MŰEGYETEM 1782

Budapest University of Technology and Economics
Faculty of Transportation Engineering and Vehicle Engineering
Department of Control and Transport Automation

B.Sc. Thesis

Traffic Parameter Estimation in Urban Road Networks based on Radio Signaling Data

Ádám Ludvig

CJFAC3

Supervisor:	External Advisor:
Tamás Tettamanti	Hunor Demeter
assistant lecturer	senior research engineer
DCTA, BME	Nokia Siemens Networks

Budapest, June 5, 2012

Contents

Abstract

There are many existing radio based technologies providing data to identify a traveller at different places in an urban traffic network. One package of information contains the position and the time stamp of one point of the traveller's trajectory and some data which is unique to the traveller. This document shows a way to estimate real time traffic parameters, especially travel time, on large scale urban road network from combined radio signal data streams of various technologies. The estimation applies Kalman filter method. The applied software environment is Matlab and validation took place on simulated traffic data from VISSIM.

Keywords: Location Based Services, Kalman filter, Urban network, GSM, Travel time estimation, VISSIM

Acknowledgement

First and foremost I offer my sincerest gratitude to my supervisor, Tamás Tettamanti, who has supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way. I attribute the level of my Bachelors degree to his encouragement and effort. One simply could not wish for a better or friendlier supervisor.

Hunor Demeter and Norbert Vékony, my fellow colleagues at Nokia Siemens Networks, taught me several aspects of modern corporal research process as datamining techniques and agile software development. Hunor shared his great experience in telecommunication technologies and Norbert was always helpful when I needed someone. Without them this thesis, too, would not have been completed or written.

János Polgár, my fellow at the University, showed me lots of tricks in VISSIM and helped me when I stuck somewhere deep in the complex simulations. He gave objective critics and tips how my thesis could ameliorate by.

Chapter 1

Introduction

Nowadays intelligent transportation system (ITS) is a popular topic in traffic and especially urban traffic management and control. An ITS is a transportation system that makes use of information and communication technology to address and alleviate transportation and congestion problems. In general, an ITS relies on location-based information: It monitors and processes the location of a certain number of vehicles (used as probes) to obtain information on estimated travel time, driving conditions and traffic incidents. Using a relatively large amount of probes, the early stages of bottlenecks can be detected, and traffic can be directed to other routes to mitigate congestion and provide more expedient and efficient itineraries to travellers. A variety of sensors can be used to obtain traffic information [5].

Immediate intervention could prevent congestions hereby decrease negative externalities in urban traffic as fuel consumption, environmental pollution and increase economical productivity. Qualitative and quantitative analyses of the external costs which urban traffic congestion brings are the base of improving travel demand management and designing relative strategies, and even will help to maximize the effective usage of urban road resources [30].

These applications require reliable source of real time traffic data to measure or estimate various intrinsic traffic parameters for efficient road traffic management in this way reducing total cost of a town and enhance efficiency of economics.

I am a forgetful person, it is a serious problem for me to silence my cell phone during classes. Years ago I was looking for a reliable solution to this problem when I first met the possibility to gain location information from the cellular network. I created small scripts and applications for my smart phone that were usable in some university projects to simplify data collection.

Later I read the book Bursts [1] from Albert-László Barabási and I got very excited about using cellular network based location information.

I was working for a year for the Department of Control and Transport Automation on various interesting projects involving PLC automation and variable message signs when Tamás, my supervisor at the department, mentioned that he is working on a research project on cellular network based traffic measurement for Nokia Siemens Networks. At that time I was very impressed by the opportunities of the method and persuaded Tamás to let me join.

In 2011 September I joined the team at Nokia Siemens Networks as a collaborative student to work on the theory of estimation as well as on the actual code to develop the software. The goal of the team is to create a working commercial grade software which brought on many challenges to resolve. One of these was the task to aggregate the gathered individual information to a consistent, simple traffic variable.

Tamás' and my duty was to examine possible ways to aggregate these pieces of information. I suggested the following method as a possible way but the team decided not to utilize it because it needed more research and was not finished at the time. Instead a basic method was chosen and implemented.

I decided to examine and finish the research work on my own on this matter because it is superior in essence and it seemed an ideal subject for my thesis.

Chapter 2

Preliminaries

2.1 Background of Radio Signaling Technology

Many different technologies could be utilised to provide the required location information stream for traffic parameter estimation.

2.1.1 GPS

A satellite navigation or SAT NAV system is a system of satellites that provide autonomous geo-spatial positioning with global coverage. It allows small electronic receivers to determine their location (longitude, latitude, and altitude) to within a few metres using time signals transmitted along a line-of-sight by radio from satellites. Receivers calculate the precise time as well as position, which can be used as a reference for scientific experiments. A satellite navigation system with global coverage may be termed a global navigation satellite system or GNSS [29].

The Global Positioning System is a satellite based navigation system maintained by the government of United States that provides position and time information. It is freely available by anyone at any place where there is an unobstructed line of sight to at least four GPS satellites [25].

2.1.2 GSM

GSM (Global System for Mobile Communications, originally Groupe Spécial Mobile), is a standard set developed by the European Telecommunications Standards Institute to describe technologies for second generation digital cellular networks [26].

GSM is a cellular network, which means that cell phones connect to it by searching for cells in the immediate vicinity. There are five different cell sizes in a GSM network — macro, micro, pico, femto and umbrella cells. The coverage area of each cell varies according to the implementation environment. Macro cells can be regarded as cells where the base station antenna is installed on a mast or a building above average roof top level. Micro cells are cells whose antenna height is under average roof top level; they are typically used in urban areas. Picocells are small cells whose coverage diameter is a few dozen metres; they are mainly used indoors. Femtocells are cells designed for use in residential or small business environments and connect to the service provider's network via a broadband internet connection. Umbrella cells are used to cover shadowed regions of smaller cells and fill in gaps in coverage between those cells [26].

Cell horizontal radius varies depending on antenna height, antenna gain and propagation conditions from a couple of hundred metres to several tens of kilometres [26].

The base station subsystem (BSS) is the section of a traditional cellular telephone network which is responsible for handling traffic and signaling between a mobile phone and the network switching subsystem. The BSS carries out transcoding of speech channels, allocation of radio channels to mobile phones, paging, transmission and reception over the air interface and many other tasks related to the radio network [26, 12].

Telephone calls are made and received through a mobile terminal that, in official GSM terminology, is denoted as Mobile Station (MS). The terminal contains the technical equipment and a Subscriber Identity Module (SIM), which is a small chip card that stores subscriber-specific identifiers, addresses, as well as keys to authentication and encryption. It is composed by the network operator and delivered to each subscriber with a valid subscription [26, 12].

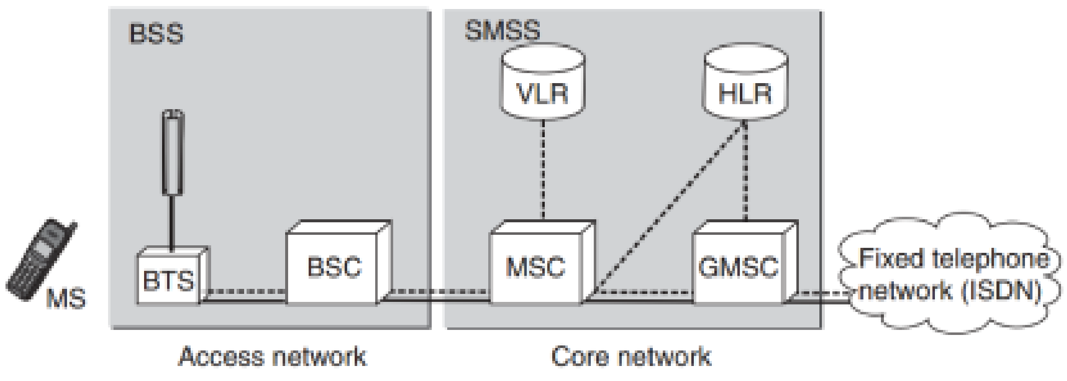

Figure 2.1: GSM architecture [12]

The BSS is responsible for monitoring and controlling the air interface. It consists of two different components, which are called Base Transceiver Station (BTS) and

Base Station Controller (BSC). BTS is the official GSM term for *base station* and thus contains transmitter and receiver equipment as well as an antenna. Figure 2.1 displays the basic architecture of GSM network. An important design goal was to keep the base stations as simple and as cheap as possible, and hence they are equipped with only very limited capabilities for signal and protocol processing. The bulk of the work, for example, allocation and release of channels at the air interface, is done by the BSC. Also, the BSC is responsible for control and execution of handover, a function which is needed to keep a circuit-switched connection if the subscriber moves between base stations. Each BSC controls several base stations, which are connected to the BSC via fixed lines or radio link systems [12].

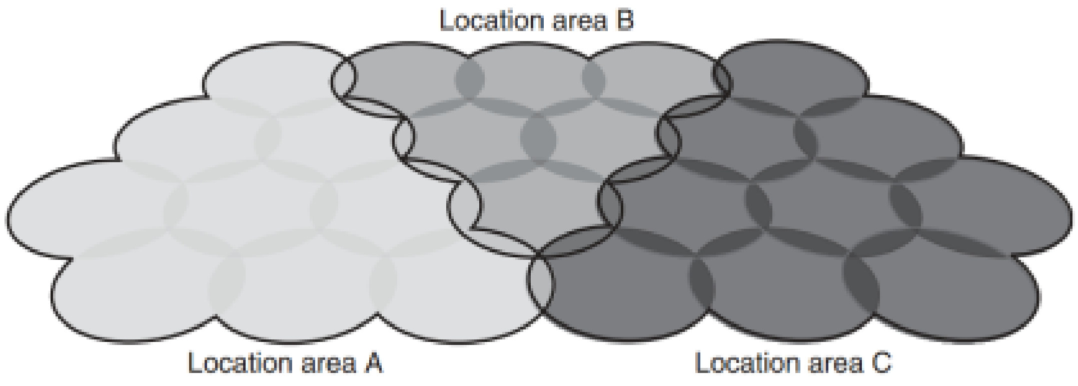

Figure 2.2: Location areas [12]

The cells of the network are grouped into location areas as seen in Fig. 2.2. This serves the purpose to roughly follow the position of each MS. If the MS crosses the boundary of the location area a location update event is emitted by the MS and it can be monitored by the operator.

Whilst the MS is in active mode, e.g. a call is in progress, each pass of a cell boundary emits a handover event as the two BTS passes the MS.

Among other possibilities these two kinds of events carry accurate enough location aware information to fund location based services [11].

2.1.3 Bluetooth

Bluetooth is a proprietary open wireless technology standard for exchanging data over short distances (using short-wavelength radio transmissions in the ISM band from 2400–2480 MHz) from fixed and mobile devices, creating personal area networks (PANs) with high levels of security. Created by telecommunication vendor Ericsson in 1994, it was originally conceived as a wireless alternative to RS-232 data cables. It can connect several devices, overcoming problems of synchronization [24].

Every device has a unique 48-bit address. However, these addresses are generally not shown in inquiries. Instead, friendly Bluetooth names are used, which can be set by the user. This name appears when another user scans for devices and in lists of paired devices [24].

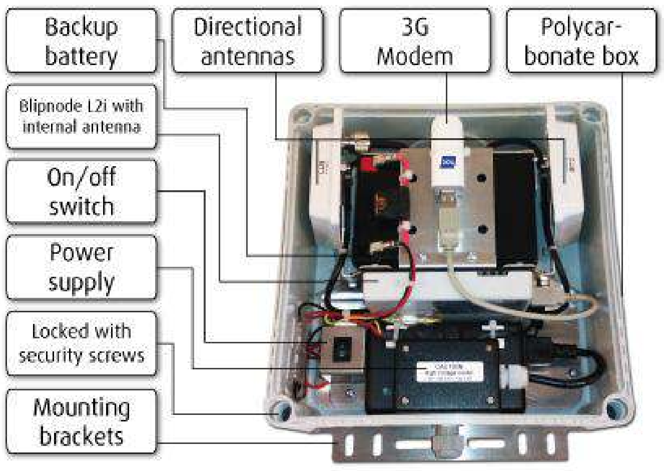

Figure 2.3: BlipTrack^{TM} Bluetooth Traffic sensor

Most phones have the Bluetooth name set to the manufacturer and model of the phone by default. Most phones and laptops show only the Bluetooth names and special programs are required to get additional information about remote devices.

A Media Access Control address (MAC address) is a unique identifier assigned to network interfaces for communications on the physical network segment. MAC addresses are used for numerous network technologies and most IEEE 802 network technologies, including Ethernet. Logically, MAC addresses are used in the Media Access Control protocol sub-layer of the OSI reference model.

The MAC address is publicly available if the device is turned on thus it is perfect to record and pair events of users passing a nearby traffic sensor like the one in Fig. 2.3 manufactured by the company BlipTrack.

2.1.4 RFID

Radio-frequency identification is the use of a wireless non-contact system that uses radio-frequency electromagnetic fields to transfer data from a tag attached to an object, for the purposes of automatic identification and tracking. The tag contains electronically stored information which can be read from up to several metres away. Unlike a bar code, the tag does not need to be within line of sight of the reader and may be embedded in the tracked object [28].

It is widely used as automatic vehicle identification method in many countries for electronic toll collection.

To track and locate vehicles along fixed routes, a technology called Signpost transmitters is employed. This is used on transit routes and rail lines where the vehicles to be tracked are operated mostly on the same linear route. A transponder or RFID chip along the vehicle route would be polled as the train or bus traverses its route. As each transponder was passed, the moving vehicle would query and receive an acknowledgement, or handshake, from the signpost transmitter. A transmitter on the mobile would report passing the signpost to a system controller. This allows supervision, a call center, or a dispatch center to monitor the progress of the vehicle and assess whether or not the vehicle was on schedule. These systems are alternative solutions inside tunnels or other conveyances where GPS signals are blocked by terrain.

2.2 Methods of data collection

There are fundamentally three different ways to collect required real time information stream about location of travellers.

2.2.1 Client side data collection

Client side data collection is based on the travellers. The traveller should have a device to obtain location data and another device to send his piece of information in real time to central traffic headquarters for processing.

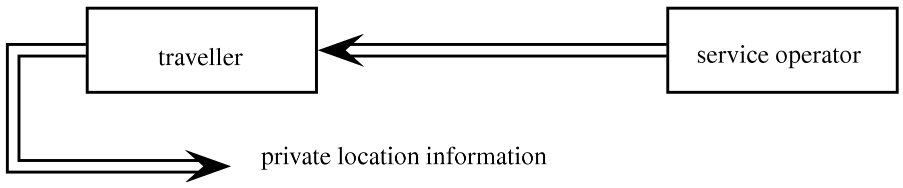

Figure 2.4: Client side data collection

Location information could be gathered directly via a GPS receiver device or indirectly from location relevant information and an appropriate database to resolve the bit of information. For example the latter could be the Cell Id identification code of local base station of a cellular network, an RFID based signpost system along the road network or simply wireless local area network identifiers through wardriving.

Nowadays mobile internet access via cellular networks is the dominant way to send gathered information to a central institute. This method charges the traveller but based on the amount of information and present-day internet usage customs these charges are negligible.

This method depends heavily on the number of travellers who participate. A small portion of travellers provide enough information to estimate traffic within customary tolerance level [20]. Mobile communication and cellular network is another key to produce real time information but urban areas are usually well covered. The GPS signals can be used well in rural and suburban territories and the inner cities, where GPS signal could be lost, are densely covered by cellular networks and wireless local networks.

The method provides accurate and quick information but needs to build an excessive user base which can be a bottleneck for a startup project.

There are many traffic information providers collecting data this way for example Google's Maps service which gets information from navigating Android devices at the time[3]. Another interesting and fast developing application is AntaresNav's EgérÚt application which provides dynamic navigation for all major smart-phone platforms restricted to Hungary for now[10].

2.2.2 Server side data collection

Server side data collection gains information from functioning, already built services by persuasion of the service provider to supply practical data about its customers.

Mainly, mobile network operators could provide usable information for traffic applications. The are location representative events happening in cellular networks like Handover and Location Update. One more required piece of information is a unique identifier of the user that can be used to group the mobility events by user. The unique identifier reveals privacy issues causing one of the key difficulties in the operator's persuasion. This identifier can be hashed and encoded but should be unique for a duration. Duration depends on the application, the technology and the dominant length of travels, for urban networks one day of uniqueness seems abundant.

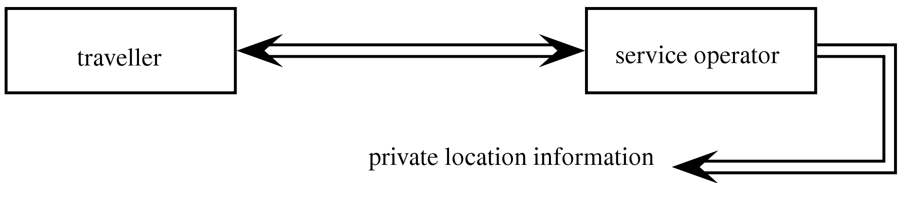

Figure 2.5: Server side data collection

This approach has the advantage of numerous data sources from the beginning without the need of building infrastructure. The acquired location information is less accurate then direct GPS information but still valuable. Accuracy of the location data depends on many parameters[11]. The persuasion and the stiff dependence on the mobile network operator are the main problems of this method.

Several operators are involved in such a research or product development around the globe [1, 5]. The actual project I participated in at Nokia Siemens Networks implements this approach, too.

2.2.3 Third party data collection

Third party data collection mechanism means to gain publicly accessible location and traveller identity information without the knowledge and attendance of the traveller.

Usually this approach requires to build infrastructure to reach such information. But the bulk of users from the start of the project can rectify the initial build costs.

In many countries, these information are publicly available nevertheless, the recording them and processing them bring up privacy issues [13].

One concrete method based on this approach that is in operation in many cities and highways is to place video cameras around the main network nodes that parse and record licence plates of vehicles. The data stream can be applied to real time traffic control [14] or in most cases for average speed limit enforcement [16].

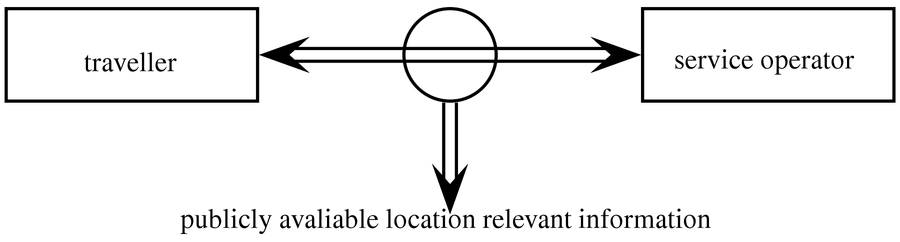

Figure 2.6: Third party data collection

Vysionics' RouteHawk product uses Automatic Number Plate Recognition (ANPR) for journey time measurement. This application plays elemental part in traffic decision support systems of many cities worldwide[23].

Another implementation is Swarco's Bluetooth based BLIDS. BLIDS receiver detects Bluetooth enabled devices, which are within reception proximity of the antenna for example handsets, laptops and hands-frees and records their unique Bluetooth addresses (MAC). The addresses are converted into anonymous user identification tokens and stored together with exact time stamps of the observations. It manages the observations into a database which can be analysed to calculate traffic parameters including travel time, origin-destination streams, traffic incidents [6].

Beyond Swarco another company utilising a similar Bluetooth based is BlipTrack. Their system is available to collect information from highways as well as urban networks including private and also public transportation [2].

2.3 Privacy

Location based services are applications utilising the information of subscribers' position. Worldwide there are several countries, including the states of European Union, where the local law enforces service providers to store and provide location data in case of emergency. Such information has a wide range of commercial application opportunities which funds a major new market for the telecommunication and telematics industry [13].

Among the huge amount of available private user information in cyberspace the location data differs in an intrinsic way bearing physical location of the subscriber. Therefore in mature countries law regularize the availability and usage of sensitive location information in varying clarity [13].

Since these services often may be implemented in a way that exposes sensitive personal information, there are several privacy issues to consider. A key question is: “Who should have access to what location information under which circumstances?”[17].

The ideal situation would be if the individual subscriber were aware and equipped with tools to directly control his own personal location privacy policies, subject to applicable rules and regulations [17].

Unlike other location based services that provide personalisation traffic parameter estimation purpose does not claim personal identification other than a single trip. In this way mathematics and cryptography offer a simple and efficient way to solve privacy issues (see Fig. 2.7).

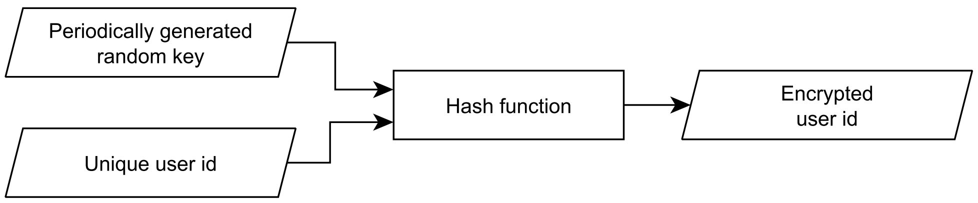

Figure 2.7: User id encryption

Travellers identification data may vary for trip to trip. A period from a few hours to a day is long enough to generate a new random encryption key depending on the ordinary trip length in the urban road network. These keys are used to salt a hash function applied on the travellers identification data. This undecipherable method grants privacy of travellers.

2.4 Traffic Applications of Radio Signaling Data

2.4.1 State of the Art

There are several finished and ongoing research projects according to location based services based on cellular networks. Two main directions are present affecting most of the research process. One portion of the articles approaches the topic from an academic viewpoint where the provided methods are more general but a real application is far from realisation. The amount of data to work on is limited due to privacy concerns. The other way of research projects is performed or funded by companies from the telecommunication industry. These papers provide valuable real life applications but no general concepts and the methods are protected by patents.

Origin-Destination Matrix

A straight application of various zone based location aware information in urban environment is the origin-destination matrix. The accuracy of location determination does not affect the estimation of origin-destination matrix and it is valuable at traffic design, traffic control and also commercial applications.

The paper of Caceres, Wideburg and Benitez about estimation of origin-destination matrix provides algorithms to proceed information from cellular network. A GSM network simulator was developed to test and validate the method. The article yields detailed examples only for highways [4].

Transport Research Laboratory of UK also analysed the possibility to gain origin-destination matrix from server side billing information of O_2 mobile operator in the Kingdom. The method was not detailed and was not intended to provide the data real time [8].

Route choice

The route choice of travellers is challenging in urban road traffic network. The route of a vehicle is hidden between each update of location information. A fortunate choice could enhance the quality as much as a blind guess could ruin the result.

The article of Yuan, Guan and Qiu offers ideas and algorithms for a wide range of traffic applications and challenges. The goal of the paper was a complex application to estimate various traffic parameters of the suburban of Beijing. GSM handover and location update zones were formed to gain server side location information and process it in multiple steps. GPS based field tests were carried out for testing and validation [31].

Tettamanti, Demeter and Varga wrote a paper about route choice based also on GSM handover and location update data. They applied Voronoi tessellation on map of GSM base stations acquired from the crowd-sourced OpenCellID database. PTV AG's VISUM software and advanced route assignment algorithm were utilised to estimate the most likely route of traveller between two districts of Budapest [19].

Travel Time Estimation

The paper of Ygnace, Drance Yim and Lacvivier from Berkley proves that cell phone equipped drivers can be used as traffic probes for travel time estimates on a road network. It depends on client side data collection and assumes that at least 5% of travellers on the highway network of San Francisco are equipped with a GPS capable cell phone. Based on the gathered information stream the travel time on the highway link network could be predicted by 95% accuracy [15].

One interesting idea of the report offers to combine and mix different sources of location information as for example GPS and cellular telephony network to improve the estimation of travel time. It is not further detailed in the paper [15].

Similar research projects where carried out in Vienna and Rome with the support of the mobile network operator, Telecom Italia. One paper identified the advantages and the limitations of current third generation cellular technology in intelligent transport applications[21], while the latter revealed a complex application integrating the consumer mobile terminals and on board units of professional drivers[5].

Vision

Prediction of human behaviour and movement is an old desire appearing in many literary works. The classic book series of Isaac Asimov titled *The Foundation* introduced the science of psycho-history. It states that above a predefined amount of people the behaviour is accurately predictable for long periods.

These books gave the idea for several topics to research that could be utilised in intelligent and effective traffic control.

Albert-László Barabási is a famous research professor in the field of linked networks and human mobility patterns. He and his colleagues participated in numerous articles and books to predicate movement patterns [7]. He used GPS based client side data collection, huge amount of collected data from cellular network [18] and also money tracking of WheresGeorge.com [1].

Governments and cellular network operator will be all affected and they will participate in estimation of mobility of the mass to control it and improve the efficiency of society. The question of privacy is still open but the attitude of young generations is slowly altering and the fabulous work of Orwell passes into oblivion [32].

2.4.2 Goals of the Thesis

This thesis has the following goals:

Large scale road networks A road network of complete city can be covered and traffic parameters could be extracted continuously from any segment of the network thus providing accurate and particular information on traffic of the whole city.

Combined data of various sources Many technologically different but methodologically similar data sources could be utilised and merged to produce a reliable and accurate estimation of traffic parameters. Different cellular providers are only the peak of the iceberg.

Reliable estimation Fault tolerance and stability are necessary for reliable and smooth estimations of traffic parameters to control traffic flow on urban and interurban road networks.

Chapter 3

Travel Time Estimation

3.1 Input specification

3.1.1 Location information

The above mentioned technologies have a common property, they provide location aware information from certain, a priori known sections of the road network. These sections are called *zones* and they have a bounded measurement error depending on the applied technology. The zones could be identified by some universal or technology dependent zone identifier named “zone id”.

The zones are constantly monitored and when a traveller passes a zone the traveller’s identifier (uid), the identifier of the zone (zone id) and the time stamp of the pass get handed to the traffic application.

$$\left\{ \begin{array}{rcl} \text{"}uid\text{"} & : & \text{"380561234567"} \\ \text{"}zoneid\text{"} & : & 1741 \\ \text{"}timestamp\text{"} & : & 1095379200 \end{array} \right\} \tag{3.1}$$

3.1.2 Data source

A *data source* provides real time information about subscribers consisting of a zone id, a unique subscriber identification value called “uid” and the time stamp when the subscriber passed the zone.

These data can be grouped by the uid and provide valuable travel information for each individual subscriber.

3.2 Kalman filter

The Kalman filter, also known as linear quadratic estimation (LQE), is an algorithm which uses a series of measurements observed over time, containing noise (random variations) and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those that would be based on a single measurement alone. More formally, the Kalman filter operates recursively on streams of noisy input data to produce a statistically optimal estimate of the underlying system state. The filter is named for Rudolf (Rudy) E. Kálmán, one of the primary developers of its theory [27].

The Kalman filter has numerous applications in technology. A common application is for guidance, navigation and control of vehicles, particularly aircraft and spacecraft.

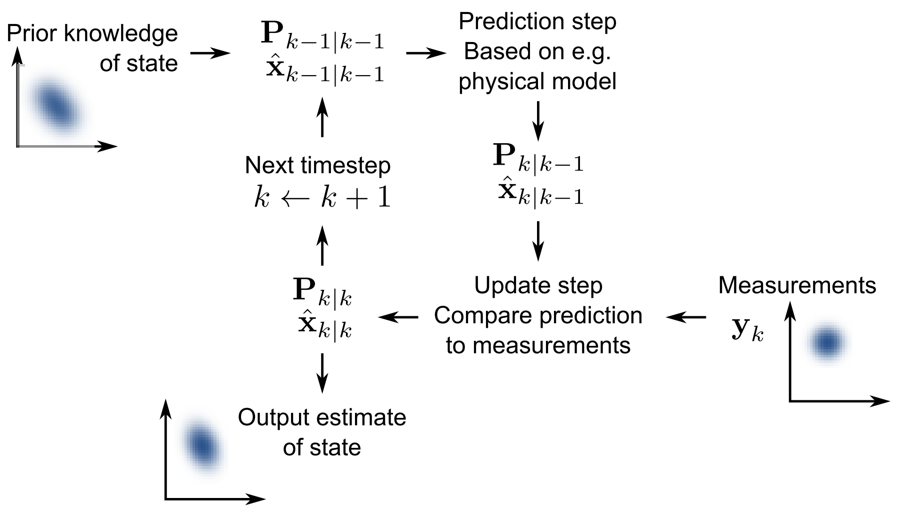
Prior knowledge of state
$\mathbf{P}_{k-1|k-1}$ $\hat{\mathbf{x}}_{k-1|k-1}$
Prediction step Based on e.g. physical model
$\mathbf{P}_{k|k-1}$ $\hat{\mathbf{x}}_{k|k-1}$
Next timestep $k \leftarrow k+1$
Update step Compare prediction to measurements
Measurements $\mathbf{y}_k$
$\mathbf{P}_{k|k}$ $\hat{\mathbf{x}}_{k|k}$
Output estimate of state

Figure 3.1: Basic concept of Kalman filtering

The algorithm works in a two-step process: in the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties. Once the outcome of the next measurement (necessarily corrupted with some amount of error, including random noise) is observed, these estimates are updated using a weighted average, with more weight being given to estimates with higher certainty. Because of the algorithm's recursive nature, it can run in real time using only the present input measurements and the previously calculated state; no additional past information is required.

From a theoretical standpoint, the main assumption of the Kalman filter is that the underlying system is a linear dynamical system and that all error terms and measurements have a Gaussian distribution (often a multivariate Gaussian distribution). Extensions and generalizations to the method have also been developed, such as the Extended Kalman Filter and the Unscented Kalman filter which work on nonlinear systems. The underlying model is a Bayesian model similar to a hidden Markov model but where the state space of the latent variables is continuous and where all latent and observed variables have Gaussian distributions.

3.3 Application of Kalman filter

The Kalman filters are based on linear dynamic systems discretized in the time domain. They are modelled on a Markov chain built on linear operators perturbed by Gaussian noise. The state of the system is represented as a vector of real numbers. At each discrete time increment, a linear operator is applied to the state to generate the new state, with some noise mixed in it and optionally some information from the controls on the system if they are known. Then, another linear operator mixed with more noise generates the observed outputs from the true ("hidden") state [27].

3.3.1 Underlying dynamic model

The general formula for the linear dynamic system assumed by the Kalman filter can be described by the 3.2 formula. The Kalman filter assumes that the k^{th} state of the system, denoted by x_k, evolves from the $(k-1)^{\mathrm{th}}$ state by multiplication of the possibly time dependent state transition matrix A_k and additional effect of control input u_k.

$$\underline{x}_k = A_k \cdot \underline{x}_{k-1} + B_k \cdot \underline{u}_k \tag{3.2}$$

At the presented application the Kalman filter is used only for state estimation thus there is no control input at all therefore the term $B_k \cdot u_k$ is abandoned.

My approach is quite straightforward way to represent a road network as a directed graph. The nodes of the graph are the junctions and the zones where location information could come from. The edges of the graph represent road connections between the nodes. The edges are weighted and the weight represents the travel time on the corresponding section of the network.

The states of the dynamic system are the time varying weights of the edges. In Eq. 3.3 $tt_{i,k}$ names i^{th} edges travel time at k^{th} discrete time step.

$$\underline{x}_k = \begin{pmatrix} tt_{1,k} \\ \vdots \\ tt_{n,k} \end{pmatrix} \tag{3.3}$$

A simple scheme was chosen for the dynamics of the system underlying the Kalman filter. Let the state-transition matrix, named A_k, be the constant identity matrix representing the idea that the travel times do not evolve through time by themselves.

3.3.2 Observation

At each time step the state of the dynamic system is updated by measurement. Equation 3.4 shows the general formula stating that a measurement, $\underline{y}_k$, consists of the linear combinations of the system states. C_k names the observation matrix which varies over time representing the actual observations and assumptions.

$$\underline{y}_k = C_k \cdot \underline{x}_k + v_k \tag{3.4}$$

The observation matrix differs for each update according to the current incoming observations from the various traffic data streams.

When trip information arrives it contains pairs of a location and a time stamp. The route of the subscriber is determined from the time ascending location data in some way. Then an appropriate observation vector could be constructed for each ascending pair of time stamp values by ones at the states representing the affected sections of the network and zeros for all other states.

These observation vectors can be arranged in an observation matrix.

The time differences between the time stamps are the values of the observations.

These observations contain v_k error which is bounded and can be assessed depending on the applied technology. Theory of Kalman filter assumes that error is white noise. The distribution of measurement in a handover zone is topic of an ongoing research process at the university. The preliminary results are suggesting that presuming white noise as the error of zones is justifiable [11].

If the data are rare or frequent update of the system states is preferred, one has to face the problem of a not observable system. For this reason I take few assumptions that help the estimation to gain stability.

3.3.3 Route creation

A key point of the estimation is that the incoming traffic information contains only places at specified date and does not indicate the actual route of the subscriber. There is an ongoing research in the topic of dynamic traffic assignment at the department [19].

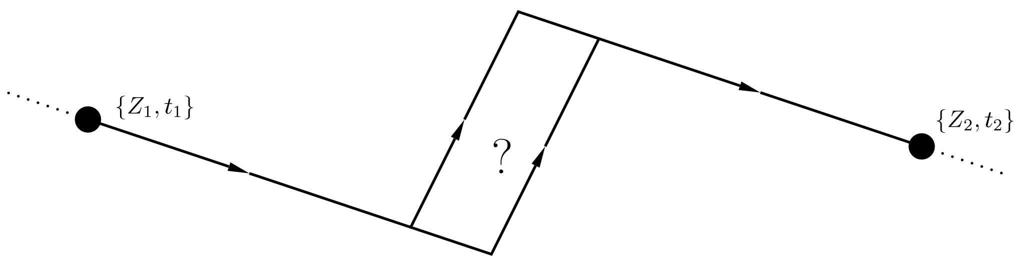

Figure 3.2: Route choice between two points of an observation

In this thesis I assumed that the zones where we can gain location data are so frequent to the junctions of the network that any other solution beside the shortest path on the graph is implausible. That way between two points the shortest path is chosen as the route of subscriber.

3.3.4 Intermediate uncertainty

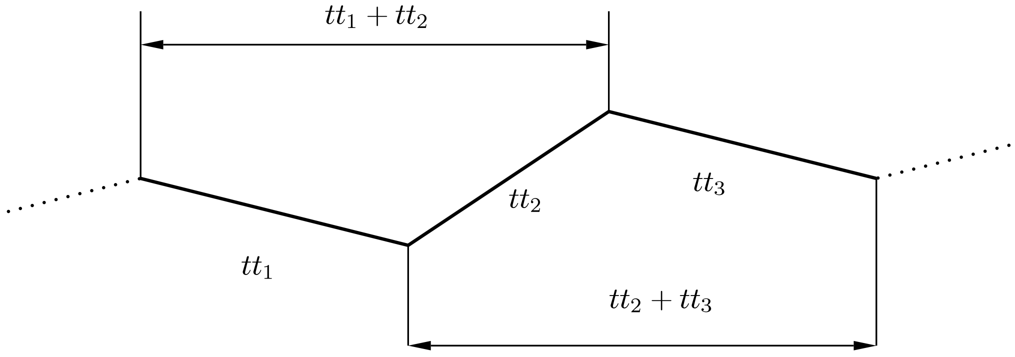

Figure 3.3: Three segmented road with two over-covering observations

For different data sources of various technologies, like GSM operators and number plate recognition systems, the roads between junctions are divided to sections for each zone. One subscriber does not provide exact travel times for each segment covered by observation but the sum of these.

The rare observations and the over-segmented road network leads to insufficient amount of information thus the system is unobservable and cannot be estimated by Kalman filter.

An assumption was chosen that states that each pair following segments' travel time rate sustain their free flow travel time rate. In the following equation $tt_{ff,i}$ denotes the i^{th} segment's free flow travel time and $tt_{real,i}$ the real travel time on the i^{th} segment.

$$\frac{tt_{ff,i}}{tt_{ff,j}} = \frac{tt_{real,i}}{tt_{real,j}} \tag{3.5}$$

Equation 3.5 can be rearranged to a valid row of the observation matrix.

$$tt_{ff,j} \cdot tt_{real,i} - tt_{ff,i} \cdot tt_{real,j} = 0 \tag{3.6}$$

$$\left(0 \cdots 0 \quad \overbrace{tt_{ff,j}}^{i^{\text{th}}} \quad 0 \cdots 0 \quad \overbrace{-tt_{ff,i}}^{j^{\text{th}}} \quad 0 \cdots 0) \right) \cdot \begin{pmatrix} tt_{real,1} \\ \vdots \\ tt_{real,n} \end{pmatrix} = 0 \tag{3.7}$$

$$\underline{C}_{i,j} \cdot \underline{x} = 0 \tag{3.8}$$

Where $\underline{C}_{i,j}$ denotes a row of the C observation matrix and $\underline{x}$ is the state vector of the system.

I add this assumption to the observation matrix for each update with a reasonable huge error variance.

The big error serves the purpose that the valuable observations are not affected by this assumption, but the system became observable and stable assuming homogeneous traffic alteration on the following segments of road network.

3.3.5 Incomplete data

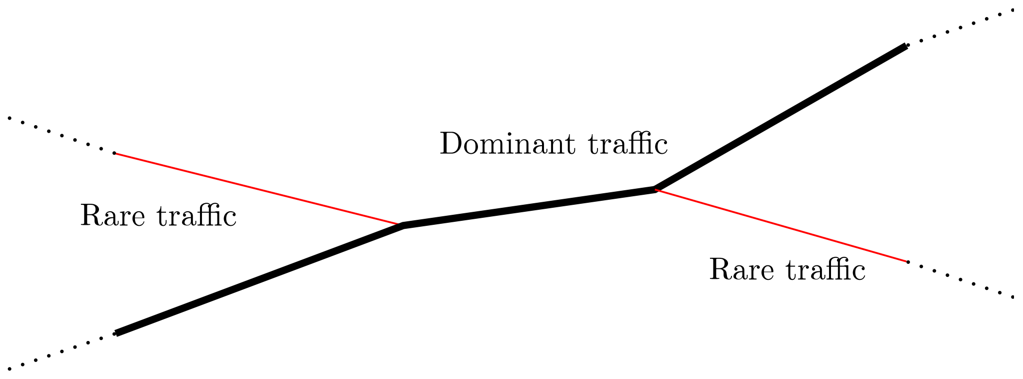

Figure 3.4: Rare traffic offers insufficient amount of observations

There is an open question of rare traffic and thus the insufficient number of observation on parts of the road network. The lack of observations could be just for a short period cause of coincidence or for longer period due to rare traffic. In the previous case the traffic on affected distance is likely not to change much in essence. The latter case is handled, according to fundamental diagram shown in Fig. 3.5, as the traffic is weak therefore the speeds and travel times of cars passing is about free flow.

A feasible assumption could be that if there is weak or no traffic on a road the travel time of the link is its default free flow travel time as the fundamental diagram of traffic flow suggests.

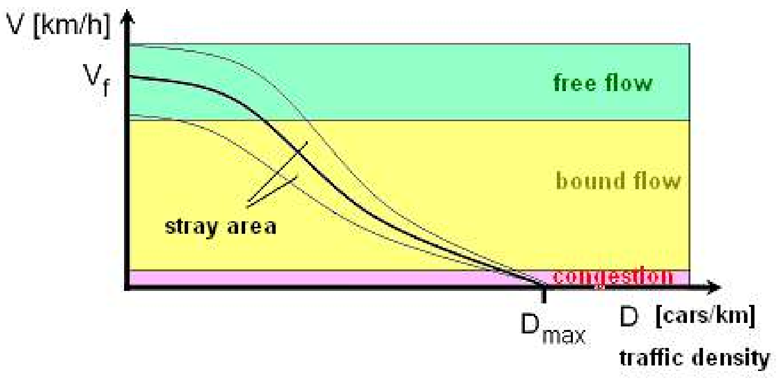

Figure 3.5: Fundamental diagram of traffic flow

For the first case keeping of previous values is needed. For the second case the constant free flow values should be set. Covering both cases power function was chosen to keep approximately to 1 and later transit to 0. The function is displayed in Fig. 3.6 for multiple exponents.

$$f(x) = \begin{cases} 1 & , x \leq 0 \\ 1 - \left(\frac{x}{t_{transient}}\right)^{exponent} & , 0 < x \leq t_{transient} \\ 0 & , t_{transient} < x \end{cases} \tag{3.9}$$

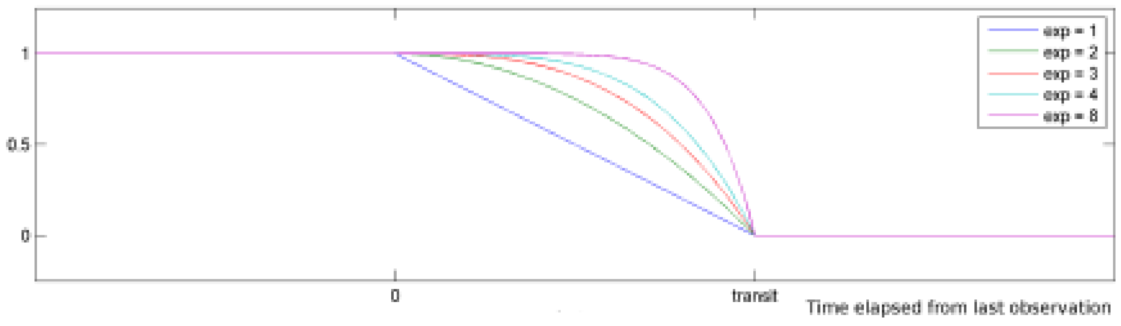

Figure 3.6: Transient function to free flow travel time value

This approach leads to the point where a few rare exclusions of the section from the observation does not imply the state to fall far from the previous values but for weak traffic the estimated value keeps to the free flow value.

Chapter 4

Validation

4.1 Matlab environment

Matlab, stands for matrix laboratory, is a numerical computing environment and a special purpose programming language developed by MathWorks from 1984. Matlab allows matrix manipulations, plotting, implementing user defined functions and much more. Matlab has a complex, built in object oriented type system. A software for Matlab can be written as a standalone application or can be run from an interactive command line.

GNU Octave is a high level numerical computing oriented interactive command line environment or interpreter. Octave code resembles Matlab code, because Octave is developed to be code compatible with Matlab. Octave has limited support for object oriented development compared to Matlab. Octave is OSI approved free software released under General Public Licence (GPL), which guarantees that the source code of Octave remains available and modifiable for every user of it.

My intention was to create the demonstration code in a way that does not strictly require the commercial Matlab software but is runnable on both Matlab and Octave. Because of Octave's limited support to object oriented discipline I wrote the demonstration code according the rather old fashion procedural programming paradigm.

4.1.1 Framework

At first there was a proof of concept code to show that the idea is feasible and has worth it and feasible to give it a try. Later a framework was developed to test flexibly various exchangeable modules for each function and feature.

Due to Octave's limited support for object oriented development the written software framework follows the imperative programming paradigm mixed with functional elements. The essential and common values are stored in global variables, the more specific variables are passed as function arguments and return values.

The framework, displayed in Fig. 4.1, has several global variables and commutable functions described in the following subsections.

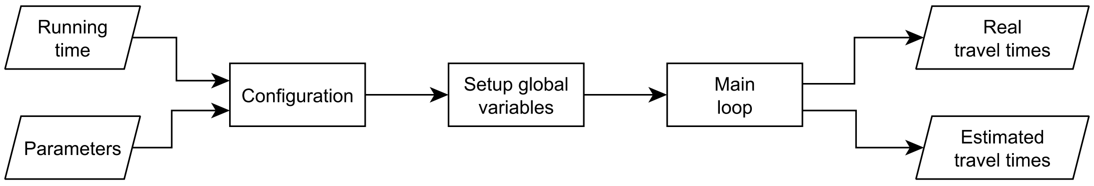

Figure 4.1: Rough flowchart of estimation test framework written for Matlab

The starting file of the framework is `main.m`. It is a Matlab function, call it as `main(300)` from the Matlab command line where the parameter is the duration of estimation.

Road network representation

The road network is represented as a directed graph. The vertices of the graph are the zones and the junctions of roads, the edges between vertices indicates directed straight road connection between two vertices.

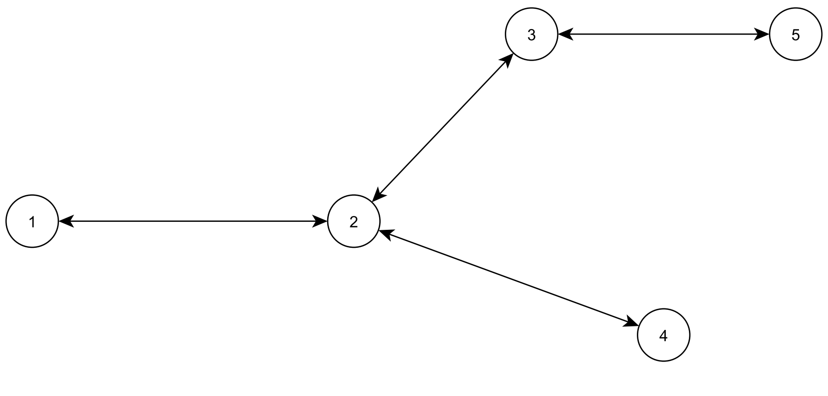

Figure 4.2: Sample road network

The graph is represented as its adjacency matrix where the weights of edges are the free flow travel times on the edges. Free flow travel time can be obtained by calculation or measurement. For the validation it was calculated from the length and speed limit of the segment.

Figure 4.2 shows a sample directed graph with the following adjacency matrix with unit weights for simplicity:

$$\mathcal{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

The framework has the `glGraph` global variable to hold the adjacency matrix. The travel times on the edges are represented as a vector. The edges are numbered depending on their position in the adjacency matrix. These numbers are used to retrieve the value from the vector for each edge. `glDefaultEdgeWeights` is a global vector having the initial free flow travel time values of the edges of the graph similar way as the vectors holding estimated and actual travel times on edges.

For shortest path calculation the Dijkstra algorithm was used coded by Michael G. Kay[9]. The function runs on the whole graph and its output is cached in the global variable `glShortestPathNeighbours`. The function needed a slight modification to become usable by Octave, the logical expressions were changed to lazy evaluation.

Estimation update loop

The main loop of the framework starts with initializing the loop variable, the elapsed time. The initial value of estimated travel time vector is the free flow travel time vector.

Figure 4.3 shows the detailed flowchart of the main loop. The process flows top down. Used variables are on the left of each function block and produced values on the right. The dashed arrows show data flow.

There are three function calls in the main loop, each configurable. These are `nextState` for the actual travel time, `getMeasurement` to get the observations, the observation matrix and the errors of the observations and `getEstimation` to do the estimation based on the observations.

Configuration

The main function of the framework calls a script first to set up the global variables, create graph and set the above mentioned three functions. Matlab's lambda function was used to replace functions in the framework. The syntax of the lambda function is described in Listing 4.1.

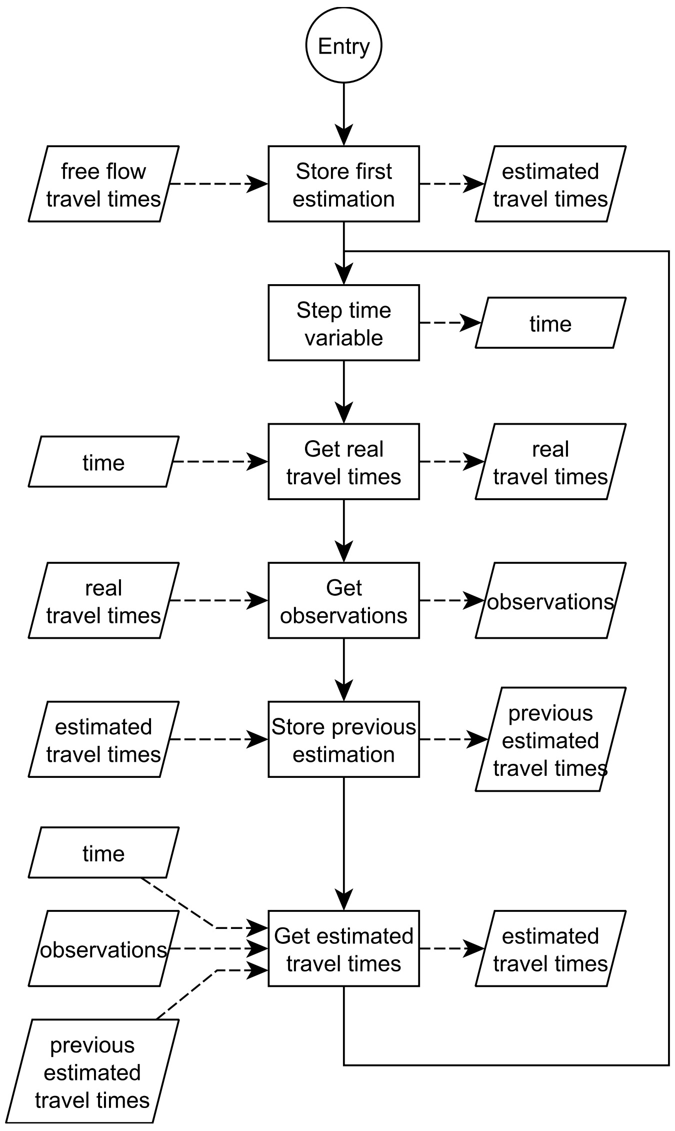

Figure 4.3: Detailed flowchart of the framework's main loop

Listing 4.1: Two argument lambda function in Matlab for multiplication

```
f = @(x,y) x*y;
```

Listing 4.2 shows the used settings script for an alternating route road network. The first row of the script calls a function to get the graph representing the road network. Then the global variables are set up. After that a global variable, `glSimulationStep`, is used to parametrize the functions for the framework.

The `nextState` and `getMeasurement` function variables are filled with corresponding csv reader functions, the former one with interpolation capabilities because the estimation loop is more frequent than the registered values in the csv file.

Listing 4.2: Settings script for alternative routes

```
Graph=alter();
```

```
setupGlobals(Graph);

global glSimulationStep;

nextState = @(state,time)
        interpolateCSV('real.csv',';',time,20,...
                [4 5 1 2 3 6 7 8]);
getMeasurement = @(graph, state, time)
        readFromCsv('vissim_alter_observation.csv',...
                ',',time,glSimulationStep);
getEstimation = @(graph,prevEstim,default,...
                obsMx,observation,qualities,time)
        KalmanWithAssumptions(graph,prevEstim,...
                default,obsMx,observation,qualities);
```

Parameters

There are a few parameters to tune the estimation. They are set in file `setupParameters.m` as global variables.

`glSimulationStep` is the parameter for frequency of the update loop. This is a time duration in seconds representing the interval between updates.

`glVarianceOfRatioEstimation` parameter sets the relative weight of artificial observations created by ratio keeping assumption compared to the real measurements.

`glVarianceOfNoDataEstimation` parameter represents the relative weight of free flow travel time assumption for weak traffic case in the Kalman filter.

`glExponentOfSpeedToFreeFlowEstimation` is a parameter setting the shape of transient function in free flow travel time assumption. For value 1 the function is piecewise linear.

`glTimeToFreeFlowIfNoData` represent the transient duration in seconds for free flow travel time assumption.

`glRandomMeasurementNumberOfMeasurementsInEachTurn` is a parameter for random observation generation, it shows that how many observations should be generated for each update.

`glRandomMeasurementDecreaseNum` and `glRandomMeasurementIncreaseNum` are parameters for the random travel time state process. They give the interval around actual state where the new state is chosen from.

Global variables

There are several global variables that each module can reach. They contain common data that are used frequently in many modules and functions or parameters tuning the working of the framework or just cache some data that is tiresome to reload on each function call like csv files.

`glGraph` stores the graph that the framework is working on.

`glShortestPathNeighbours` is a cached data for generating shortest paths on the graph. It is the result of the function `dijkstra(glGraph,[],[])` that computes shortest paths between each pair of vertices.

`glNumberOfVertices` and `glNumberOfEdges` store data about the graph. They are used for loops and boundary checks.

`glEdges` is a helper matrix resembling the adjacency matrix but contains the ordinal number of each graph instead of the edge weights.

`glDefaultEdgeWeights` stores the initial, free flow travel times for each edge read from the adjacency matrix.

`glTime` is the actual time in the simulation.

`glTimeOfLastData` stores the time ticks for each edge when it was updated from data based on real observations last time.

4.1.2 Estimation

Kalman filter

Matlab has the Kalman filter based linear quadratic estimator included. The reference of the function is in Listing 4.3. The function calculates the state estimation for the dynamic system described in Eq. 4.1 and 4.2. The matrices `sigw` and `sigv` denote the covariance matrices of noise vectors w and v. In this case `sigv` stores the variances of observations and assumptions in a diagonal matrix. The result is the gain matrix `L` that can be used for estimation as described in Eq. 4.3.

Listing 4.3: dlqe function in Matlab

```
function [L, m, p, e] = dlqe (A, G, C, sigw, sigv)
```

$$x_{k+1} = \mathcal{A} \cdot x_k + \mathcal{B} \cdot u_k + \mathcal{G} \cdot w_k \tag{4.1}$$

$$y_k = \mathcal{C} \cdot x_k + \mathcal{D} \cdot u_k + v_k \tag{4.2}$$

$$z_{k|k} = z_{k|k-1} + L\left(y_k - \mathcal{C} \cdot z_{k|k-1} - \mathcal{D} \cdot u_k\right) \tag{4.3}$$

A simple wrapper script, shown in Listing 4.1.2, was written around `dlqe` to do the latter calculation and give back only the estimated state vector.

```
function estimation = KalmanFilter(...
   graph,prevEstim,default,obsMx,observation,qualities)

   I = eye(length(prevEstim));
   [Lk,m,P,e] = dlqe(I,I,obsMx,I,diag(qualities));
   estimation = prevEstim + Lk*(observation-obsMx*prevEstim);

end
```

The identity matrix `I` was used as state transition matrix $\mathcal{A}$, noise matrix $\mathcal{G}$ and noise covariance matrix `sigw`.

Octave does not include `dlqe` function in default installation. The “control” package from Octave-Forge includes the `dlqe` function among other control theory oriented functions.

Kalman filter with assumptions

The observation matrix denoted by $\mathcal{C}$ in Eq. 4.2 varies for each update. Thus it is a common situation the dynamic system is unobservable, the states of the system are not well defined based on the actual observations.

Listing 4.4: Kalman filter with assumptions function code

```
function estimation = KalmanWithAssumptions(graph, ...
        prevEstim,default,mesMx,measurement,mesQ)
    [corMx, correction, corQ] =
        getCorrections(graph,default,mesMx,prevEstim);

    ObsMatrix = vertcat(mesMx, corMx);
    Observation = vertcat(measurement, correction);
    Qualities = vertcat(mesQ,corQ);

    estimation = KalmanFilter(graph,prevEstim, ...
        default,ObsMatrix,Observation,Qualities);
end
```

The function in Listing 4.4 gets the following input parameters: the graph adjacency matrix, the previously estimated states, the free flow travel times, the observation matrix and corresponding travel times and their variances. The output of the function is the estimated state of the system at the time.

At first the assumptions are calculated based on the graph, the free flow values, previous estimation and actual observations by the function `getCorrections`. It is detailed in Listing 4.5 The result is concatenated to the actual observations and the basic Kalman filter wrapper function is called.

Listing 4.5: Source of getCorrections function

```
function [corMx, correction, corQ] = getCorrections(...
        graph, default, mesMx, prevEstim)

[ffMx, ff, ffQ] = defaultIfNoData(...
                graph,default,mesMx,prevEstim);
[ratMx,Z,ratQ] = keepDefaultRatio(graph);

corMx = vertcat(ffMx, ratMx);
correction = vertcat(ff,Z);
corQ = vertcat(ffQ,ratQ);

end
```

The results of the two assumptions of free flow rate and no data free flow are concatenated and returned to the caller function.

Free flow in case of scarce data

For the case of rare traffic and scarce observations a transient function was defined to continuously move the estimated travel times of the affected road segments towards their free flow travel times.

After starting with the initialization of a bunch of global variables the function checks if there are any observations. If not it returns with the free flow values for each segment. Otherwise it sums the observation matrix by segments and checks in a loop for each segment that whether it is included in any observations. If so it updates the corresponding element of the vector `glTimeOfLastData` otherwise calculates a transient value based on the transient function defined in Section 3.3.5.

Listing 4.6: Function to transit estimated travel time to free flow value

```
function [corMx, correction, corQ] = ...
   defaultIfNoData(graph,default,mesMx,prevEstim)

```

```
global glVarianceOfNoDataEstimation;
global glExponentOfSpeedToFreeFlowEstimation;
global glNumberOfEdges;
global glTimeToFreeFlowIfNoData;
global glTimeOfLastData;
global glTime;
global glSimulationStep;
len = glNumberOfEdges;

d = glTimeToFreeFlowIfNoData;

exponent = glExponentOfSpeedToFreeFlowEstimation;
Quality = glVarianceOfNoDataEstimation;

   if isempty(mesMx)
      [corMx, correction, corQ] = ...
         defaultEstimation(graph,default,Quality);
   else
      corMx = [];
      num=0;
      correction=[];
      for k = [1:len;sum(mesMx,1)]
         if k(2)==0
            line = [zeros(1,k(1)-1) 1 zeros(1,len-k(1))];
            corMx=[corMx;line];
            t = glTime-glTimeOfLastData(k(1));
            pt = t - glSimulationStep;
            if t < d
                  trans = (1-(t/d)^exponent);
                  transPrev = (1-(pt/d)^exponent);
                  correction=[correction; ...
                     (prevEstim(k(1)) - default(k(1))) ...
                     /transPrev * trans + default(k(1))];
            else
               correction=[correction; default(k(1))];
            end
            num = num + 1;
            else
                glTimeOfLastData(k(1)) = glTime;
            end
        end
        corQ = Quality*ones(num,1);
    end
end
```

The transient function is calculated in 32nd-36th lines of Listing 4.6. Variable `d` denotes $t_{transient}$ in the code. The formula is explained by Eq. 4.4 and Fig. 4.4 where *exponent* and $t_{transient}$ are parameters, $t_{current}$ and $t_{previous}$ denote the time stamp of the current and previous update cycle, t_{last} denotes the time stamp of last valid observation for the corresponding segment, $Z_{previous}$ stores the value of previous estimation of travel time and $TT_{freeflow}$ is the default free flow travel time value for the segment. `trans` and `transPrev` are the transient function values at the current and previous time stamps.

$$Z_{current} = \frac{1 - \left(\frac{t_{current} - t_{last}}{t_{transient}}\right)^{exponent}}{1 - \left(\frac{t_{previous} - t_{last}}{t_{transient}}\right)^{exponent}} \cdot (Z_{previous} - TT_{freeflow}) + TT_{freeflow} \qquad (4.4)$$

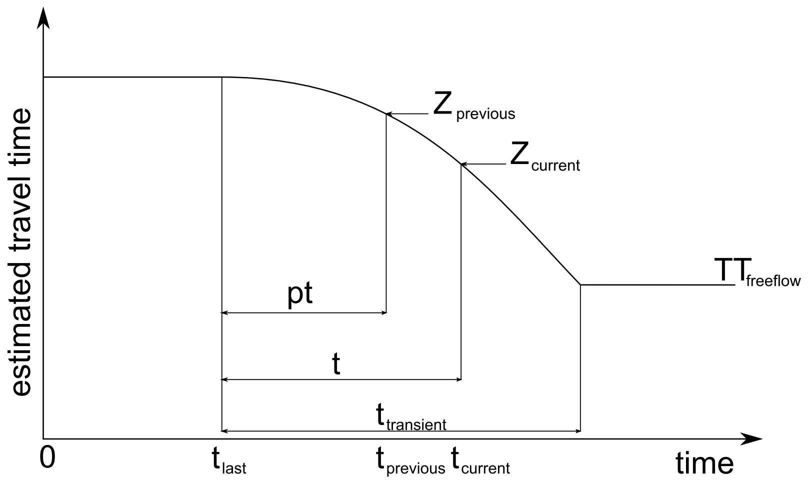

Figure 4.4: Current and previous estimation on transient function

Keep free flow ratio

The function `keepDefaultRatio` processes a graph adjacency matrix, lists all combinations of following directed edges and generates an observation matrix of them to keep their free flow travel time rate.

The function, listed in Listing 4.7, started with reading global variables about the graph. Then the simple inline function `genLine` was defined as a lambda function to generate a row of observation matrix from two edges according to Eq. 3.7.

The following edges of the graph was determined by a simple algorithm listing all three combinations of vertices in the graph and checking if they are neighbours. If so the function `genLine` generates a row appended to the observation matrix. Finally the constant tags are set to zero and the variance vector to a predefined constant value.

Listing 4.7: Function to give free flow rates of following segments as observations

```
function [corMx, correction, corQ] = keepDefaultRatio(graph)
global glEdges;
global glNumberOfVertices;
global glNumberOfEdges;
global glVarianceOfRatioEstimation;

genLine=@(A,B,A_w,B_w) ...
   [zeros(1,A-1) B_w zeros(1,B-A-1) ...
      - A_w zeros(1,glNumberOfEdges-B)];

corMx=[];
for vertex=1:glNumberOfVertices
   for from=1:glNumberOfVertices
      from_edge_w = graph(from,vertex);
      if(from_edge_w~=0)
         for to=1:glNumberOfVertices
            to_edge_w=graph(vertex,to);
            if(to_edge_w~=0)

               from_edge = glEdges(from,vertex);
               to_edge = glEdges(vertex,to);

               line = [];
               if from_edge < to_edge
                  line = genLine(from_edge, ...
                     to_edge,from_edge_w,to_edge_w);
               elseif to_edge < from_edge
                  line = genLine(to_edge, ...
                    from_edge,to_edge_w,from_edge_w);
                end

                corMx = [corMx; line];
            end
        end
    end
end
end

correction = zeros(size(corMx,1),1);
corQ = ones(size(corMx,1),1) ...
    * glVarianceOfRatioEstimation;

end
```

4.1.3 Visualisation

Three functions were written to handle plotting easily and efficiently.

The function `plotRealEstimation` takes the real and estimated travel time matrices returned by the framework and a number representing one section and plots that one segment's real and estimated states over time.

The function `plotInMatrix` takes the real and estimated state matrices and a subfigure layout matrix representing where and which segment to plot on a complex plot window. The sub plot layout matrix can be easily generated by function `figureLayout` . The function `plotInMatrix` results a plot like Fig. 4.5.

4.2 Simulation based on random process

For testing of the idea and various parts of the framework at first a random state transient and measurement model were set up.

4.2.1 State as random process

A simple solution to simulate a random process is a Markov process. The implementation is simple the next state depends only on the current one. The simulation takes the upper and lower bound around the current state between the next state

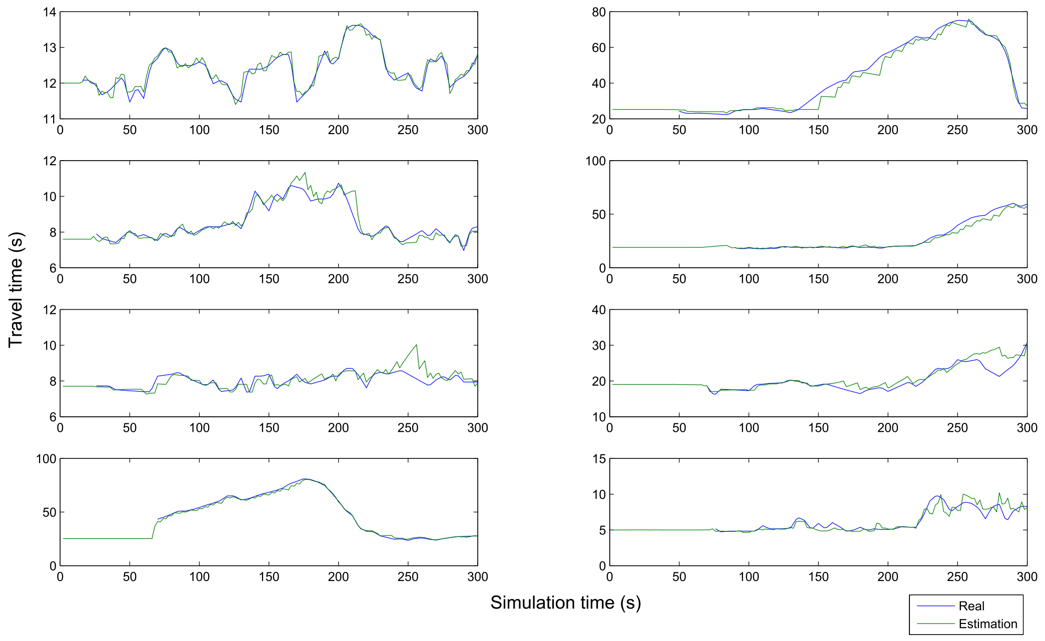

Figure 4.5: Output of `plotInMatrix` function

is chosen by uniform distribution. It is a continuous state discrete time Markov process, a random walk formulated in Eq. 4.5.

$$X_{k+1} \sim UNI\left(X_k + LowerBound, X_k + UpperBound\right) \tag{4.5}$$

This approach is simple to implement but does not follow real traffic variation. As can be seen in Fig. 4.6 the fluctuations of travel times is not realistic but may be suitable for testing purposes.

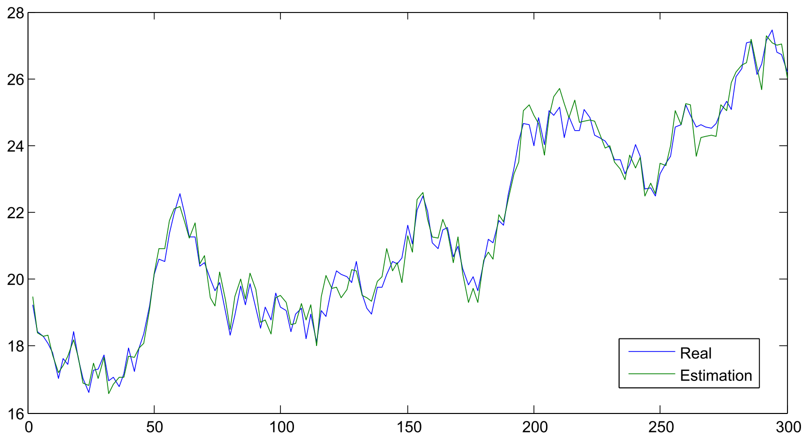

Figure 4.6: Random process as travel time of a segment plotted over time

The built in function `rand` was used and wrapped in `noiseUni` function to fit in the framework.

4.2.2 Random observation

The observations could be generated randomly from the known state of the road network. The function `randomMeasurement` was created for the purpose as a part of the framework. It takes the graph and the current travel times as input arguments and the number of observations to generate as parameters. It can add some noise to the generated observations.

The function calls multiple times the function `randomRoute` to get a random route on the graph. The source of `randomRoute` is contained in Listing 4.8.

Listing 4.8: Random route generation in graph

```
function obsVector = randomRoute(graph)
global glShortestPathNeighbours;
global glNumberOfVertices;
global glEdges;
global glNumberOfEdges;

    len = glNumberOfVertices;

    a = randi(len);
    b = randi(len);
    while a==b
        b=randi(len);
    end

    start = min(a,b);
    finish = max(a,b);

    obsVector = zeros(1,glNumberOfEdges);

    nextVert = finish;
    prevVert = glShortestPathNeighbours(start,finish);
    while prevVert~=0
        obsVector(glEdges(prevVert,nextVert))=1;
        nextVert = prevVert;
        prevVert = glShortestPathNeighbours(start,nextVert);
    end
end
```

Function `randomRoute` chooses two random vertices uniformly distributed from the graph and tries to create the shortest path between the pair of vertices utilising the previously calculated matrix by Dijkstra algorithm. If the route cannot be calculated a new pair of vertices are chosen and so on.

4.3 Simulation by VISSIM

VISSIM is a leading microscopic simulation program for multi-modal traffic flow modelling developed by the German PTV AG. With its unique high level of detail it accurately simulates both urban and highway traffic, including cyclists and motorized vehicles. It can simulate varying traffic on road networks. Figure 4.7 shows the typical user interface of VISSIM.

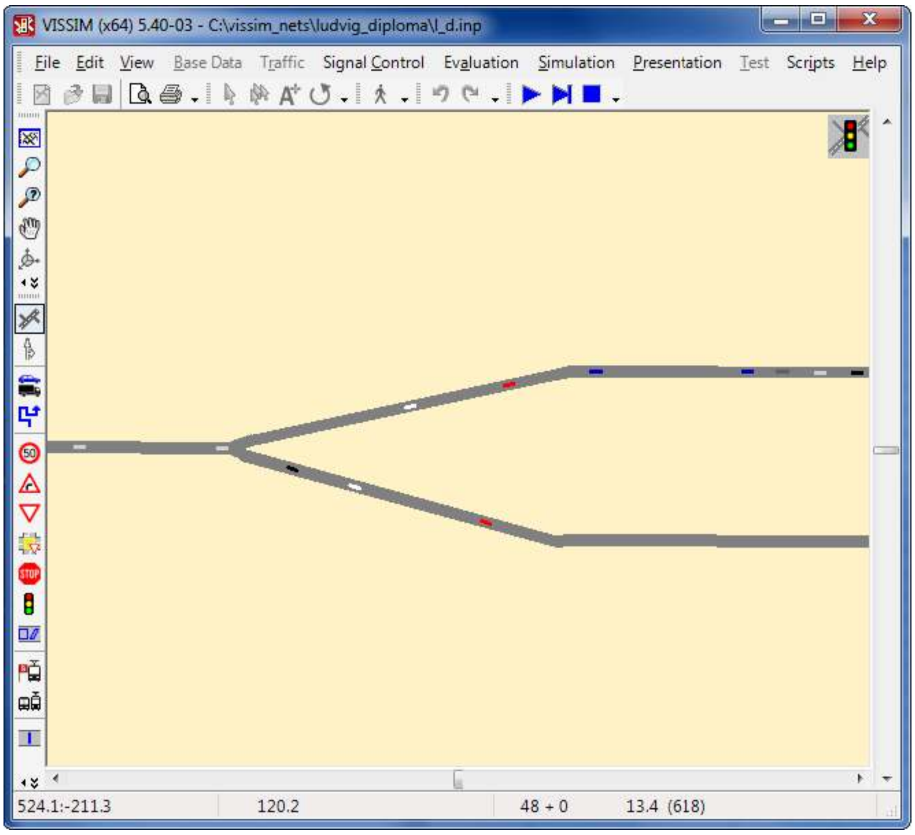
VISSIM (x64) 5.40-03 - C:\vissim_nets\ludvig_diploma\l_d.inp
File Edit View Base Data Traffic Signal Control Evaluation Simulation Presentation Test Scripts Help
524.1:-211.3
120.2
48 + 0
13.4 (618)

Figure 4.7: VISSIM graphical user interface

4.3.1 Model

The road network shown in Fig. 4.8 was modelled in VISSIM. At the red marks time dependent speed limits are used as traffic disturbance to simulate an accident caused congestion. The connectors between links marked blue in Fig. 4.8 function as zones

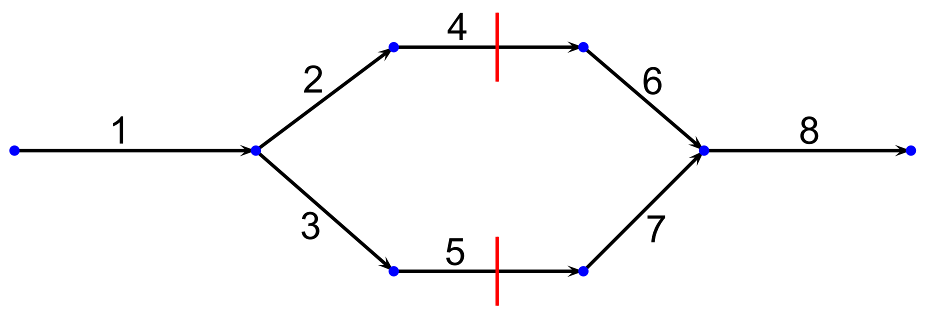

Figure 4.8: Simulated road network

in the simulation. Their size vary between 5 and 20 meters. Table 4.1 summarises the length, speed limit and free flow travel time for each link.

Table 4.1: Segment properties of the road network

id	length	speed limit	free flow travel time
	m	km/h	s
1	170	50	12.24
2	108	50	7.78
3	110	50	7.92
4	407	50	25.70
5	407	50	25.70
6	266	50	19.15
7	265	50	19.08
8	69	50	4.95

The simulation has a 300 seconds duration and is stepped by 0.1 seconds. The speed limit is set to $50km/h$ on the whole network but at the marks for a short period. On the upper segment, the left one according to heading, the section marked red has a $12km/h$ speed limit restriction from the 30th to 120th seconds of the simulation. The lower segment, the right one, has a similar speed limit restriction starting at the 60th and lifted at the 150th seconds as displayed in Table 4.2.

Table 4.2: Temporal speed limits on links

link	start	end	speed limit
	s	s	km/h
4	30	120	12
5	60	150	12

4.3.2 Evaluation tools

VISSIM provides wide range of tools to evaluate the simulation, supports programmable COM interface to access the most details or allows extension via application programming interface.

Two built in standard tools were used to gain raw data from simulation.

Travel times

"Each section consists of a start and a destination cross section. The average travel time (including waiting or dwell times) is determined as the time required by a vehicle between crossing the first (start) and crossing the second (destination) cross sections."[22]

"During a simulation run, VISSIM can evaluate average travel times (smoothed) if travel time measurement sections have been defined in the network. Via EVALUATION – WINDOWS, the travel time records can be displayed in a window on the desktop. They can also be stored as an output file if option Travel time is ticked via EVALUATION – FILES." [22]

One travel time measurement section is placed along each link in the road network and the measurement result is saved to file with extension "rsz".

Listing 4.9: Sample file of travel times[22]

```
Table of Travel Times

File: C:\VISSIM520\Examples\Demo\LRT-Priority.LU\lux3_10.inp
Comment: Luxembourg, SC 3-10
Date: Wednesday, May 13, 2009 12:26:34 PM
VISSIM: 5.20-00 [18711M]

No. 1: from link 1 at 0.6 m to link 9 at 19.0 m, Distance 18.4 m
No. 2: from link 2 at 298.4 m to link 3 at 33.8 m, Distance 106.8 m

          Time;    Trav;#Veh;        Trav;#Veh;
          VehC;    All;;             All;;
          No.:;    1;1;              2;2;
           60;     132.6; 49;        142.0; 219;
          120;     134.6; 61;        140.4; 249;
```

Travel time measurement tool was developed to measure longer periods of simulation. VISSIM averages the travel times of travellers passed the link in the actual time period thus the actual travel time cannot be updated too often. Five seconds update frequency was chosen to update the values and intermediate values are calculated by linear interpolation.

Frequent update is required to compare the measured values to the estimated travel times. One goal of the estimation application is quick, on-line evaluation of radio signaling data to aid real-time traffic control.

The resulting ";" delimited file can be processed by Matlab's `dlmread` function. The wrapper Matlab function is named interpolateCSV.m to handle travel time measurement as real values to compare estimation to.

Vehicle record

VISSIM offers Vehicle record tool to save the desired properties of vehicles to file. The required information is the vehicle's identification number, the time stamp of the simulation, the link which the vehicle is located on and the position of the vehicle on the link. The measurement results in a file with extension "fzp".

Listing 4.10: Sample file of vehicle record

```
Vehicle Record

File:        c:\vissim_nets\ludvig\l_d.inp
Comment:
Date:        Friday, May 11, 2012 2:03:27 PM
VISSIM:      5.40-03 [33277]

VehNr : Number of the Vehicle
t     : Simulation Time [s]
Link  : Number of the Active Link
x     : Link Coordinate [m] at the end of the simulation step

    VehNr;       t;   Link;      x;
        1;     3.1;      3;    1.1;
        1;     3.2;      3;    2.5;
        1;     3.3;      3;    4.0;
```

The file was slightly modified and imported to an SQLite database. After some filtering, observations were formed from pairs of following location information about each vehicle. The observation table has among others "link", "time stamp" and "travel time" fields. The table was exported to a csv file and used with Matlab's `dlmread` function. I created a wrapper function to use observations in the Matlab framework for estimation named readFromCsv.m.

4.4 Simulation result

There are three road networks to test and tune the estimation framework. Two of them has simulated road traffic from VISSIM and all three can be used also with random state process.

The main network to test the estimation is presented in Fig. 4.8. It represents two alternative routes and congestions on each route at different times.

To qualify the different parameter values the relative standard deviances (RSD) were calculated of the states by formula 4.6. $tt_{est,i}(j)$ denotes the estimated travel time for the i^{th} segment in the j^{th} time tick. $tt_{real,i}(j)$ defines the real travel times on segments similarly to the estimated travel times.

$$RSD_i = \sqrt{\frac{\sum_{j=0}^{n}\left(\frac{tt_{est,i}(j)-tt_{real,i}(j)}{tt_{real,i}(j)}\right)^2}{n}} \tag{4.6}$$

The RSD was calculated by Matlab's built in function `nanmean` according to Listing 4.11. `R` and `E` denotes the real and estimated travel times for each segment and time tick in a matrix form. Because real travel times are not available for all time period was used the `NaN` version of the `mean` function to ignore "not a number",`NaN` values.

Listing 4.11: Calculation of RSD value for all states

```
sqrt(nanmean(((R-E)./R).^2,1))
```

4.4.1 Update frequency

The update interval is highly dependent on the segmentation of the road network and the amount of observations over time.

The longer period of update results smoother, more accurate travel time values but the shorter one provides lower accuracy and more immediate, spot values applicable for quick decision making.

Several values were tried as update frequency. Figure 4.9 displays the real travel time on segment 1 from the 200th second till the 245th second and the estimated travel times for $1sec$, $2sec$, $5sec$ and $10sec$ update intervals.

As can be seen the estimations with $1sec$ and $2sec$ update period followed immediately the real state, the other two were significantly slower but smoother. The estimation with $1sec$ update interval is fragmented due to scarce amount of observation for each update but the one with $2sec$ seems to be smooth enough.

The relative deviations of the estimations of the various tests can be seen in Table 4.3.

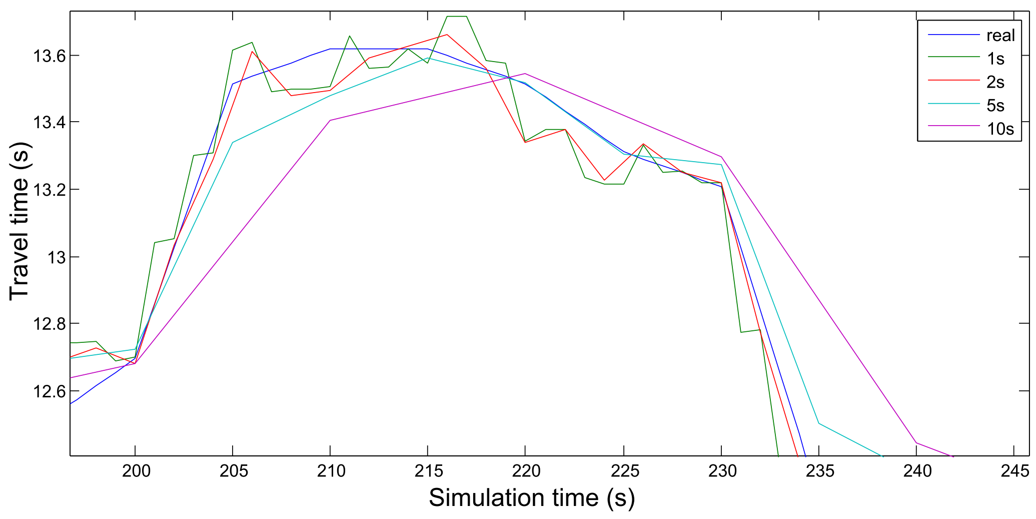

Figure 4.9: Effect of different update periods on the estimation

Table 4.3: Deviance of estimation for all segments with varying update frequency

update	RSD of estimation on segment							
interval	1	2	3	4	5	6	7	8
1*sec*	0.0115	0.0487	0.0787	0.0442	0.0783	0.0981	0.1227	0.1037
2*sec*	0.0102	0.0352	0.0463	0.0287	0.0696	0.0639	0.0891	0.0972
5*sec*	0.0101	0.0287	0.0260	0.0411	0.0780	0.0655	0.0809	0.1060
10*sec*	0.0231	0.0391	0.0336	0.0656	0.0998	0.0836	0.0835	0.1017

The real state function values are interpolated for each second, it was calculated just for every fifth second and is under sampled.

4.4.2 Variance of assumptions

The assumptions discussed in Section 3.3.4 and 3.3.5 are implemented as fake observations. Lots of artificial observations are generated and passed to the Kalman filter for each update cycle to gain observability of the dynamic system and amend the estimation of states. These observations are concatenated to the real, incoming observations and are treated as real ones.

A real observation is weighted by technology dependent variance of error. The Kalman filter uses the variance in the estimation process as some kind of weight. The observations with small variance means more accurate value thus has a greater weight at estimating the real state.

The variances of fake observations are parameters of the estimation process. As they are artificial observations the value of variances is freely eligible.

The variances should be large enough not to contradict the real observations but small enough to be error prone and stable.

Free flow rate keeping

The assumption generates the observation

$$tt_{ff,j} \cdot tt_{est,i} - tt_{ff,i} \cdot tt_{est,j} = 0$$

also discussed in Eq. 3.6. The variables are the estimated values $tt_{est,i}$ and $tt_{est,j}$ in the equation. The constant term is 0 indicating that the difference of rates of estimated and free flow travel times of following segments should be minimal.

The physical dimension of the constant term of the equation is sec^2 since the equation represents the product of travel times.

Take the difference of the two sides of the equation as a random variable that should be estimated. The approach of Kalman filter requires that the random variables are of normal distribution. The assumption states that the expected value of D is 0.

$$D = tt_{ff,j} \cdot tt_{est,i} - tt_{ff,i} \cdot tt_{est,j} - 0 \tag{4.7}$$

$$Var\left[D\right] = \mathbb{E}\left[(D - \mathbb{E}[D])^2\right] \tag{4.8}$$

$$Var\left[D\right] = \mathbb{E}\left[D^2\right] \tag{4.9}$$

The variance of the observation represent the random variable of the difference of the two sides of the equation. The higher value tolerates more difference than the lower.

Table 4.4: Deviance of estimation for all segments for different variances of artificial rate keeping observations

variance of	RSD of estimation on segment							
observations	1	2	3	4	5	6	7	8
400	0.0773	0.3149	0.4061	0.2666	0.1692	0.6694	0.5191	0.2769
4 000	0.0157	0.1526	0.2461	0.1504	0.1208	0.3630	0.3583	0.1217
40 000	0.0098	0.0572	0.1094	0.0549	0.0813	0.1272	0.1626	0.1065
400 000	0.0097	0.0283	0.0338	0.0269	0.0689	0.0553	0.0810	0.1069
4 000 000	0.0097	0.0265	0.0226	0.0267	0.0670	0.0531	0.0769	0.1069

Table 4.4 shows the deviation of the estimation from the real travel time on the segments for four different value of variance. As can be seen the higher value influences less the real observations and the correctness of the estimation.

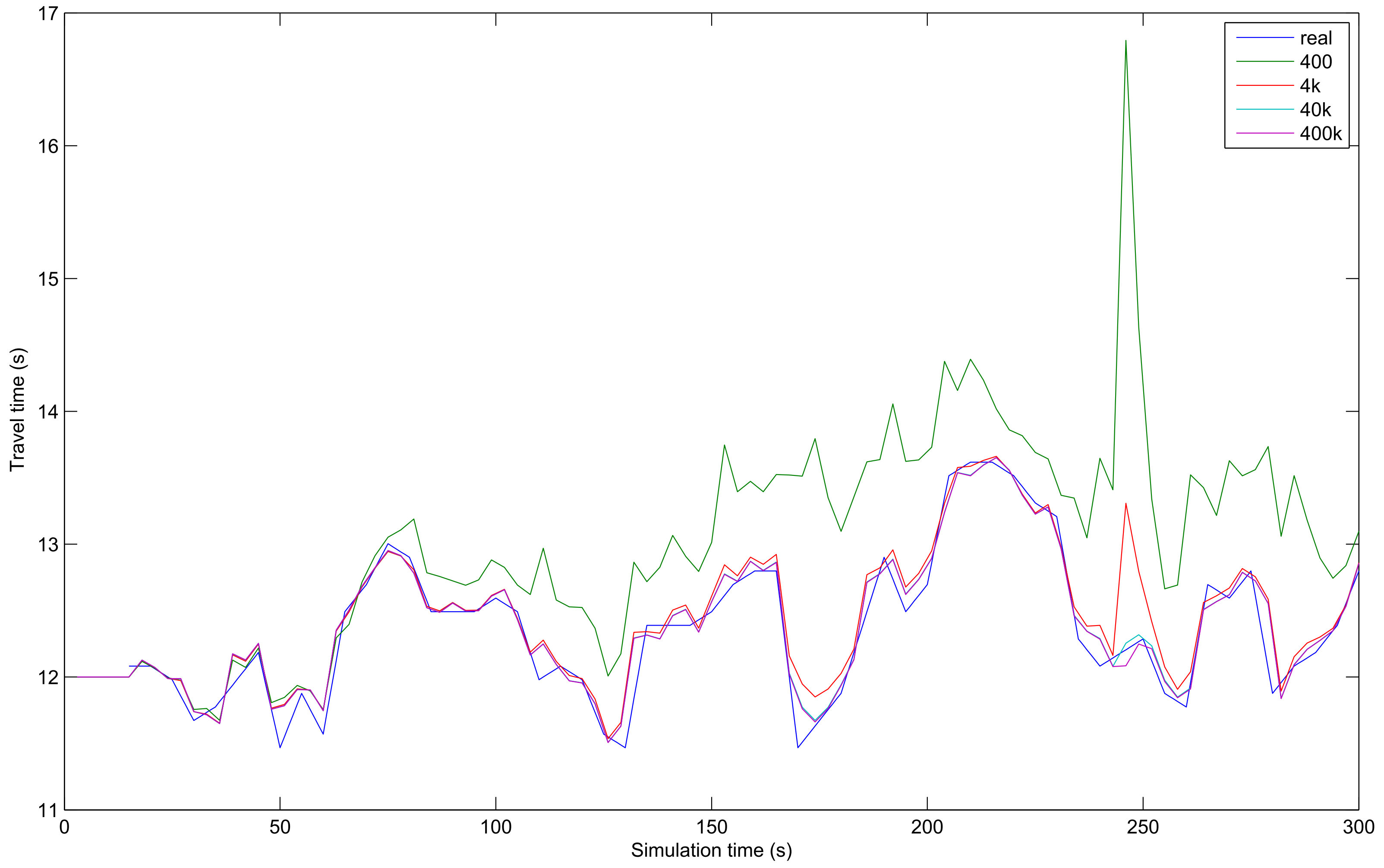

Figure 4.10: Effect of different update periods on the estimation of travel time on segment 1

For a heterogeneous segmentation of the road network with widely varying travel times of the segments there should be different values set as variances for each pair of segments. The simulated networks were quite homogeneous in this aspect so the variance of the observations were the same constant parameter for all following segment pair.

As seen in Table 4.1 the free flow travel times of segments are about 15sec. The typical value of a term in the Eq. 3.6 is about $15 \cdot 15 = 225$ omitting the dimension unit. The dimension of the variance of D is the square dimension of D, it is sec^4.

A good point to start to guess values for variance is the 4th power of the typical travel time. Figure 4.10 displays the estimations for four different order of magnitude set as variance.

As can be seen the plots for variance values 40000 and 400000 are overlapping as the higher magnitudes, too. The plot for variance of 4000 can be differentiated from the previously mentioned ones and the plot for value 400 has a somewhat huge constant error.

Free flow estimation for weak traffic

The assumption for the case of rare traffic discussed in Section 3.3.5 states that a transient function should be utilised to estimate free flow travel time for segments without observations overlapping them. This assumption creates artificial observations that are served to the Kalman filter among real observations.

These fake observations do not conflict with real ones because they are injected for segments without any observation overlapping. Therefore the only observations to cope with are from the assumption of free flow rate keeping.

It is an open question to decide which assumption should be the stronger, which should have less variance then the other.

There are two possible scenarios that these assumptions get in conflict with each other, both are reasonable.

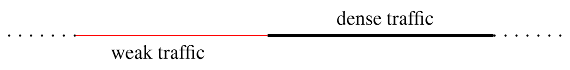

Figure 4.11: Scenario of following segments with differing observation amount

One possible scenario is when the two segments one without information about traffic and the other with a valid estimation based on real observations are following pair without a road junction. Figure 4.11 shows this situation. The scene shows that for the first segment there is weak traffic or it cannot be measured. For the first case a source or sink should be the explanation while the latter case suggests error in the infrastructure.

The solution of the former case is similar to the second scenario. A solution could be for the second case that homogeneous traffic is assumed and the rate keeping assumption is the stronger. Because the framework does not have the feature to test errors in the infrastructure and data sources the variance parameter was set up according to second scenario.

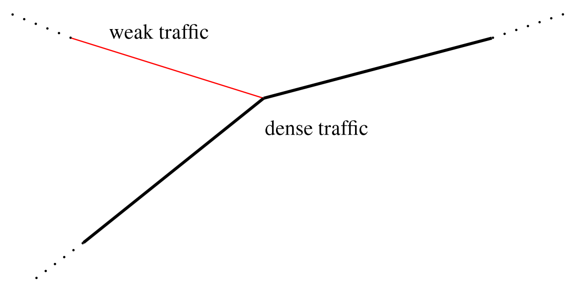

Figure 4.12: Scenario of junction with a dominant route

The second scenario as displayed in Fig. 4.12 states that there is a dominant route through the junction and an arm with unmeasurable traffic.

In this case the free flow travel time is more appropriate estimation for the branch without observations because a traveller can pass that segment in free flow state.

Table 4.5: Deviance of estimation for different values of variance of assumption

variance of	RSD of estimation on segment							
observations	1	2	3	4	5	6	7	8
2	0.0112	0.0339	0.0226	0.0819	0.1195	0.0919	0.0853	0.1077
20	0.0111	0.0325	0.0301	0.0463	0.0850	0.0723	0.0866	0.1044
200	0.0115	0.0487	0.0787	0.0442	0.0783	0.0981	0.1227	0.1037
2000	0.0128	0.0652	0.1187	0.0577	0.0823	0.1387	0.1638	0.1037
20 000	0.0149	0.0678	0.1262	0.0628	0.0839	0.1463	0.1706	0.1041
200 000	0.0154	0.0675	0.1261	0.0640	0.0843	0.1455	0.1700	0.1040

The artificial observation created for a segment contains only that segment and its value is the free flow travel time on the segment. Thus the physical unit of the variance is sec^2.

As can be seen in Table 4.5 and Fig. 4.13 the huge values of the variance are inconsistent with the free flow rate keeping assumption causing flickering in the estimation. Especially it causes the estimation flicker at frequent update cycles and scarce amount of observations.

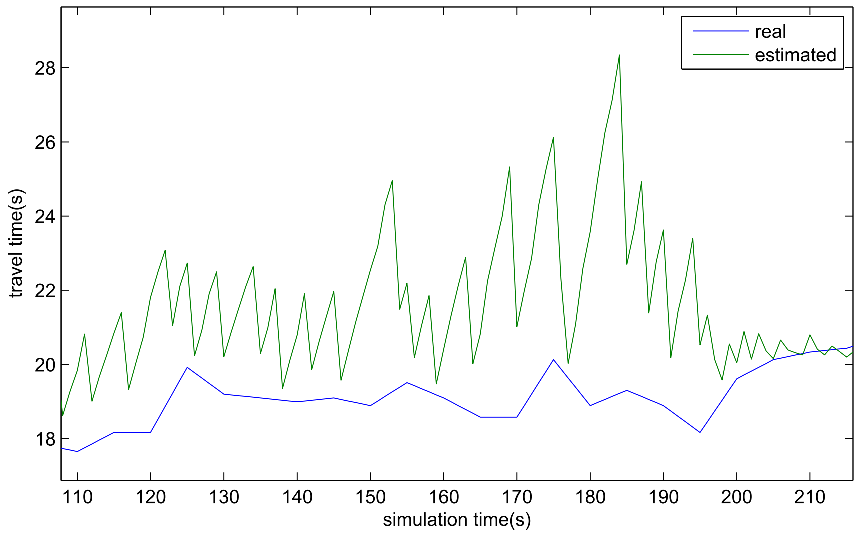

Figure 4.13: Estimation for high variance of scarce data assumption

4.4.3 Transient function to free flow estimation

The assumption for the case of rare traffic discussed in Section 3.3.5 states that a transient function should be utilised to estimate free flow travel time for segments without observations overlapping them.

The transient function of Eq. 3.9 has two parameters to set up. First the duration of the transit noted as $t_{transient}$ and second the exponent that determines the shape of the function.

The time interval and the shape of the function supports the purpose that depending on the current traffic conditions and the applied technology the short gap between valid observations could be filled. For the case of weak or no traffic on the road segment the free flow travel time should give a correct guess.

Duration of transit to free flow travel time

The correct duration of transit to free flow of the current road segment depends mainly on the road network. This is implemented as a simple constant parameter in the framework but the fundamental diagram of the traffic flow could also be utilised to a quick and accurate estimation method.

The only scenario that tested the duration values were about the short period of observation scarcity, just a short time that observations are missing for a few update cycles.

That is the reason why the deviation of estimation does not get worse as the higher values are probed. For values above $30sec$ the RSD does not really change.

Table 4.6 and Fig. 4.14 shows the estimations and their relative deviances.

Table 4.6: Deviance of estimation for different $t_{transient}$ duration times

duration	RSD of estimation on segment							
of transit	1	2	3	4	5	6	7	8
$3sec$	0.0098	0.0251	0.0258	0.0378	0.0828	0.0635	0.0765	0.1069
$15sec$	0.0097	0.0280	0.0325	0.0273	0.0701	0.0556	0.0807	0.1069
$30sec$	0.0097	0.0282	0.0336	0.0270	0.0691	0.0553	0.0810	0.1069
$45sec$	0.0097	0.0283	0.0337	0.0269	0.0689	0.0553	0.0810	0.1069
$150sec$	0.0097	0.0283	0.0338	0.0269	0.0688	0.0553	0.0811	0.1069

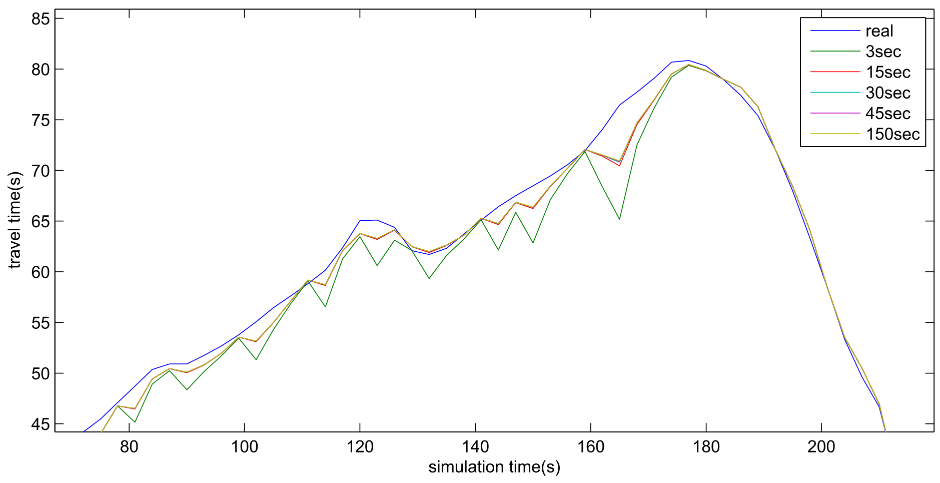

Figure 4.14: Real and estimated travel times on segment 4

Exponent of power function

The exponent influences the shape of the transient function which is basically a power function. If the exponent is 1 then the function is linear in the transient phase. The key role of the power function is that it should slowly transit the estimation to free flow in mind of the case of short period of data absence. Therefore the exponent values beneath one are unnecessary to try.

4.4.4 Results

With the tune of parameters according to the above tests Fig. 4.15 shows the plot of real and estimated values of the simulated road network of two altering routes described in Section 4.3.1.

The parameters are set as in Table 4.7. The results relative standard deviance is displayed in Table 4.8 for each road segment.

Table 4.7: Parameters for simulation

Parameter	Symbol	Value
Update interval	`glSimulationStep`	$3sec$
Variance of assumption of ratio keeping	`glVarianceOfRatioEstimation`	$400000sec^4$
Variance of assumption of free flow travel time	`glVarianceOfNoDataEstimation`	$200sec^2$
Duration of transit	`glTimeToFreeFlowIfNoData`	$30sec$
Exponent of transient power function	`glExponentOfSpeedToFreeFlowEstimation`	2

Table 4.8: Deviation of estimation on segments

Segment	RSD
1	0.0097
2	0.0282
3	0.0336
4	0.0270
5	0.0691
6	0.0553
7	0.0810
8	0.1069

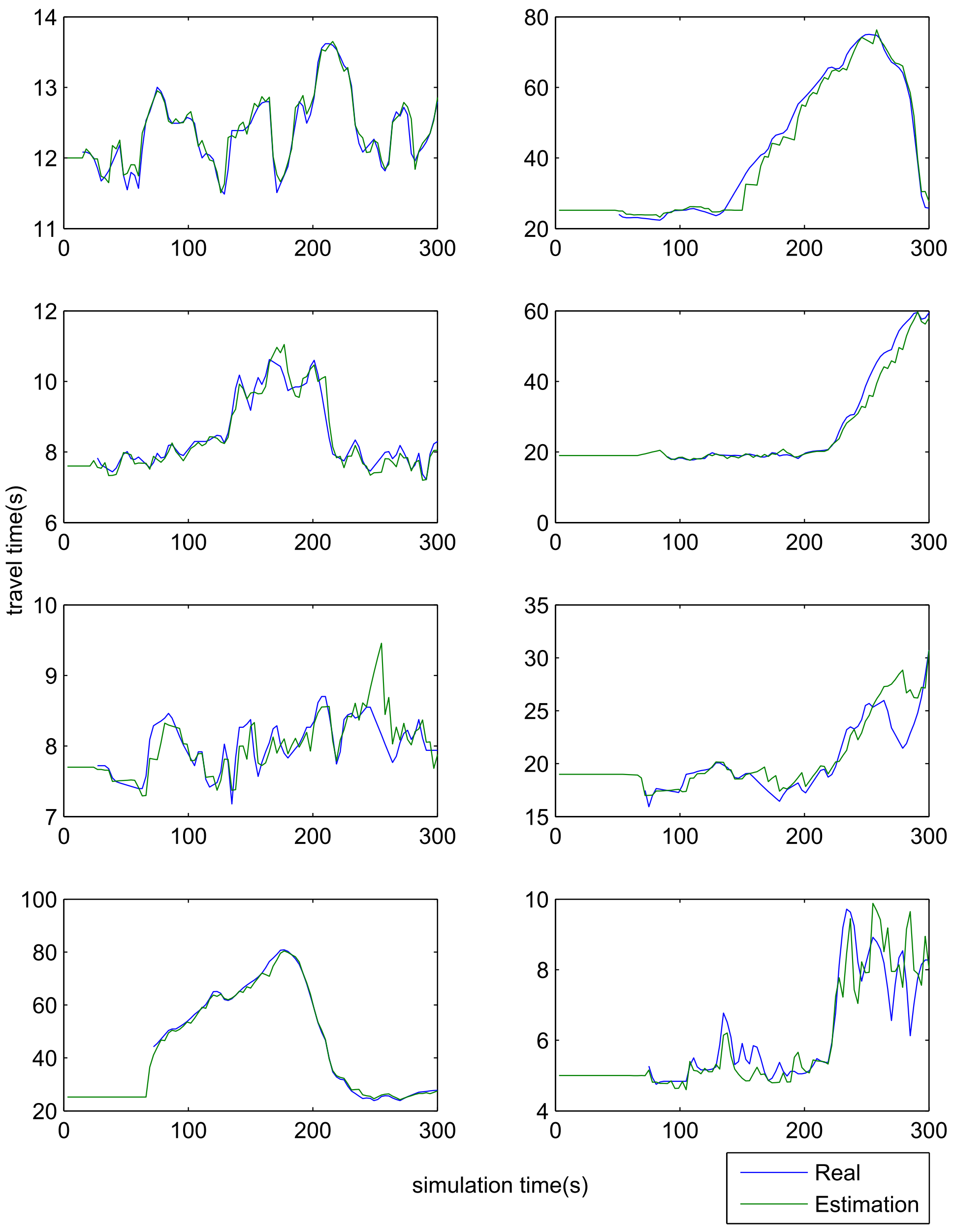

Figure 4.15: Simulation result

Chapter 5

Conclusion

The results of the simulation is encouraging but the validation and testing period would reveal lots of difficulties and possibilities of development.

5.1 Implementation

The implementation related difficulties could be separated as theory related and technical toughness.

5.1.1 Probability theory

Kalman filter method assumes Gaussian errors and white noises over the observations and states. The noises in the simulation were not examined to be true Gaussian and data sources could have diverse technology-depending random noise.

The divert kinds of random noise could point to other estimation processes than the linear quadratic underlying the Kalman filter. The behaviour of noises could be examined by field testing or analysing the mathematical models of the field like the theory of electromagnetic radiation used for analysing cellular networks.

Additionally, if a great amount of location relevant data is already recorded, data mining algorithms could be utilised to unfold hidden location relevant relations in the recorded data set.

5.1.2 Implementation difficulties

The current implementation is not a pleasant code in the sense of software development. It started from a proof of concept script and evolved in a quite flexible but not really reusable framework. It is capable of testing the feasibility of the idea and it accomplished.

A key feature of the framework is that it can run without the commercial Matlab environment and can be used for testing with the open source Octave. This feature required to omit complex data structures from the code thus the clean code and recycling were sacrificed.

5.1.3 Application in traffic control

The simulation result pointed out that the algorithm is feasible and could be used as a basis for complex urban traffic control. It can provide reliable, detailed and accurate information about traffic flow on a complex road network and can merge location aware observations from divert technologies as sources.

A real application of the idea needs more research and tune of the parameters and reimplementation of the code in a more robust and flexible environment.

5.2 Future works

5.2.1 Fundamental diagram based traffic assumption

The fundamental diagram of traffic flow defines the connection between the traffic density and traffic flow velocity. Depending on the applied technologies the amount and the complexion of the incoming observations could be used to estimate traffic density and thus improve the estimation of traffic flow parameters.

For example the automatic licence plate recognition technology registers all passing travellers and can measure traffic flux directly.

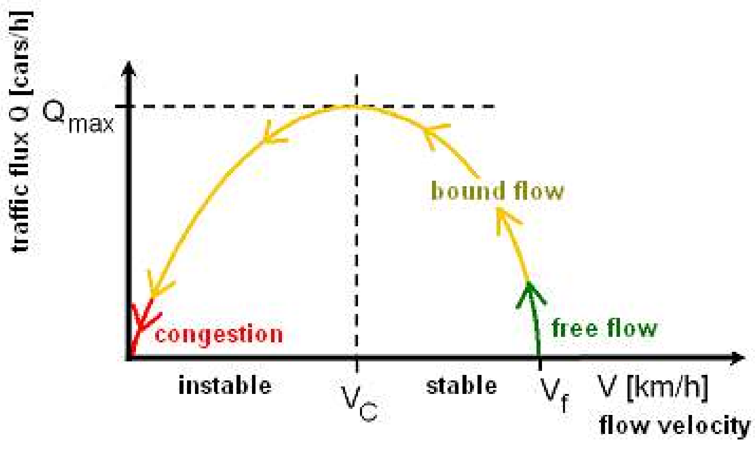

Figure 5.1: Speed-Flow diagram

The connections of other technologies and the traffic flux are less direct and need more researches both on the technology side and on the user customs on that technology, for example mobile telephony or bluetooth devices.

In the current implementation only a small section of the fundamental diagram was used to enhance the estimation, the case of unmeasurable weak traffic flow.

5.2.2 Continuous time data update

The method of Kalman filter assumed cyclic updates of the states because the underlying dynamic system has a discrete time cycle. An interesting research topic would be the application of a continuous time mathematical model. It could update the states for each observation when it arrives and correlate with the dynamic traffic flow more precisely. The short update cycles and the assumptions serve the same purpose from a discrete viewpoint.

5.2.3 Incident detection

Section 4.4 about tuning of parameters showed that it is possible to detect sudden alternation of traffic flow and incident could be detected on that basis.

A rapid increase in travel times can indicate a congestion if it is extended over the road network or a traffic accident if it affects just a few segments of the network.

If multiple data sources are used, they could be used to check each other and errors in the infrastructure could turn out. The estimation process could detect a flaw in one data source and notify the supervisor or it could adapt itself to the detected slip. This latter one should be analysed because a perfunctory implementation could cause inaccurate estimation and thus pile up faults.

5.2.4 VISSIM based integrated simulation environment

The increasing number of location based services and the demand of intelligent transportation systems point towards the claim of a testing framework in the development processes of these applications.1

A microscopic traffic modelling software like VISSIM could fulfil the need and provide a usable integrated test environment if additional software modules were developed to bridge the traffic simulation and the developed ITS application.

An integrated toolbox was created for VISSIM to form zones over the road network and during the simulation it could provide one or more data source streams of location information about travellers of the network.

Appendix A

Source code of framework

A.1 alter.m

```
function G = alter()

%         o---o
%        /     \
%  o---o         o---o
%        \     /
%         o---o

G=[

 0  12   0   0   0   0   0   0;
 0   0  7.6   7.7  0   0   0   0;
 0   0   0   0  25.2   0   0   0;
 0   0   0   0   0  25.2   0   0;
 0   0   0   0   0   0  19   0;
 0   0   0   0   0   0  19   0;
 0   0   0   0   0   0   0  5;
 0   0   0   0   0   0   0   0
];


end
```

A.2 defaultEstimation.m

```matlab
function [corMx, correction, corQ] = ...
   defaultEstimation(graph,default,quality)
corMx = eye(size(default,1));

correction = default;

corQ = ones(size(default))*quality;

end
```

A.3 defaultIfNoData.m

```matlab
function [corMx, correction, corQ] = ...
   defaultIfNoData(graph,default,mesMx,prevEstim)

global glVarianceOfNoDataEstimation;
global glExponentOfSpeedToFreeFlowEstimation;
global glNumberOfEdges;

global glTimeToFreeFlowIfNoData;
global glTimeOfLastData;

global glTime;
global glSimulationStep;

len = glNumberOfEdges;

if isempty(glTimeOfLastData)
    glTimeOfLastData = zeros(len,1);
end

d = glTimeToFreeFlowIfNoData;

exponent = glExponentOfSpeedToFreeFlowEstimation;
Quality = glVarianceOfNoDataEstimation;

```

```
    if isempty(mesMx)
      [corMx, correction, corQ] = ...
         defaultEstimation(graph,default,Quality);
   else
      corMx = [];
      num=0;
      correction=[];
      for k = [1:len;sum(mesMx,1)]
            if k(2)==0
            line = [zeros(1,k(1)-1) 1 zeros(1,len-k(1))];
            corMx=[corMx;line];

                t = glTime-glTimeOfLastData(k(1));
                pt = t - glSimulationStep;
                if t < d
                    trans = (1-(t/d)^exponent);
                    transPrev = (1-(pt/d)^exponent);
                    correction=[correction; ...
                         (prevEstim(k(1))- ...
                     default(k(1)))/transPrev*...
                        trans+default(k(1))];
                else
                    correction=[correction; default(k(1))];
                end
            num=num+1;
            else
                glTimeOfLastData(k(1)) = glTime;
            end
      end
      corQ = Quality*ones(num,1);
    end
end
```

A.4 figureLayout.m

```
function H=figureLayout(a,b)

H=zeros(a,b);

for i=1:a*b
    H(i)=i;
end

end
```

A.5 getEdges.m

```
function edges = getEdges(graph)
global glNumberOfVertices;
   len = glNumberOfVertices;

   edges = zeros(len);

   n = 0;
   for i=1:len
      for j=1:len
         e = j+(i-1)*len;
         if graph(e)~=0
            n=n+1;
            edges(j,i)=n;
         end
      end
   end

end
```

A.6 interpolateCSV.m

```
function state = interpolateCSV(file,delimeter,time,skip,map)
global glRealVissimCsvFile;
global glNumberOfEdges;

if isempty(glRealVissimCsvFile)
    glRealVissimCsvFile = ...
      sortrows(dlmread(file,delimeter,skip,0),3);
end

state = zeros(glNumberOfEdges,1);
csv = glRealVissimCsvFile;

times = csv(:,1);

corr = [1.0239 1.1064 1.0877 1.01 1.01 1.0323 1.0214 1.0748];
% because vissim links are shorter of the connectors

for i=1:glNumberOfEdges
    values = csv(:,2*i);
    cars = csv(:,2*i+1);
    state(map(i)) = corr(map(i))* ...
      interp1(times(cars>0),values(cars>0),time);
end
end
```

A.7 interpolateMatrix.m

```
function yi=interpolateMatrix(X,Y,xi)

yi=zeros(size(xi));

for i=1:size(X,2)
   yi(i)=interp1(X(:,i),Y(:,i),xi(i));
end
end
```

A.8 KalmanFilter.m

```
function estimation = ...
   KalmanFilter(graph,prevEstim,default,obsMx,...
      observation,qualities)

   I = eye(length(prevEstim));
   [Lk,m,P,e] = dlqe(I,I,obsMx,I,diag(qualities));
   estimation = prevEstim + Lk*(observation-obsMx*prevEstim);

end
```

A.9 KalmanWithAssumptions.m

```
function estimation = ...
   KalmanWithAssumptions(graph,prevEstim, ...
      default,mesMx,measurement,mesQ)

   [corMx, correction, corQ] = ...
      getCorrections(graph,default,mesMx,prevEstim);

        ObsMatrix = vertcat(mesMx, corMx);
        Observation = vertcat(measurement, correction);
   Qualities = vertcat(mesQ,corQ);

   estimation = ...
      KalmanFilter(graph,prevEstim,default, ...
         ObsMatrix,Observation,Qualities);

end
```

A.10 keepDefaultRatio.m

```
function [corMx, correction, corQ] = keepDefaultRatio(graph)

global glEdges;
global glNumberOfVertices;
global glNumberOfEdges;
global glVarianceOfRatioEstimation;

genLine=@(A,B,A_w,B_w) [zeros(1,A-1) B_w zeros(1,B-A-1)...
   -A_w zeros(1,glNumberOfEdges-B)];

corMx=[];

for vertex=1:glNumberOfVertices
   for from=1:glNumberOfVertices
      from_edge_w = graph(from,vertex);
      if(from_edge_w~=0)
         for to=1:glNumberOfVertices
            to_edge_w=graph(vertex,to);
            if(to_edge_w~=0)

               from_edge = glEdges(from,vertex);
               to_edge = glEdges(vertex,to);

               line = [];
               if from_edge < to_edge
                  line = genLine(...
                     from_edge,to_edge,from_edge_w,to_edge_w);
               elseif to_edge < from_edge
                  line = genLine(...
                     to_edge,from_edge,to_edge_w,from_edge_w);
               end

               corMx = [corMx; line];

            end
            end
        end
    end
end

correction = zeros(size(corMx,1),1);
corQ = ones(size(corMx,1),1)*glVarianceOfRatioEstimation;

end
```

A.11 long.m

```
function G = long()

% multi.inp
% o---o---o---o---o---o
%

G=[

 0 10.38 0    0    0    0 ;
 0    0 10.77 0    0    0 ;
 0    0    0 10.55 0    0 ;
 0    0    0    0 10.43 0 ;
 0    0    0    0    0 10.58;
 0    0    0    0    0    0

];


end
```

A.12 main.m

```
function [Time,AllTraveltimes,AllEstimations] = main(run)
%settings_random;
%settings_vissim_long;
settings_vissim_alter;

global glTime;
global glSimulationStep;
step = glSimulationStep;

FreeflowTraveltimes = states(Graph);
ActualTraveltimes = FreeflowTraveltimes;

AllTraveltimes = zeros(run/step,size(FreeflowTraveltimes,1));
AllEstimations = zeros(run/step,size(FreeflowTraveltimes,1));
EstimatedTraveltimes = FreeflowTraveltimes;

Time = step:step:run;
for time=Time
   glTime = time;

   ActualTraveltimes = nextState(ActualTraveltimes,time);
   [MeasurementMatrix, Measurements, Qualities] = ...
        getMeasurement(Graph, ActualTraveltimes,time);

    PreviousEstimation = EstimatedTraveltimes;
   EstimatedTraveltimes = getEstimation(Graph,...
      PreviousEstimation,FreeflowTraveltimes,...
      MeasurementMatrix,Measurements,Qualities);

    AllTraveltimes(time/step,:) = ActualTraveltimes';
    AllEstimations(time/step,:) = EstimatedTraveltimes';

end
end
```

A.13 negyzet.m

```
function G = negyzet()

% o---o---o
% |   |   |
% o---o---o
% |   |   |
% o---o---o


G=[

 0 20  0 20  0  0  0  0  0;
20  0 20  0 20  0  0  0  0;
 0 20  0  0  0 20  0  0  0;
20  0  0  0 20  0 20  0  0;
 0 20  0 20  0 20  0 20  0;
 0  0 20  0 20  0  0  0 20;
 0  0  0 20  0  0  0 20  0;
 0  0  0  0 20  0 20  0 20;
 0  0  0  0  0 20  0 20  0

];


end
```

A.14 noiseUni.m

```
function next = noiseUni(state,minVar,maxVar);

len = max(size(state));
next = state + rand(len,1)*(maxVar-minVar)+ones(len,1)*minVar;

end
```

A.15 plotInMatrix.m

```matlab
function plotInMatrix(time, real, estimation, how)

[Height, Width] = size(how);

for x=1:Width
   for y=1:Height

      subplot(Height,Width,(y-1)*Width+x);
      line = how(y,x);
      plot(time,[real(:,line) estimation(:,line)])

   end
end

legend('Real','Estimation')

end
```

A.16 plotRealEstimation.m

```matlab
function plotRealEstimation(time,real,estimation,line)

plot(time,[real(:,line) estimation(:,line)])
legend('Real','Estimation')

end
```

A.17 randomMeasurement.m

```
function [mesMx, mes, qua] = randomMeasurement(graph, state);

global glRandomMeasurementNumberOfMeasurementsInEachTurn;
global glRandomMeasurementDecreaseNum;
global glRandomMeasurementIncreaseNum;

numMes = glRandomMeasurementNumberOfMeasurementsInEachTurn;
minVar = glRandomMeasurementDecreaseNum;
maxVar = glRandomMeasurementIncreaseNum;


   len = max(size(state));
   mesMx = zeros(numMes,len);

   for i=1:numMes
      mesMx(i,:) = randomRoute(graph);
   end

   mes = noiseUni(mesMx*state, minVar, maxVar);

   qua = ones(numMes,1)*(maxVar-minVar)^2/12;

end
```

A.18 randomRoute.m

```
function obsVector = randomRoute(graph)
global glShortestPathNeighbours;
global glNumberOfVertices;
global glEdges;
global glNumberOfEdges;

   len = glNumberOfVertices;

   a = randi(len);
   b = randi(len);
   while a==b
      b=randi(len);
   end

   start = min(a,b);
   finish = max(a,b);

   obsVector = zeros(1,glNumberOfEdges);

   nextVert = finish;
   prevVert = glShortestPathNeighbours(start,finish);
   while prevVert~=0
      obsVector(glEdges(prevVert,nextVert))=1;
      nextVert = prevVert;
      prevVert = glShortestPathNeighbours(start,nextVert);
   end

end
```

A.19 readFromCsv.m

```matlab
function [mesMx, mes, qua] = ...
   readFromCsv(file, delimeter, time, step)
% header: links,time,traveltime

global glObservationCsvFile;
global glNumberOfEdges;

if isempty(glObservationCsvFile)
    glObservationCsvFile = ...
      sortrows(dlmread(file,delimeter,1,0),2);
end

csv = glObservationCsvFile;

rows = csv(csv(:,2)>(time-step) & csv(:,2)<=time,:);

variance = 1;

numOfMeasurements = size(rows,1);
if numOfMeasurements==0
    mesMx=[];
    mes=[];
    qua=[];
else
    mesMx=zeros(numOfMeasurements,glNumberOfEdges);
    mes=zeros(numOfMeasurements,1);
    qua=ones(numOfMeasurements,1)*variance;

    for i = 1:numOfMeasurements
        mesMx(i,:)= bitget(rows(i,1),2:(glNumberOfEdges+1));
        mes(i)=rows(i,3);
    end
end
end
```

A.20 settings_random.m

```

Graph=negyzet();

setupGlobals(Graph);

nextState = @(state,time) noiseUni(state, -1, 1);
getMeasurement = @(graph, state, time) ...
    randomMeasurement(graph, state);
getEstimation = @(graph,prevEstim,default,...
    obsMx,observation,qualities,time) ...
    KalmanWithAssumptions(graph,prevEstim,...
        default,obsMx,observation,qualities);
```

A.21 settings_vissim_alter.m

```
Graph=alter();

setupGlobals(Graph);

global glSimulationStep;

nextState = @(state,time) ...
    interpolateCSV('real.csv',';',time,20,[4 5 1 2 3 6 7 8]);
getMeasurement = @(graph, state, time) ...
    readFromCsv('vissim_alter_observation.csv',',', ...
        time,glSimulationStep);
getEstimation = @(graph,prevEstim,default, ...
    obsMx,observation,qualities,time) ...
    KalmanWithAssumptions(graph,prevEstim,default, ...
        obsMx,observation,qualities);
```

A.22 settings_vissim_long.m

```

Graph=long();

setupGlobals(Graph);

global glSimulationStep;

nextState = @(state,time) ...
    interpolateCSV('multi.rsz',';',time,17,[1 2 3 4 5]);
getMeasurement = @(graph, state, time) ...
    readFromCsv('vissim_long_observation.csv',...
        ',',time,glSimulationStep);
getEstimation = @(graph,prevEstim,default,...
    obsMx,observation,qualities,time) ...
    KalmanWithAssumptions(graph,prevEstim,default,...
        obsMx,observation,qualities);
```

A.23 setupGlobals.m

```
function setupGlobals(G)

global glGraph;
global glShortestPathNeighbours;
global glNumberOfVertices;
global glEdges;
global glDefaultEdgeWeights;
global glNumberOfEdges;
global glTime;
global glTimeOfLastData;

glGraph = G;

[dist, neighbour] = dijkstra(glGraph,[],[]);
glShortestPathNeighbours = neighbour;
%randi = @(n) round(rand()*(n-1))+1;
glNumberOfVertices = max(size(glGraph));
glEdges = getEdges(glGraph);
glDefaultEdgeWeights = states(glGraph);
glNumberOfEdges = max(size(glDefaultEdgeWeights));
glTime = 0;
glTimeOfLastData = zeros(glNumberOfEdges,1);

global glObservationCsvFile;
glObservationCsvFile=[];

global glRealVissimCsvFile;
glRealVissimCsvFile=[];

setupParameters();
end
```

A.24 setupParameters.m

```
function setupParameters()

global glVarianceOfRatioEstimation;
global glVarianceOfNoDataEstimation;
global glExponentOfSpeedToFreeFlowEstimation;
global glRandomMeasurementNumberOfMeasurementsInEachTurn;
global glRandomMeasurementDecreaseNum;
global glRandomMeasurementIncreaseNum;
global glSimulationStep;
global glTimeToFreeFlowIfNoData;

glVarianceOfRatioEstimation = 400000;
glVarianceOfNoDataEstimation = 200;

glExponentOfSpeedToFreeFlowEstimation = 2;

glRandomMeasurementNumberOfMeasurementsInEachTurn = 30;
glRandomMeasurementDecreaseNum = -1;
glRandomMeasurementIncreaseNum = 1;

glSimulationStep = 3;

glTimeToFreeFlowIfNoData = 30;
end
```

A.25 states.m

```
function S = states(G)
S=[];
   for i=1:prod(size(G))

      if G(i)~=0
         S=[S; G(i)];
      end
   end
end
```

List of Figures

List of Tables

Bibliography

[1] Albert-László Barabási. *Bursts.* Dutton Adult, 2010.

[2] BlipTrack. Homepage. `http://www.bliptrack.com/traffic/area-of-operations/`, May 2012.

[3] Rick Burgess. Google maps gets real-time traffic, crowd-sources android gps data. `http://www.techspot.com/news/48015-google-maps-gets-real-time-traffic-crowdsources-android-gps-data.html`, Apr 2012.

[4] N Caceres, J P Wideberg, and F G Benitez. Deriving origin – destination data from a mobile phone network. *Engineering and Technology*, 1(1):15–26, 2007.

[5] Francesco Calabrese. Real-time urban monitoring using cell phones: A case study in rome. *IEEE Transactions on Intelligent Transportation Systems*, 2011.

[6] Swarco Traffic Systems Gmbh. BLIDS flyer. `http://www.swarco.com/en/content/download/8058/101437/file/DRIVE-ON-1-2011_1.pdf`, Sep. 2011.

[7] Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008.

[8] I. Wells J. White. Extracting origin-destination information from mobile phone data. Technical report, 2002. (unpublished working paper).

[9] Michael G. Kay. dijkstra.m. `http://www.ise.ncsu.edu/kay/`, May 2012.

[10] GeoX Kft. EgérÚt Homepage. `http://www.egerut.com`, May 2012.

[11] Zoltán Koppányi. Emberi mozgásminták vizsgálata GSM alapú helymeghatározással. Master's thesis, BUTE, 2011.

[12] Axel Küpper. *Location-based Services.* John Wiley & Sons, 2005.

[13] Toshio Miki Linda Ackerman, James Kempf. Wireless location privacy: Law and policy in the U.S., EU and Japan. *Internet Society*, 2003.

[14] Trafficmaster Ltd. Real time traffic information. `http://www.trafficmaster.co.uk/content/1/60/real-time-traffic-information.html`, Mar. 2012.

[15] Jean luc Ygnace Chris Drane, Y. B. Yim, Renaud De Lacvivier, Jean luc Ygnace Inrets, Chris Drane Uts, and Renaud De Lacvivier Enac /sodit. Travel time estimation on the San Francisco Bay Area network using cellular phones as probes. Technical report, 2000.

[16] NL Openbaar Ministerie. Trajectcontrole. `http://www.flitsers.nl/trajectcontrole`, Mar. 2012.

[17] Einar Snekkenes. Concepts for personal location privacy policies. In *ACM Conference on Electronic Commerce, 2001.*

[18] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of Predictability in Human Mobility. *Science*, 327(5968):1018–1021, February 2010.

[19] Tamás Tettamanti, Hunor Demeter, and István Varga. Route choice estimation based on cellular signaling data. *Acta Polytechnica Hungarica*, in press.

[20] János Tóth PhD. Közúti informatika jegyzet. `http://www.kku.bme.hu/kepzes_bsc/segedletek/BMEKOKUA212/kozuti_informatika.pdf`, 2012.

[21] D. Valerio. Road traffic information from cellular network signaling. Technical report, Telecommunications Research Center Vienna, 2009.

[22] PTV Vision. *VISSIM 5.40-01 - User Manual.* PTV AG, 2011.

[23] Vysionics. Routehawk jtms. `http://www.vysionics.com/RouteHawk-JTMS/`, May 2012.

[24] Wikipedia. Bluetooth. `http://en.wikipedia.org/wiki/Bluetooth`, Feb 2012.

[25] Wikipedia. Global positioning system. `http://en.wikipedia.org/wiki/GPS`, May 2012.

[26] Wikipedia. Global system for mobile communications. `http://en.wikipedia.org/wiki/GSM`, Feb 2012.

[27] Wikipedia. Kalman filter. `http://en.wikipedia.org/wiki/Kalman_filter`, Jan 2012.

[28] Wikipedia. Radio-frequency identification. `http://en.wikipedia.org/wiki/RFID`, Feb 2012.

[29] Wikipedia. Satellite navigation. `http://en.wikipedia.org/wiki/Satellite_navigation`, Feb 2012.

[30] Qing yu Luo. Measuring the external costs of urban traffic congestion. Technical report, International Conference on Transportation Engineering, 2007.

[31] Yueming Yuan, Wei Guan, and Wei Qiu. Map matching of mobile probes based on handover location technology. In *ICNSC*, pages 587–592. IEEE, 2010.

[32] Mark Zuckerberg. Facebook. `http://www.facebook.com/`, May 2012.