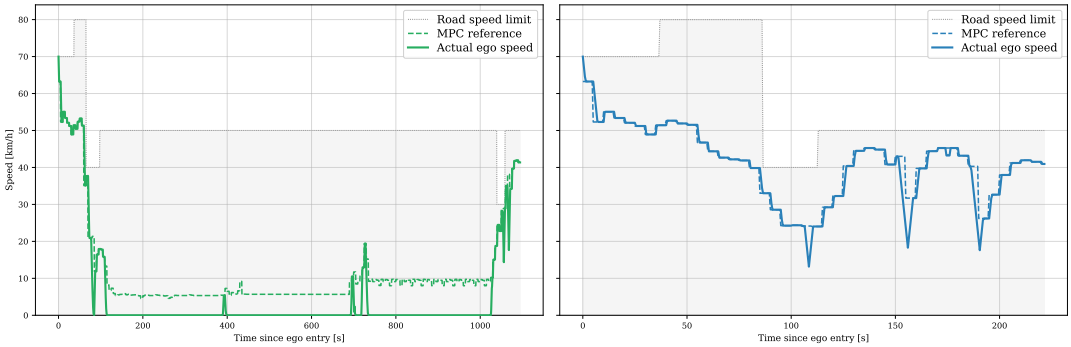


Figure 6.7: **Congested network**, **traffic-aware controller**: *fixed path* (left, 1094 s, the EGO is stuck in the queue) versus *adaptive re-routing* (right, 222 s, the EGO escapes onto free side streets).

6.6.2 Free traffic case

Figure 6.8 repeats the not-aware versus aware comparison on the free (night) network. The two panels nearly coincide because the congestion term (6.23) is

gated by the normalised density, it is essentially zero on an empty road, so the aware reference nearly equals the not-aware one.

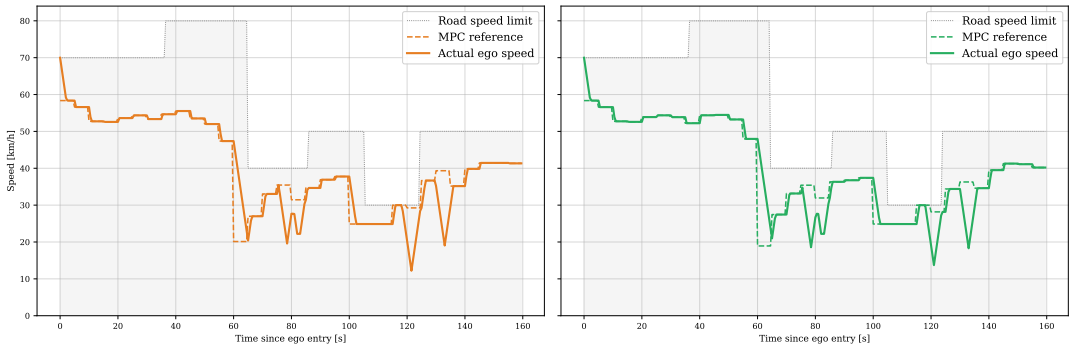


Figure 6.8: **Free network** (night), **fixed path**: *not traffic aware* (left) and *traffic aware* (right) are almost identical, because the density-gated congestion term is ≈ 0 on an empty network.

6.6.3 Constraint satisfaction: plan vs execution

As said at each control instant the MPC computes an optimal speed reference and sends only its first value $v(0)$ to SUMO through `setSpeed`. We hand over a target speed (e.g. like the copilot in the rally motorsport) and the driver chooses how to get there. Therefore: does the plan respect the patient's comfort constraints, and does the physical execution respect them too?

Figures 6.9 and 6.10 answer both, for the congested and the free run respectively. Each figure has two columns:

- **left, the MPC reference**: the commanded speed v and its own derivatives $a = \dot{v}$ and $j = \ddot{v}$, obtained as finite differences of the reference (6.16). The three are mutually consistent, since when v is constant $a = 0$ and $j = 0$;
- **right, the actual EGO** driven by SUMO while it tracks the reference: the realised speed, acceleration and jerk, drawn against the *same* comfort bands.

The *plan respects the constraints*, as expected; on the left column of Figures 6.9, 6.10 the commanded speed stays inside the road-limit envelope (acceleration stays inside $[a_{\min}, a_{\max}]$, and the reference jerk inside $\pm j_{\max}$) because the plan is comfortable by construction (constraints satisfaction, Section 6.4). Table 6.6 collects all the information for the Congested run, Table 6.7 for the Free (night) run.

In contrast, *the physical execution does not*: on the right column of Figures 6.9 and 6.10 the actual EGO speed follows the reference (closely on the free road; capped below it inside the jam, where the collision-safe speed dominates), and the actual acceleration still stays inside the band, because SUMO's own accel and decel limits ($\pm 1.5\,\mathrm{m/s^2}$) happen to be even tighter than the patient's. The actual jerk, however, leaves the $\pm j_{\max}$ band because SUMO's car-following is not jerk-limited, so it can switch acceleration abruptly from one step to the next.

Table 6.6: Congested run (traffic aware, Patient Level 2/5).

Quantity	**Comfort bound**	**MPC reference (plan)**	**Actual EGO (SUMO)**
speed v	$0 \le v \le \bar{v}_{\mathrm{road}}$	inside the envelope, 0% over-limit	tracks the reference, capped below it by the safe speed in the queue
accel. a	$[-2.70,\ 1.585]\,\mathrm{m/s^2}$	respected	respected (bounded by SUMO's $\pm 1.5\,\mathrm{m/s^2}$)
jerk j	$\lvert j\rvert \le 1.03\,\mathrm{m/s^3}$	respected	**exceeds** $j_{\max}$ (SUMO is not jerk-limited)

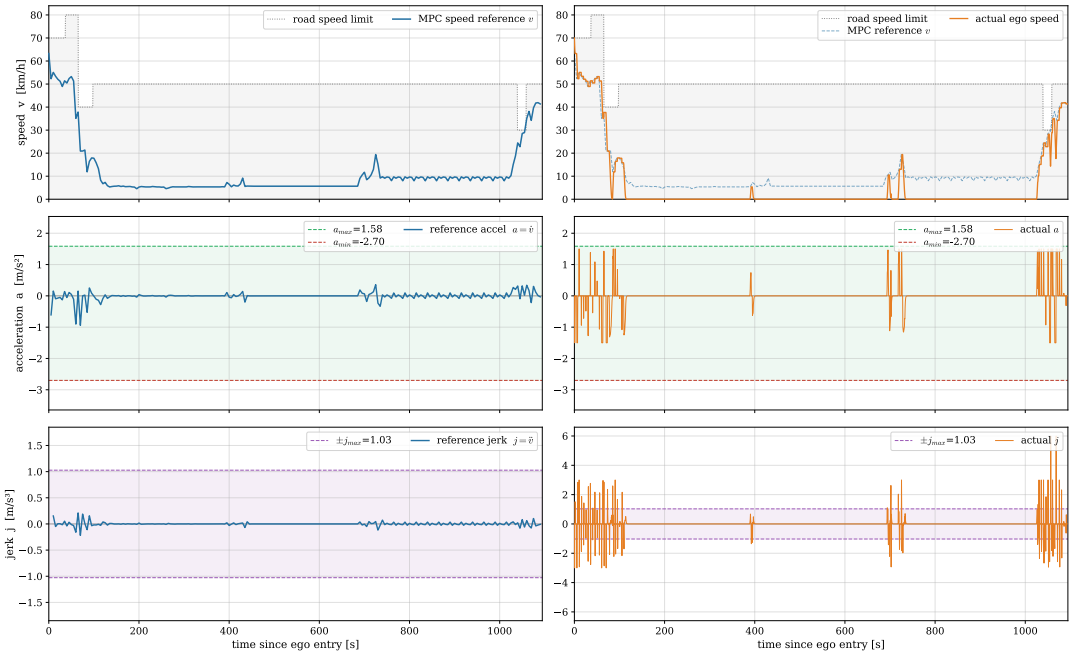


Figure 6.9: Constraint satisfaction, Congested run (Patient Level 2/5). Left: the MPC reference v and its derivatives a, j all stay inside the questionnaire bounds. Right: the actual v_{ego} driven by SUMO tracks the reference and keeps a inside the band, but its jerk leaves the $\pm j_{\max}$ band.

Table 6.7: Free (night) run (traffic aware, Patient Level 2/5).

Quantity	**Comfort bound**	**MPC reference (plan)**	**Actual EGO (SUMO)**
speed v	$0 \le v \le \bar{v}_{\text{road}}$	inside the envelope, 0% over-limit	tracks the reference closely (free road)
accel. a	$[-2.70,\ 1.585]\,\text{m/s}^2$	respected	respected (bounded by SUMO's $\pm 1.5\,\text{m/s}^2$)
jerk j	$\lvert j\rvert \le 1.03\,\text{m/s}^3$	respected	**exceeds** $j_{\max}$ (SUMO is not jerk-limited)

In other words, comfort is guaranteed on the reference we compute, not on the way SUMO decides to realise it. For example just as a driver may brake more sharply than a passenger would like when simply asked to slow down: we control *what* the car should aim for but the low-level execution remains in the hands of SUMO.

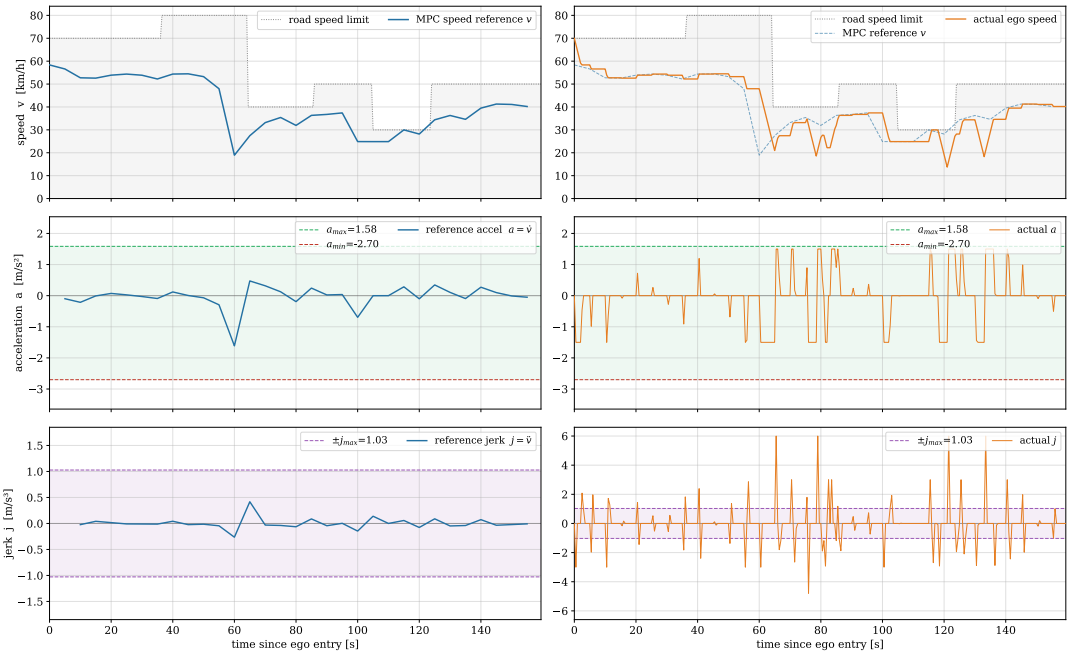


Figure 6.10: Constraint satisfaction, free (night) run (Patient Level 2/5). With almost no traffic the actual EGO speed tracks the reference closely; the plan (left) is comfortable on v, a and j. The same conclusions hold for the right part.

6.7 Application Outlook: analysis on solutions and existing hardware

The pipeline presented is meant to run in real time on the patient vehicle, so it is worth closing the chapter by asking where the computation should physically

take place. The recurring cost is:

- the routing operations of Chapter 5, measured in the sub-millisecond range (a single `findRoute()` at about 0.59 ms and a congestion check more than an order of magnitude cheaper, Table 5.7),
- while the traffic-aware MPC solves one small nonlinear program per control step of $T = 5\,\mathrm{s}$.

Denoting by t_{MPC} the solve time of a single receding-horizon step, the loop is real time as long as $t_{\mathrm{MPC}} \ll 5\,\mathrm{s}$, a condition met with a very wide margin on ordinary hardware. Two deployment strategies are possible, and they trade computational power against communication latency in opposite ways.

6.7.1 On-board computation

In the first strategy the whole pipeline lives inside the car. In the morning the patient boards, fills in the questionnaire once, and sets the destination. From that moment all the tasks are performed and computed on-board such as the route search, the periodic congestion checks, the per-step MPC and so on; no need for external connectivity. Two hardware option paths can be followed:

- **Dedicated ECU**

  Because the recurring load is a sub-millisecond route scan every 2.5 s and one small NLP every 5 s, a single automotive micro-controller of the class already used for vehicle control (e.g. the Infineon AURIX or NXP S32 families, certified up to ASIL-D and drawing only a few watts) is sufficient. The result is a small, deterministic and functionally-safe unit whose power draw is negligible, in an electric vehicle or a combustion engine car with the 12 V service battery.

- **Partition of an existing domain controller**

  Vehicles capable of Level 2 to Level 4 automation already carry a high-performance ADAS system-on-chip (e.g. NVIDIA Drive Orin which goes up to ∼254 TOPS at roughly 50 W to 75 W [39] or Qualcomm Snapdragon Ride) alongside a safety micro-controller. Our task is considered small in the field and can be hosted as a low-priority, time-sliced partition under a mixed-criticality hypervisor or AUTOSAR stack, adding essentially no marginal power and no extra hardware. The price to pay is *temporal*

isolation, due to the fact that the scheduler must guarantee that the optimisation never steals cycles from the safety-critical perception and control tasks.

This on-board route has the structural advantages that doesn't needs connectivity, so it's decoupled from cellular antennas, and it keeps the patient's clinical answers inside the vehicle, which is desirable for privacy. Figure 6.11 schematises the functioning logic.

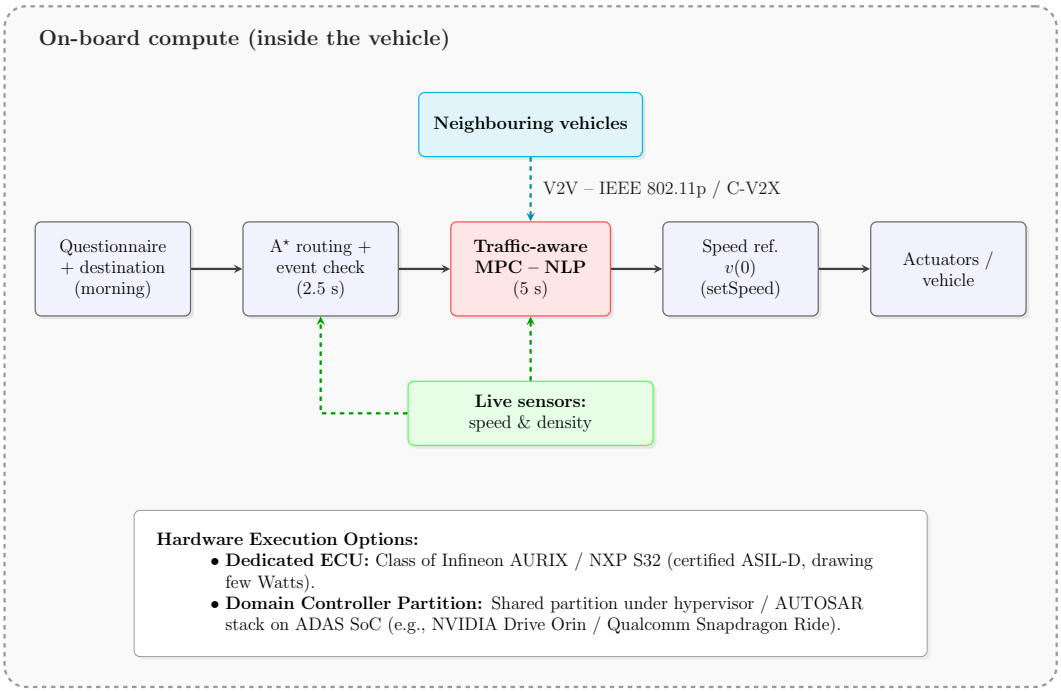


Figure 6.11: On-board deployment. The whole pipeline with morning questionnaire, patient-adaptive A* routing with event-driven checks, and the traffic-aware MPC running inside the vehicle, on a dedicated ECU or a time-sliced partition of the ADAS domain controller. Neighbouring vehicles may feed live speed and density into the METANET term over a V2V link (IEEE 802.11p / C-V2X).

Cooperative awareness over Wi-Fi. Even a fully on-board system can be improved by letting cars exchange information directly. The vehicular variant of Wi-Fi, IEEE 802.11p (DSRC, operating in the 5.9 GHz band with a one-hop transmission time of about 0.4 ms and an end-to-end safety budget near 100 ms), and the cellular alternative C-V2X in its direct mode (transmission around 1 ms and sub-20 ms latency) [40], let neighbouring vehicles broadcast their measured speed and density. Fed into the METANET term, these messages give the MPC a wider picture of the traffic ahead of the EGO car.

6.7.2 Edge and Cloud Server side computation

Now the heavy optimisation is offloaded to dedicated servers, so it's performed a partition of the map into regions, each served by an edge server (a roadside unit or a base-station-collocated MEC node) that handles the vehicles currently inside its service region. This strategy has the big advantage that can handle multiple cars, so we can now suppose multiple patients in a rehabilitative plan. Following the rationale, a server has far more compute power than any embedded unit, so each solve is faster and it can maintain an updated traffic estimate for the whole region (even co-optimising several vehicles at once) instead of every car estimating its own METANET state in isolation [41].

Therefore the focus now is no longer the solve time alone but the *end-to-end* latency,

$$t_{\text{loop}} = t_{\text{Tx}} + t_{\text{queue}} + t_{\text{compute}} + t_{\text{Rx}}, \tag{6.30}$$

the sum of:

- the uplink transmission,
- the queueing and scheduling on a server shared by many cars,
- the computation, and the downlink.

A 5G Ultra-Reliable Low-Latency (URLLC) air interface targets about 1 ms, an edge/MEC server is typically reachable within one to low tens of milliseconds round-trip, whereas a distant cloud adds tens to over a hundred milliseconds.

Because our control step is 5 s, even cloud latency is tolerable for updating the reference. What must never leave the car is the safety informations such as the collision-safe speed cap, health and clinical situations described in Chapter 5.

In terms of *protocols*, lightweight publish/subscribe messaging such as MQTT, or a real-time middleware such as DDS (deterministic and already common in automotive and ROS 2 stacks), suits the vehicle-to-server exchange, while standardised V2X message sets (SAE J2735 Basic Safety Messages in the U.S., the ETSI ITS-G5 CAM/DENM in Europe) carry the cooperative data, transported over C-V2X (LTE-V2X or 5G NR-V2X) or 802.11p/bd [40]. As density rises the wireless channel needs its own congestion control, mirroring, at the communication level, the scalability argument already made before.

Figure 6.12, following the same logic, represent the functioning operations.

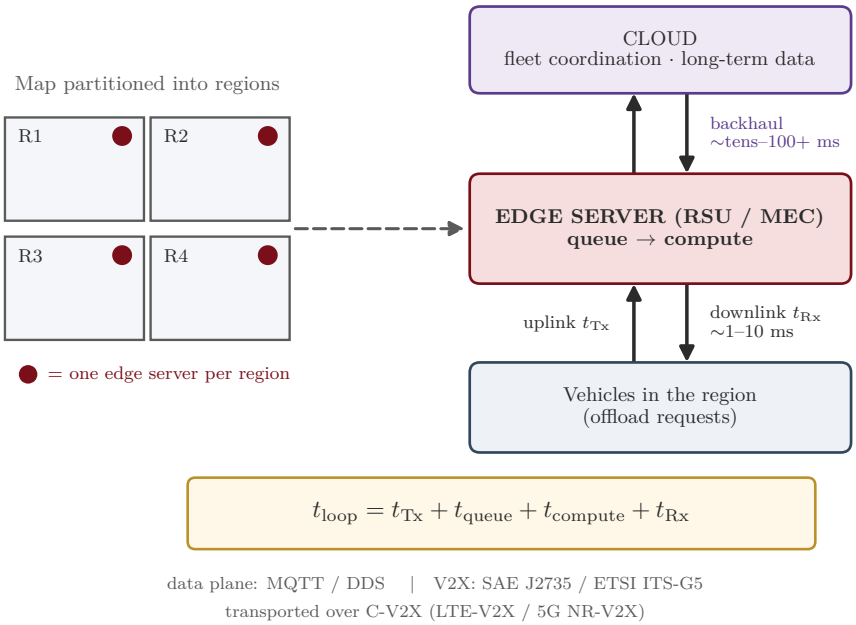


Figure 6.12: Server-side deployment. The map is partitioned into regions, each served by an edge (RSU / MEC) server that handles the vehicles currently inside it. Edge servers connect to a cloud backhaul and exchange data through MQTT/DDS and standardised V2X message sets over C-V2X.

6.7.3 Hybrid architecture

The best possible solution is a mix of the two, with a hybrid architecture where the safety, low-latency inner loop runs on-board, while the heavier, optimisation and the region-wide traffic estimate run on EDGE. Thus obtaining a safety fall-back to purely on-board operation whenever connectivity drops. Table 6.8 summarises the trade-off.

Table 6.8: Detailed comparison between the solutions discussed.

Features	**On-board dedicated ECU**	**On-board shared partition**	**Edge / cloud server**
Compute available	low, sufficient	high (shared)	very high
Added latency	none (local)	none (local)	∼1–100 ms (edge→cloud)
Connectivity needed	no	no	yes
Region-wide co-optimisation	no	no	yes
Data privacy	high	high	needs safeguards
Extra hardware cost	small unit	none	server infrastructure

With the measured on-board times of this thesis, is already shown that the local route is feasible on automotive hardware today; the server route becomes attractive when scaling to fleets and region wide co-optimisation, at the price of the communication and data management layer, whose latency and protocols are outlined. To conclude, Figure 6.13 describes the Hybrid solution.

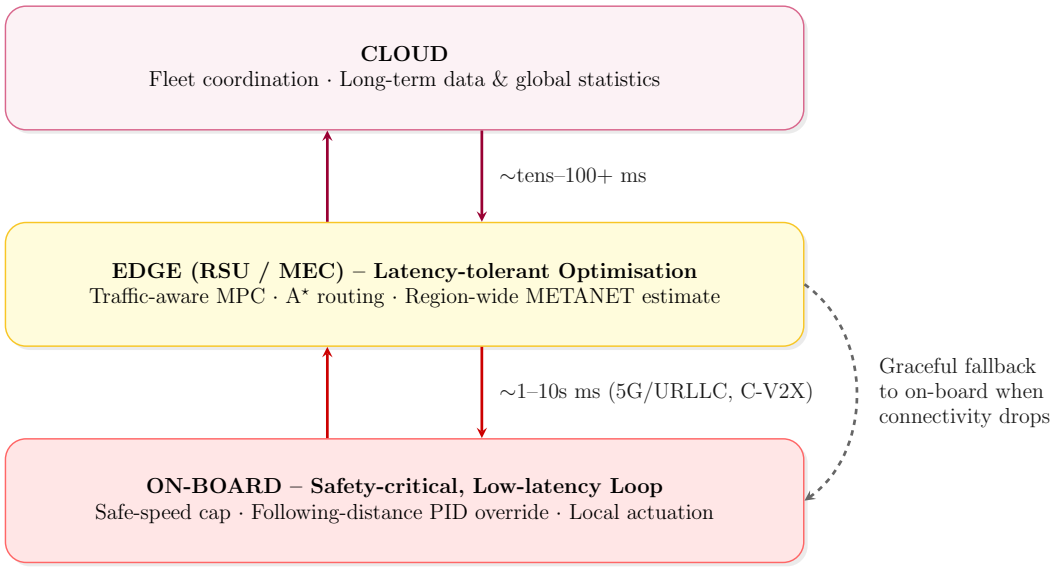


Figure 6.13: Hybrid, latency-aware architecture. The safety-critical low-latency loop stays on-board, while the latency-tolerant optimisation and the region-wide traffic estimate live on the edge (and coordination on the cloud), with fall-back option to purely on-board operation when connectivity drops.

Chapter 7

Final Conclusions

This closing chapter looks back at what the thesis set out to do, gathers the results obtained across its chapters, and examines them against the purpose that motivated the work.

7.1 Synthesis of the results

The health-adaptive routing of Chapter 5 captures the situation of the patient through the clinical answers and creates an influence matrix which translates into concrete guidance parameters. The patient-adaptive A^* planner then penalises the edges that a given clinical profile should avoid. The replanning snapshots of Figure 5.2 show the planner behaving exactly as intended.

Replacing the unconditional periodic reroute with a cheap congestion check that triggers the planner only when needed cut the number of routing queries, thus a saving on the CPU in terms of computational time, preserving route quality (Table 5.7, Figure 5.3). The complexity analysis further showed that this advantage grows with the size of the network (Figure 5.4), a useful property for the resource-constrained on-board computers discussed in Chapter 6.

The final controller casts the guidance as a traffic-aware speed MPC, with the EGO speed as the control, and when re-routing is enabled the patient escapes the jam and reaches its destination in 222 s against 1094 s on the fixed path (Figure 6.7); on a free road the density-gated term correctly vanishes and leaves the reference untouched (Figure 6.8). Thus the plan is comfortable and optimal by construction; the commanded speed, acceleration and jerk all stay inside the questionnaire-derived bounds.

The limit is that comfort is guaranteed on the reference we compute, not on

the way SUMO's generic car-following realises it, whose jerk can leave the band. This gap between plan and execution is precisely what a microscopic patient model, below the reference layer, is meant to close [42].

Taken together, the system never drives for the patient, it shapes the driving task so that it stays within the patient's current capacity. The guidance is measured and re-assessed at every trip, matching the graduated notion of fitness to drive of Annex III discussed in Chapter 1, and it is designed to be dialled down as the patient recovers.

In rehabilitative terms, the proposed work serves as substantial help for the patient in the pathway described in Section 1.3, which is strictly connected to the microscopic work mentioned in the following.

7.2 Towards a two-level, patient-in-the-loop system

This thesis is one half of a larger idea that Davide Faiola and I began together and then divided along the two natural scales of the traffic problem, shared at the upper level where the SUMO and EGO framing characterisation of the patient is described, then from there divided into a macroscopic and microscopic layer.

The companion Thesis [42] addressed how a specific patient is modelled as an imperfect longitudinal controller, characterised by reaction delay, reduced promptness and responsiveness, with acceleration and comfort limits. A threshold-based PID assist controller intervenes only when the speed-tracking error becomes significant, supporting the driver during the trip.

In that framework the EGO tracks a speed trajectory generated by a higher-level module. Conversely, his patient profiles and assist controller are the natural replacement for SUMO's generic car-following, which we already identified above as the link between plan and execution. Coupling the two would let the macroscopic reference be shaped not for an abstract driver but for the measured tracking behaviour of the specific patient. The result would be a two-level, patient-in-the-loop rehabilitative system: the merging of the two works combines into a classic cascade control scheme, sketched in Figure 7.1. The macroscopic layer of this work is the slow *outer loop* that generates the optimal speed reference $v^\star$, while the microscopic controller of Faiola [42] is the fast *inner loop* that tracks it on the EGO vehicle.

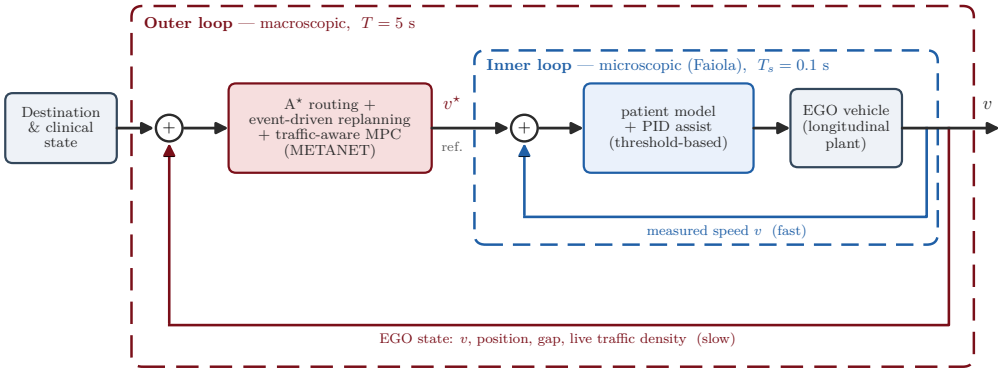


Figure 7.1: Two-level cascade architecture: the macroscopic outer loop of this thesis (routing and traffic-aware MPC) provides the speed reference $v^\star$ to the microscopic inner loop [42] (patient model and threshold-based PID assist) acting on the EGO vehicle.

7.3 Further developments

Beyond the nested loops, possible improvement could be related to the real-time deployment sketched in Chapter 6, on a dedicated on-board unit or on region-partitioned edge servers. May be prototyped and measured a vehicle-to-vehicle messages feeding the congestion term with live data, with the framework that should move from simulation toward an on-road study, validated together with the driving-rehabilitation specialists and clinicians described in Chapter 1, so that the questionnaire and the patient model can be calibrated on real scenarios. Also, the patient model itself can be enriched, both in the number of clinical dimensions it captures and in the fidelity of the microscopic controller that reproduces the patient's behaviour.

More than a century ago, in *The Machine Stops*, E. M. Forster imagined a humanity so wholly surrendered to an all-providing Machine that people forgot their importance and the reach of their own will, until, in Kuno's warning, "We created the Machine, to do our will, but we cannot make it do our will now" [43].

Here every kilometre covered under the guidance is meant to hand a fragment of control back to the person behind the wheel, so that the highest achievement of the machine becomes its own gradual withdrawal. Technology, in this vision, never sits at the centre, it stands in the service of the human being helping him to heal; its truest success is the day it's no longer needed.

Acknowledgements

I would like to thank my Advisor, Professor Antonella Ferrara, for the opportunity to work on this innovative project, giving us the full freedom to express and apply our studies.

I thank my Supervisor, Nikolas Sacchi, who mentored me throughout this journey with passion and dedication. He always left the creative initiative in my hands while guiding me to achieve the most complete and cohesive work possible.

I would also like to thank my colleagues, Davide Faiola and Gianluca Broglia, who accompanied me during the early collaborative phases of this project. They consistently provided opportunities for shared reflection along with moments of lightheartedness that will leave a wonderful memory in my mind, marking the conclusion of this academic journey at my University, to which I will always be grateful.

* * *

Ringrazio la mia Relatrice e Professoressa Antonella Ferrara per l'opportunità di lavorare a questo progetto innovativo, dandoci la piena possibilità di espressione dei nostri studi.

Ringrazio il mio Supervisore Nikolas Sacchi che mi ha seguito durante questo percorso con passione e cura, lasciando l'inventiva sempre nella mia mano e guidandomi per la realizzazione di un elaborato più completo e coeso possibile.

Ringrazio i miei compagni Davide Faiola e Gianluca Broglia che mi hanno accompagnato per le prime fasi collaborative del lavoro, fornendo sempre situazioni riflessive comuni con momenti di leggerezza che mi lasceranno un ricordo stupendo nella mia mente; alla chiusura di questo percorso di Studi nella mia Università, a cui sarò sempre riconoscente.

Bibliography

[1] European Parliament and Council of the European Union, *Directive 2006/126/EC on driving licences (recast)*, Official Journal of the European Union, L 403, pp. 18–60, Annex III: Minimum standards of physical and mental fitness for driving a power-driven vehicle. Amended by Commission Directives 2009/113/EC and 2014/85/EU, 2006. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32006L0126.

[2] Repubblica Italiana, *Decreto legislativo 18 aprile 2011, n. 59 — attuazione delle direttive 2006/126/CE e 2009/113/CE concernenti la patente di guida*, Gazzetta Ufficiale n. 99 del 30 aprile 2011, Supplemento Ordinario n. 104, 2011. [Online]. Available: https://www.gazzettaufficiale.it/eli/id/2011/04/30/011G0104/sg.

[3] H. Devos, C. A. Hawley, A. M. Conn, S. C. Marshall, and A. E. Akinwuntan, "Driving after stroke," in *Clinical Pathways in Stroke Rehabilitation: Evidence-based Clinical Practice Recommendations*, T. Platz, Ed., PMID: 36315689, Springer, 2021, pp. 243–260. DOI: 10.1007/978-3-030-58505-1_13.

[4] Association for Driver Rehabilitation Specialists (ADED), *Certified driver rehabilitation specialist (CDRS)*, https://www.aded.net, Accessed July 2026, 2026.

[5] C. A. Unsworth, A. Baker, N. A. Lannin, P. Harries, J. Strahan, and M. Browne, "Predicting fitness-to-drive following stroke using the Occupational Therapy – Driver Off Road Assessment Battery," *Disability and Rehabilitation*, vol. 41, no. 15, pp. 1797–1802, 2019, Published online 28 February 2018. PMID: 29488407. DOI: 10.1080/09638288.2018.1445784.

[6] A. Ferrara, S. Sacone, and S. Siri, *Freeway Traffic Modelling and Control* (Advances in Industrial Control). Cham, Switzerland: Springer Interna-

tional Publishing, 2018, ISBN: 978-3-319-75959-3. DOI: 10.1007/978-3-319-75961-6.

[7] M. J. Lighthill and G. B. Whitham, "On kinematic waves. II. A theory of traffic flow on long crowded roads," *Proceedings of the Royal Society of London A*, vol. 229, no. 1178, pp. 317–345, 1955.

[8] F. van Wageningen-Kessels, H. van Lint, K. Vuik, and S. P. Hoogendoorn, "Genealogy of traffic flow models," *EURO Journal on Transportation and Logistics*, vol. 4, no. 4, pp. 445–473, 2015. DOI: 10.1007/s13676-014-0045-5.

[9] S. P. Hoogendoorn and P. H. L. Bovy, "State-of-the-art of vehicular traffic flow modelling," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 215, no. 4, pp. 283–303, 2001. DOI: 10.1177/095965180121500402.

[10] B. D. Greenshields, "A study of traffic capacity," *Highway Research Board Proceedings*, vol. 14, pp. 448–477, 1935.

[11] P. I. Richards, "Shock waves on the highway," *Operations Research*, vol. 4, no. 1, pp. 42–51, 1956.

[12] C. F. Daganzo, "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory," *Transportation Research Part B: Methodological*, vol. 28, no. 4, pp. 269–287, 1994.

[13] H. J. Payne, "Models of freeway traffic and control," in *Mathematical Models of Public Systems*, ser. Simulation Council Proceedings Series 1, G. A. Bekey, Ed., vol. 1, La Jolla, CA: Simulation Councils Inc., 1971, pp. 51–61.

[14] G. B. Whitham, *Linear and Nonlinear Waves*. New York, NY, USA: John Wiley & Sons, 1974, ISBN: 978-0-471-94090-6.

[15] C. F. Daganzo, "Requiem for second-order fluid approximations of traffic flow," *Transportation Research Part B: Methodological*, vol. 29, no. 4, pp. 277–286, 1995.

[16] A. Aw and M. Rascle, "Resurrection of "second order" models of traffic flow," *SIAM Journal on Applied Mathematics*, vol. 60, no. 3, pp. 916–938, 2000.

[17] H. M. Zhang, "A non-equilibrium traffic model devoid of gas-like behavior," *Transportation Research Part B: Methodological*, vol. 36, no. 3, pp. 275–290, 2002.

[18] A. Messmer and M. Papageorgiou, "METANET: A macroscopic simulation program for motorway networks," *Traffic Engineering and Control*, vol. 31, no. 8–9, pp. 466–470, 1990.

[19] M. Papageorgiou, J.-M. Blosseville, and H. Hadj-Salem, "Modelling and real-time control of traffic flow on the southern part of Boulevard Périphérique in Paris: Part I: Modelling," *Transportation Research Part A: General*, vol. 24, no. 5, pp. 345–359, 1990. DOI: 10.1016/0191-2607(90)90047-A.

[20] A. Kotsialos, M. Papageorgiou, C. Diakaki, Y. Pavlis, and F. Middelham, "Traffic flow modeling of large-scale motorway networks using the macroscopic modeling tool METANET," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 4, pp. 282–292, 2002. DOI: 10.1109/TITS.2002.806804.

[21] N. Sacchi, G. Basile, S. Siri, M. Minetti, A. Bonfiglio, and A. Ferrara, *Optimal dispatch of connected and autonomous electric vehicles to enhance short-term grid flexibility in smart cities*, Work developed within the E-COSMOS project (Italian MUR, Italy–Serbia Joint Call 2024–2026), 2026. arXiv: 2605.25847 [eess.SY]. [Online]. Available: https://arxiv.org/abs/2605.25847.

[22] V. Dinopoulou, C. Diakaki, and M. Papageorgiou, "Applications of the urban traffic control strategy TUC," *European Journal of Operational Research*, vol. 175, no. 3, pp. 1652–1665, 2006. DOI: 10.1016/j.ejor.2005.02.032.

[23] Eclipse SUMO Development Team, *SUMO – Simulation of Urban MObility (version 1.26.0)*, https://sumo.dlr.de, Released 29 January 2026. Accessed: May 2026, 2026. DOI: 10.5281/zenodo.18406080.

[24] P. A. Lopez et al., "Microscopic traffic simulation using SUMO," in *Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 2575–2582. DOI: 10.1109/ITSC.2018.8569938.

[25] S. Krauß, "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics," Ph.D. dissertation, University of Cologne, German Aerospace Center (DLR), 1998.

[26] OpenStreetMap contributors, *OpenStreetMap*, `https://www.openstreetmap.org`, Accessed: March 2026, 2024.

[27] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "TraCI: An interface for coupling road traffic and network simulators," in *Proceedings of the 11th Communications and Networking Simulation Symposium (CNS '08)*, Ottawa, ON, Canada: ACM, 2008, pp. 155–163. DOI: 10.1145/1400713.1400740.

[28] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA, USA, 2008, pp. 11–15.

[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 4th. Cambridge, MA: MIT Press, 2022, ISBN: 978-0-262-04630-5.

[30] P. Bolzern, R. Scattolini, and N. Schiavoni, *Fondamenti di Controlli Automatici*, 4th ed. Milano, Italy: McGraw-Hill Education, 2015, ISBN: 978-88-386-6882-1.

[31] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Transactions of the ASME*, vol. 64, pp. 759–768, 1942.

[32] K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tuning*, 2nd ed. Research Triangle Park, NC, USA: Instrument Society of America (ISA), 1995, ISBN: 978-1-55617-516-9.

[33] L. Magni and R. Scattolini, *Advanced and Multivariable Control*. Bologna, Italy: Pitagora Editrice, 2014, ISBN: 978-88-371-1905-8.

[34] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[35] K. N. de Winkel, T. Irmak, R. Happee, and B. Shyrokau, "Standards for passenger comfort in automated vehicles: Acceleration and jerk," *Applied Ergonomics*, vol. 106, p. 103 881, 2023. DOI: 10.1016/j.apergo.2022.103881.

[36] L. L. Hoberock, "A survey of longitudinal acceleration comfort studies in ground transportation vehicles," *Journal of Dynamic Systems, Measurement, and Control*, vol. 99, no. 2, pp. 76–84, 1977, Originally issued as Research Report 40, Council for Advanced Transportation Studies, University of Texas at Austin, 1976. DOI: 10.1115/1.3427093.

[37] A. Bemporad and D. Barcelli, "Decentralized model predictive control," in *Networked Control Systems*, ser. Lecture Notes in Control and Information Sciences, A. Bemporad, M. Heemels, and M. Johansson, Eds., vol. 406, Berlin, Heidelberg: Springer-Verlag, 2010, pp. 149–178.

[38] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, et al., "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020, SciPy 1.0 Contributors. DOI: 10.1038/s41592-019-0686-2.

[39] NVIDIA Corporation, *NVIDIA DRIVE AGX Orin development platform*, https://developer.nvidia.com/downloads/drive/docs/nvidia-drive-agx-orin-platform-for-developers.pdf, Orin-X SoC: up to 254 INT8 TOPS; DevKit system power 200 W. Accessed July 2026, 2025.

[40] G. Naik, B. Choudhury, and J.-M. Park, "IEEE 802.11bd & 5G NR V2X: Evolution of radio access technologies for V2X communications," *IEEE Access*, vol. 7, pp. 70 169–70 184, 2019. DOI: 10.1109/ACCESS.2019.2919489.

[41] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Networks and Applications*, vol. 26, pp. 1145–1168, 2021. DOI: 10.1007/s11036-020-01624-1.

[42] D. Faiola, "Microscopic Traffic Simulation and Patient-In-The-Loop Longitudinal Control in a Driving Rehabilitation Context," Master's thesis, University of Pavia, 2026.

[43] E. M. Forster, "The machine stops," in *The Eternal Moment and Other Stories*, Originally published in *The Oxford and Cambridge Review*, November 1909, London, UK: Sidgwick & Jackson, 1928.