Real-time Traffic Estimation on Highways Using Cellular Data

Master Project

Metin Balaban

August 17, 2017

Advisors: Prof. Patrick Thiran, Dr. Dan-Christian Tomozei

School of Computer and Communication Sciences, EPFL

Abstract

Recent advances on distributed and efficient processing of high-volume geolocation data has enabled real-time tracking and information extracting applications. Extracting road traffic information based on real-time, user-provided GPS location data has gathered a lot of attention in recent years. In this work, we investigate methods for extracting real-time road traffic information based on noisy cellular localization data. We investigate two different approaches to this problem and identify challenges related to them. Firstly, we introduce hidden Markov models to provide a solution that deals with any type of measurement noise that might occur during a drive in a highway. Secondly, we present a Kalman Filter based approach which does not require any model for driver behavior other than the basic laws of classic physics. Finally, after presenting the infrastructure of Swisscom Big Data Platform, we make a comparative analysis of various real-time algorithms tested on this platform and discuss the results.

Contents

Chapter 1

Introduction

Getting real-time road traffic information has become a very natural action for drivers thanks to the increasing availability of this information. Especially, after the proliferation of cell phones and real-time traffic services of big technology companies such as Google, Yandex, and Here this technology became part of every-day use. In last decades, we witnessed a big paradigm shift in monitoring traffic state. Traditionally, data collection is done through road sensors, loop detectors, cameras, and emergency calls from drivers. Clearly, this sensor technology requires a high public infrastructure and monitoring cost and does not provide a large coverage due to the economic limitations. Proliferation of GPS-equipped smart phones and 3G/4G communication technology has created an appealing alternative to traditional approach. Driven by the high precision of position estimation GPS provides, traffic state estimation methods developed by many researchers such as Work et al. (2008); Tao et al. (2012); Thiagarajan et al. (2009). In 2008, Mobile Century experiment was performed by Amin et al. (2008) which demonstrated that GPS-enabled cell phones can realistically be used as traffic sensors, while preserving individuals' privacy. Following the success of GPS based passive methods, the idea of using alternative passive (does not require installation of any hardware) data sources such as social network data and cellular network data for traffic estimations is investigated by several research groups and companies (Valerio et al., 2009c; Lai and Kuo, 2016; Tosi, 2017). Among the passive methods cellular networks has become more prominent due to the fact that next generation cell networks have penetrated to the market and their geographical coverage increased immensely in last 10 years. As accuracy of positioning algorithms gets better and bet-

ter (Lai and Kuo, 2016), cellular operators get the chance to establish their presence among the other traffic estimation service providers. Swisscom is currently the market leader in Switzerland in terms of mobile service subscribers (60% of all users) (Swisscom, 2016). Redzero project aims to create extra value out of existing telecommunication infrastructure by providing real-time traffic estimation and monitoring service to federal road office or other service providers.

The goal of this report is to investigate challenges in traffic state estimation in highways using cellular network data and present solutions suitable for the nature of the problem. The organization of this report is as follows. The following section presents the mathematical definition of the problem. After presenting the literature review on traffic state estimation problem in Chapter 2, we give the preliminary information about cellular network data. In Chapter 4, we propose a Hidden Markov Model based solution for the problem. After presenting our Kalman filter based solution Chapter 5, the real-time system architecture allowing real-time processing is introduced in detail. Finally, we evaluate our methods in Chapter 7 and discuss the results.

1.1 Problem Definition

A road is a line segment with an origin, an endpoint, and a direction. A road is also represented with consequent non-overlapping continuous chunks, which span all the road and are called *sections*. Therefore, we represent a road R with $R = \{s_1, s_2, \cdots, s_{n_R}\}$. A section s_i has length l_i, and origin p_i and a destination p_{i+1}. Highway networks consist of a set of roads. We assume that traffic state in each road is statistically independent, which lets us compute traffic states of these roads separately and in a broadly-distributed manner. In the rest of the report, we will assume that there is only one road, R. without loss of generality.

Likewise, we assume that physical state of each vehicle (i.e. position, speed, acceleration) on road R is statistically independent, which lets us compute internal states of these vehicles separately and in a broadly-distributed manner. Without loss of generality, we will assume until Chapter 6 that there is only one user (with user id *user_id*) on the road R.

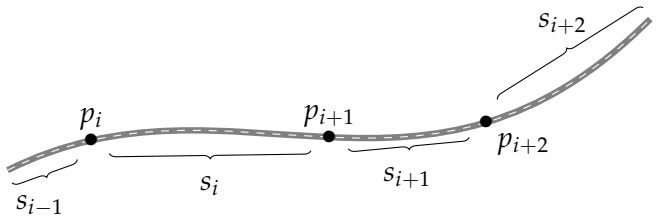

Figure 1.1: Sections indicated on road R

Cellular localization data consists of sequence of observations which are obtained from the cell cites the agent's mobile device asynchronously connected. In that case the number of possible observations is limited by the number of cell sites (this is a simplification of real life scenario because we are ignoring the "type" of the connection for a moment). We call the finite set of cells $C = \{c_1, c_2, \cdots, c_k\}$. An observation event O_t is a timestamped tuple $(t, user\_id, event)$, where event belongs to C.

Traffic state mapping problem Traffic state at time t in road R is usually represented with a mapping $M = \{s_1 \to v_1, s_2 \to v_2, \cdots, s_{n_R} \to v_{n_R}\}$ where v_i is l_i divided by the expected travel time between p_i and p_{i+1}. Given a stream of tuples $(t', user\_id, event)$ where $t' \leq t$, computing M is is called traffic state mapping problem.

We compute those v_is by estimating unknown physical states (position and velocity, for example) of individual vehicles. The current state estimation problem under noisy observations is known as filtering problem (Julier et al., 1995):

Filtering problem Without loss of generality, we consider a single vehicle moving along the road R. Let $X(t_i)$ indicate the vehicle's state at time t_i with respect to the origin of R. $Y(t_i)$ is the measurement of the vehicle's state at time t_i. Given a sequence of measurements $Y(t_1), Y(t_2), \cdots, Y(t_n)$, computing estimation of vehicle's last state $\hat{X}(t_n)$ is called filtering problem.

Chapter 2

Literature Review

Vehicular traffic estimation methods are categorized by the sort of data they require. In the literature, we see that these methods are based on either *Eulerian* or *Lagrangian* measurements. Eulerian sensing describes the sensor setting where sensor locations are fixed. Eulerian sensors such as loop detectors, speed cameras, virtual tripwires, and speed radars has been intensively used on many freeways across the world for several decades (Claudel and Bayen, 2008; Herrera and Bayen, 2010). Following the advances in the mobile devices technology, Lagrangian measurements, in which the sensors are mobile, have started to be used extensively. GPS sensors in the mobile phones and the cellular network infrastructure provided companies with cost-effective alternative to placing sensors all over the highways (Valerio et al., 2009a).

In the context of traffic state estimation, —similar to the measurement type— models for processing measurement data fall into two main categories: traffic flow physics and statistical methods. The latter class of models focus on real-time speed and travel time estimation while the former, known as second order models, yield more specific estimations such as density of the traffic flow. Flow based models gathered a lot of attention in last decades. Lighthill and Whitham (1955) propose a partial differential equation (LWE) that describes the traffic flow in a road. Han et al. (2012) use LWE model for highway traffic estimation in Lagrangian-based framework. Herrera and Bayen (2010) extend LWE model and fuse Eulerian measurements with Lagrangian measurements using Kalman filter. Work et al. (2008) introduce a new partial differential equation based on LWE model, as a flow model for

Lagrangian measurements (GPS). Authors translate this PDE into a Velocity Cell Transmission Model (CTM-v), which is a discrete-time discrete-space nonlinear dynamical system. Similarly, Work et al. (2009) introduce an ensemble Kalman filtering algorithm to reconstruct the state of the traffic using GPS enabled phones. The mixture Kalman filter (Sun et al., 2003), particle filter (Mihaylova et al., 2007; Mihaylova and Boel, 2004), unscented Kalman filter (Mihaylova et al., 2006), extended Kalman filter (Wang and Papageorgiou, 2005), and distributed local Kalman consensus filter (Sun and Work, 2014) based solutions are proposed for the Eulerian measurements setting in highways. Porikli and Li (2004) estimate traffic congestion by processing speed camera video streams using Markov models (HMM).

On the other hand, statistical methods —known as first order methods— are gaining attention in last two decade increasingly, with the proliferation of smart phones as traffic sensors (Herrera and Bayen, 2010). Thiagarajan et al. (2009) propose VTrack system for travel delay estimation which uses cell phones as WiFi and GPS localization probes. Although the main focus of the paper is travel time estimation and energy efficiency of the sensors used, authors present a neat HMM based map matching algorithm to estimate the route driven by agents. Tao et al. (2012) propose a real-time traffic state estimation framework using A-GPS (Assisted GPS) mobile phones. The framework collects positioning and tracking information from users using A-GPS, produces individual position and speed estimates, aggregates the results and displays them on users' cell phones. Authors use a simulation to refine their model as it provides "ground truth" locations and speeds of probes. Yoon et al. (2007) advocate Bayesian framework for traffic state estimation provided that GPS location data is available. Instead of tracking and detecting speeds of individual probes, authors show that after analyzing historical data, irregular driving patterns and anomalies in the usual traffic flow can be detected using algorithm such as maximum likelihood and maximum a posteriori estimation.

Bolla and Davoli (2000) is the first paper known to us which attempt to use cellular network infrastructure for road traffic estimation. Authors use simulations to show their methodology is capable of estimating vehicle density, vehicle velocity, and vehicle flow within 5% error in average across the time and space. (Valerio et al., 2009b) investigate the idea of using cellular network in depth and present a cellular-based road monitoring framework

which influence the framework we use in this project. In Chapter 6, we lay emphasis on the roadmap authors point out.

Cheng et al. (2006), and Schneider (2005) suggest using handover mechanism for computing average traffic speed. Cheng et al. (2006) implement this idea by presenting two Bayesian framework based road traffic estimation algorithm. The first algorithm is based on first-order traffic models. Since the first order traffic model is linear, authors propose using either Kalman or Particle filter to estimate state variables. Authors designed a particle filter for the second-order traffic model.

Discussion

In this section, we present how traffic state estimation problem is approached by previous works. We show that the classic approach for traffic state estimation has been using traffic flow physics and Eulerian measurements until the proliferation the smart phones as a localization probe. Smart phones allow a cheap alternative to expensive loop detectors as they can collect Lagrangian measurements such as GPS, WiFi, and cellular localization. Even though the potential of using cellular network for traffic estimation is put forward by several researchers, there is still lack of investigation of comparative analysis, and application in real scenarios. Therefore, we conclude that an in-depth analysis on how to build a real-time system in a real-life situation still makes up an interesting case for traffic state estimation research.

Chapter 3

Preliminaries

3.1 Characterization of Localization Data

In the context of cellular positioning, translating observation events to coordinates of locations in the road can be achieved through different strategies. One strategy is the following: each observation is mapped to a close shape in the map. That area represents the coverage of the observed cell tower. Then, each point in the road becomes associated with a single cell or some cells, (or no cell) if the point lies within the coverage radius of a cell. $\mathbb{P}(s|c)$ can be set based on this coverage analysis.

The main problem with this approach is that radius and coverage information cannot be obtained reliably due to several reasons such as geography of the area in question, or fluctuations of the signal power. Therefore, obtaining the distribution $\mathbb{P}(s|c)$ is not possible by relying on only the geography and specifications of cell cites.

The second approach is to empirically sample the distribution $\mathbb{P}(s|c)$ by making measurements along the road and saving the observation data for its further use to estimate $\mathbb{P}(s|c)$.

Drive Traces A drive trace T is a sequence of road position and observation pair denoted as $(X(t_1), C_1), (X(t_2), C_2), \cdots, (X(t_k), C_k)$ where C_is are random variables with sample space C. Practically, the road position $X(t)$ is obtained via a localization medium which is more sensitive than cellular localization, typically GPS, and is matched to the road using a map-matching algorithm.

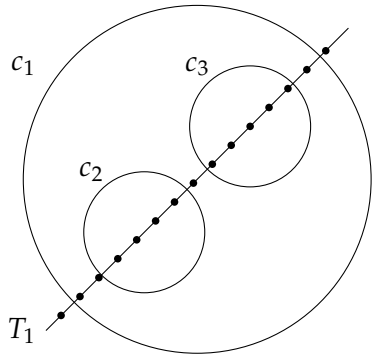

Figure 3.1: An example case showing the coverage of three cell sites c_1,c_2, and c_3, a road, and a drive trace T_1 along the road. Each black point along the road represents a position-observation pair. In each pair, the innermost cell cite that encloses the observation point is observed.

Given a set of drive traces $\mathbb{T} = \{T_1, T_2, \cdots, T_N\}$, estimation of discrete distribution $\mathbb{P}(s|c)$, represented by $\tilde{\mathbb{P}}(s|c)$, is calculated through the standard histogram to distribution transformation:

$$\tilde{Pr}(s_j|c_i) = \frac{|\{k \in \{1,2,...,M\} : (X(t_k), c_i) \in \mathbb{T}_u\}|}{Z(c_i)} \tag{3.1}$$

where $X(t_k) \in s_j$, $\mathbb{T}_u$ is union of all drive traces in $\mathbb{T}$, M is number of observation pairs in $\mathbb{T}_u$, and $Z(c_i)$ is a normalization factor.

Depending on the placement of cell-cites in the area, $\tilde{\mathbb{P}}(s|c)$ can display multi-modal character the due to overlaps or due to the geometry (see Figure 3.1). Therefore, it is hard to make assumptions (like Gaussian) on the characteristics of the observation noise in real life situations.

3.2 Handovers

Handover is a mechanism that transfers the ongoing connection of a cell phone from one cellular tower to another. The trigger for this event is the changes in the signal level. Based on these changes (usually occurs when the cell phone is moving), the network centrally transfers connection of the phone to another station. Handover takes place the in areas being covered by two or more base stations. Compared to the whole area of coverage for a cellular tower, the handover areas are smaller and are found in the borders of the coverage (See Figure 3.3). Therefore, handover events indicate higher certainty with regard to the location of the vehicle compared to the other

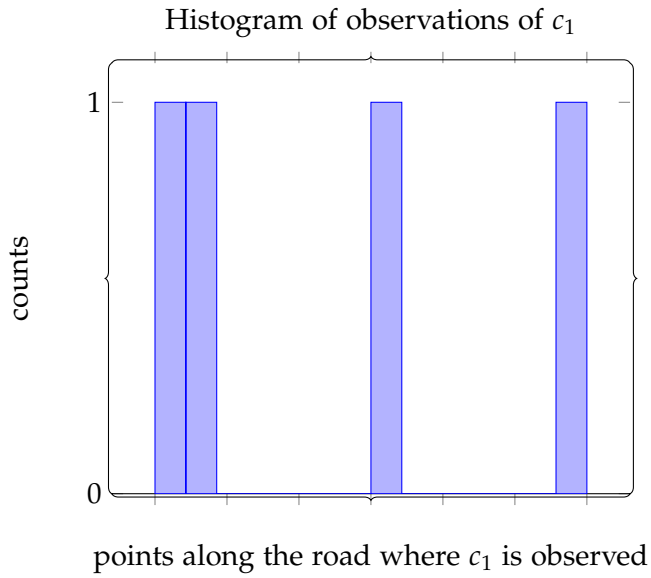

Figure 3.2: Histogram of observations of c_1 from the example above. We can assume that each bin in this histogram is corresponding into a segment. The example shows that the distribution can be multi-modal.

events. Handover events are similar to the other type of events received by base station:

$$E = (user\_id, cell\_id, timestamp, handover\_indicator)$$

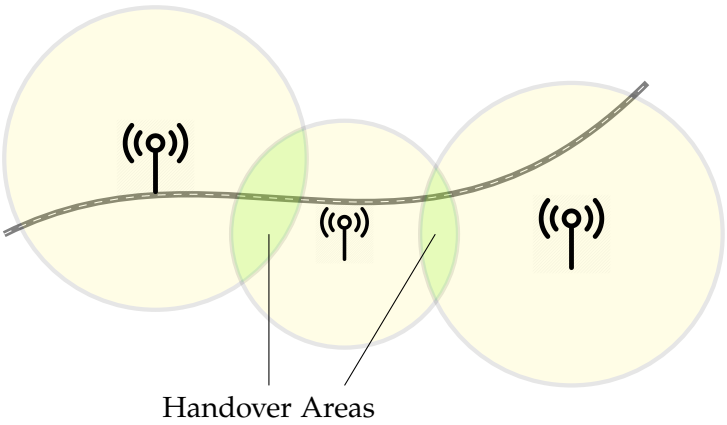

Figure 3.3: Some section of a highway passing through handover areas of nearby cells.

We exploit this useful property of handover events. In our algorithms, cell towers are associated with the portions of the highways they cover so that, when an event from a cell phone is received we can roughly localize the

device. We perform this mapping twice for each cell tower: one for handover events and one for non-handover events. Naturally, handover events will indicate a higher degree of certainty.

Chapter 4

The First Approach: Hidden Markov Models

As discussed in Section 1.1, the speed estimates per road section can be computed from individual speed or position estimates of the agents on the road.

4.1 Hidden Markov models

Hidden Markov models (HMMs) are special type of Bayesian networks, where underlying Markovian process are hidden from the observer and the hidden states are observed through observations which are conditionally independent from rest of the process, given the hidden state. $S_1, S_2, \cdots, S_n$ are states of the underlying Markov chain and are hidden from the observer. $O_1, O_2, \cdots, O_n$ are observed variables where $Pr(O_n|S_{1:n}O_{1:n-1}) = Pr(O_n|S_n)$. Hidden Markov model assumes that observations are sampled at a constant rate, which makes t a discrete variable. The period of these observation sequence is denoted by Δt.

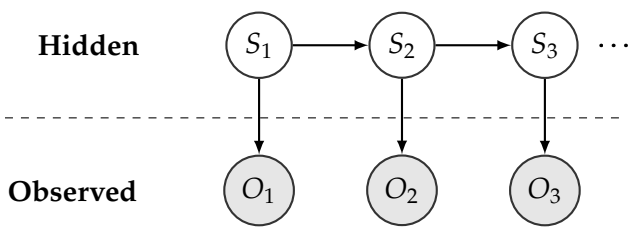

Figure 4.1: The Bayesian network which demonstrates conditional independence relations for hidden Markov model.

Hidden Markov models are useful to compute the most likely sequence of $S_1, S_2, \cdots, S_n$ when we have no direct access to those variables. We have access to the state transition probabilities of underlying Markov process and the distribution $\mathbb{P}(O|S)$. Viterbi and forward-backward algorithms are two algorithms that compute the most likely sequence of hidden variables.

In order to have the correct setup for a HMM based solution for filtering problem, we discuss three aspects of HMMs.

4.1.1 State definition and Markov chain transition probabilities

The very first assumption for HMM is that underlying stochastic process must satisfy Markov property. It is intuitive to include the road segment that a person is situated to the state definition of the Markov process. However this state definition is not sufficient alone for satisfying Markov property: the probability distribution of next state depends on the agent's current velocity. Therefore, in the minimal scenario, we need to include agent's velocity in the state definition. One can consider including acceleration in state definition too; however, this approach has two drawbacks. (1) This implies a bigger state space, raising up efficiency concerns. (2) Also, since the observations are very noisy, making correct predictions on the second derivative of the observed variable is very hard.

We assume that road is divided into continuous equal length segments (they differ from sections due to the fact that segments have equal length), given in the sorted order $s_1, s_2, \cdot, s_m$. So the states of the Markov chain are $S_t = (s_t, v_t)$ where s_t is the road segment and v_t is velocity. v_t is a discrete variable too: the continuous velocity spectrum is discretized into several speed profiles in order to reduce the state space. So, each v_t corresponds to a range of speed values (i.e. 0-60 km/h), which we call *speed profile*.

Being inspired from the notion of inertia in classical mechanics, the transition probabilities are assigned according to the following principles:

Conservation of speed: agents have tendency to preserve their speed profiles.

Obeying the speed limits: agents follow the speed limits of the highway.

Although we propose using a transition assignment scheme following above criteria, finding more mathematically robust and sound scheme for specifying the transition probabilities is a future work. In Appendix A, an example Markov chain following the principles above is given.

4.1.2 Observation probabilities

In real life scenario, the observation of a vehicle in the road at time t only depends on its position. That is, the observation does not depend on the velocity of the vehicle (statistically independent). Therefore $\tilde{Pr}(c_t|s_t) = \tilde{Pr}(c_t|s_t, v_t)$. However, $\tilde{\mathbb{P}}(c|s)$ is not known directly. We derive this distribution from $\tilde{\mathbb{P}}(s|c)$.

$$\begin{aligned}\tilde{Pr}(c_j|s_i) &= \frac{\tilde{Pr}(s_i|c_j)\cdot\tilde{Pr}(c_j)}{\tilde{Pr}(s_i)}, \text{ from Bayes rule} \\ &= \frac{\tilde{Pr}(s_i|c_j)\cdot\tilde{Pr}(c_j)}{\sum_k \tilde{Pr}(s_i, c_k)} \\ &= \frac{\tilde{Pr}(s_i|c_j)\cdot\tilde{Pr}(c_j)}{\sum_k \tilde{Pr}(s_i|c_k)\tilde{Pr}(c_k)}\end{aligned}$$

$\tilde{Pr}(c_j)$'s are obtained through their occurrence frequency in drivetraces.

4.1.3 Event quantization and synchronization

Hidden Markov model is a useful tool for filtering problems when observations are fed with a constant frequency. In our real time speed estimation problem, the observations are streaming in asynchronous manner and receiving observations from users at a constant rate is never the case. So the biggest challenge in modeling our position and speed estimation problem as hidden Markov model is effectively process asynchronous observation updates. Discretization of observation timestamps introduces some error in the computation. This error decreases with increasing sampling frequency. However, this frequency cannot be increased arbitrarily, since we are unable to capture small changes in position due to the large noise margin in our observations.

Another challenge is to handle erasures of messages (observations) resulting from the missing observations at time t. In other words, we need a mecha-

nism to handle the case where there is no observation made in an update period in the stream. We propose a solution in Section 4.2 by altering the underlying Bayesian network and giving an algorithm which is a variation of Forward algorithm.

4.2 Forward Algorithm, Extension, and Adaptation to Real Time Processing

4.2.1 Forward Algorithm

Forward algorithm is a dynamic programming algorithm which computes the most likely hidden state at time t that results in the sequence of observations.

The goal of the forward algorithm is to compute the joint probability $Pr(S_n, O_1, \cdots, O_n)$. We use dynamic programming to store intermediate states during the recursive calculation of this joint probability. The recursive formula is found as follows:

$$\alpha_n(S_n) := Pr(S_n, O_1, \cdots, O_n) \qquad \text{(definition)}$$

$$(4.1)$$

$$\begin{aligned}
\alpha_n(S_n) &= \sum_{S_{n-1}} Pr(S_{n-1}, S_n, O_1, \cdots, O_n) && \text{(marginalization)} \\
&= \sum_{S_{n-1}} Pr(S_n, O_n | S_{n-1}, O_1, \cdots, O_{n-1}) \\
&\cdot Pr(S_{n-1}, O_1, \cdots, O_{n-1}) && \text{(conditional prob.)} \\
&= \sum_{S_{n-1}} Pr(S_n, O_n | S_{n-1}, O_1, \cdots, O_{n-1}) \\
&\cdot \alpha_{n-1}(S_{n-1}) && \text{(Eq. 4.1)} \\
&= \sum_{S_{n-1}} Pr(S_n | S_{n-1}, O_1, \cdots, O_{n-1}) \\
&\cdot Pr(O_n | S_n, S_{n-1}, O_1, \cdots, O_{n-1}) \\
&\cdot \alpha_{n-1}(S_{n-1}) && \text{(conditional prob.)} \\
&= Pr(O_n | S_n) \cdot \sum_{S_{n-1}} Pr(S_n | S_{n-1}) \alpha_{n-1}(S_{n-1}) && \text{(cond. independence)}
\end{aligned}$$

$$(4.2)$$

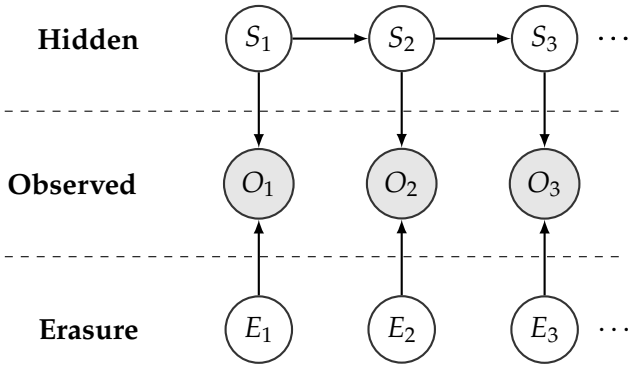

Figure 4.2: The Bayesian network of extended hidden Markov model in presence of erasures.

After computing the joint probability $\alpha_n(S_n)$ for every state S_n can possibly be, the most likely state turns out the one that maximizes $\alpha_n(S_n)$.

4.2.2 Handling Erasures by Extending the Model and Forward Algorithm

To establish the mathematical ground for erasures, the erasure observation event is introduced, represented by ε. Consequently, the observation event set $\mathbb{C} = c_1, c_2, \cdots$ extends to $\mathbb{C}_e = \mathbb{C} \cup \{\varepsilon\}$. We extend the Bayesian network of HMM with binary random variables $E_1, E_2, ..$ where E_t represents whether erasure takes place at time step t (See Fig. 4.2). The probability distributions are given as follows: for every i, $\mathbb{P}(O_i|S_i, E_i = 0) = \mathbb{P}(O_i|S_i)$, $Pr(O_i = \varepsilon|S_i, E_i = 1) = 1$, and $Pr(O_i \neq \varepsilon|S_i, E_i = 1) = 0$.

Thanks to the introduction of erasures, the time periods without observation event is filled with ε. One can derive the recurrence relation for forward algorithm with this setup as

$$\alpha_n(S_n) = Pr(O_n|S_n, E_n) \cdot \sum_{S_{n-1}} Pr(S_n|S_{n-1})\alpha_{n-1}(S_{n-1})$$

which boils down to:

$$\alpha_n(S_n) = \begin{cases} Pr(O_n|S_n) \cdot Q & O_n \neq \varepsilon \\ Q & O_n = \varepsilon \end{cases}$$

where $Q = \sum_{S_{n-1}} Pr(S_n|S_{n-1})\alpha_{n-1}(S_{n-1})$.

4.2.3 Practical Aspects

Forward algorithm given above has time complexity of $O(nm^2)$ where n is the length of the sequence of observations and m is the number of states in Markov chain. Provided the assumptions that

(1) in filtering problem, the agents are traveling in reasonable speeds and
(2) the road is long enough such that $n \gg d$ where d is the maximum degree of the graph representation of underlying Markov chain

hold, time complexity of the implementation can be reduced to $O(nmd)$ by, instead of a sparse matrix, using a dense matrix to keep transition probabilities.

Chapter 5

The Second Approach: Kalman Filter

Kalman filter is one of the most acclaimed and widespread algorithms used for stochastic estimation from noisy state measurements. Provided that certain conditions are met, such as characteristic of the noise, the algorithm computes estimates with minimum mean squared error. In Appendix B, we give a simple derivation for Kalman filter that proves the minimum mean squared error condition is satisfied. Kalman filter has some commonalities and differences with Hidden Markov Model. These two models are primary methods for solving filtering problem. On the other hand, Kalman filter presumes that the state space of the hidden variables is continuous and all hidden and observed variables follows a (multivariate) Gaussian distribution; whereas in Hidden Markov model, the state space of the hidden variable are discrete and there is no assumption on the characteristics of the distribution of hidden and observation variables.

5.1 The Algorithm

Kalman filter operates in two consequent stages that are called "Predict" and "Update". The variables in the following equations are given below.

- $\hat{\mathbf{x}}_k \in \mathbb{R}^n$: The estimation of the state vector containing the terms of interest for the system at time-step k given all observations until time-step k including the one at k. In other terms, it is so-called *a posteriori* estimate.

- $\hat{\mathbf{x}}_k^- \in \mathbb{R}^n$: The estimation of the state vector containing the terms of interest for the system at time-step k given all observations until time-

step $k-1$ including the one at $k-1$. In other terms, it is so-called *a priori* estimate.

- $\mathbf{A} \in \mathbb{R}^{nxn}$: State transition matrix which relates consequent state vectors.
- $\mathbf{u}_k \in \mathbb{R}^l$: The (optional) control input for the system. It contains any controllable terms in the system which alter the terms in the state vector.
- $\mathbf{B} \in \mathbb{R}^{nxl}$: It relates the control input to the state $\mathbf{x}$.
- $\mathbf{P}_k^- \in \mathbb{R}^{nxn}$: It is a priori estimate error covariance. In fact, $\mathbf{P}_k^- = \mathbb{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T]$.
- $\mathbf{P}_k \in \mathbb{R}^{nxn}$: It is a posteriori estimate error covariance. In fact, $\mathbf{P}_k = \mathbb{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T]$.
- $\mathbf{Q} \in \mathbb{R}^{nxn}$: It is (optional) process noise covariance. The error in the process is distributed with $N(0, \mathbf{Q})$.
- $\mathbf{z}_k \in \mathbb{R}^m$: The measurement vector is the collection of the measurements of the system.
- $\mathbf{H} \in \mathbb{R}^{nxm}$: It relates the state vector to the measurement vector.
- $\mathbf{R} \in \mathbb{R}^{mxm}$: It is the measurement noise covariance. The error in the measurement is distributed with $N(0, \mathbf{R})$.

Table 5.1: Kalman Filter "Predict" stage

$$\hat{\mathbf{x}}_k^- = \mathbf{A} \cdot \hat{\mathbf{x}}_{k-1} + \mathbf{B} \cdot \mathbf{u}_k \tag{5.1}$$

$$\mathbf{P}_k^- = \mathbf{A} \cdot \mathbf{P}_{k-1} \cdot \mathbf{A}^T + \mathbf{Q} \tag{5.2}$$

5.2 Applying Kalman Filter to Cellular Localization Problem

?? In Section 3.1, we have shown that cellular localization data can have a multi-modal distribution. However, Gaussian noise is still a good approx-

Table 5.2: Kalman Filter "Update" stage

$$\mathbf{K} = \mathbf{P}_k^- \cdot \mathbf{H}^T \cdot (\mathbf{H} \cdot \mathbf{P}_k^- \cdot \mathbf{H}^T + \mathbf{R})^{-1} \tag{5.3}$$
$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K} \cdot (\mathbf{z}_k - \mathbf{H} \cdot \hat{\mathbf{x}}_k^-) \tag{5.4}$$
$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K} \cdot \mathbf{H}) \cdot \mathbf{P}_k^- \tag{5.5}$$

imation to the measurement noise in most of the real life scenarios. From this viewpoint, we can define filtering problem in terms of Kalman algorithm. From simple equations of motion,

$$x_k = x_{k-1} + \dot{x}_{k-1} \cdot \Delta t$$
$$\dot{x}_k = \dot{x}_{k-1}$$

we can derive the following relation:

$$\mathbf{x}_k = \mathbf{A} \cdot \mathbf{x}_{k-1}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

and Δt is the time difference between step $k-1$ and k. Note that the time difference between time-steps can be arbitrary. So $\mathbf{A}$ changes in every update.

The control input $\mathbf{u}_k$ can be added to the model in order to reflect the agents' behavior (such as respecting speed limits) or other factors. We consider using $\mathbf{u}_k$ in our application in future.

In cellular localization scheme we are discussing in this paper, only the position can be measured. As a result of this, measurement vector is defined as $\mathbf{z}_k = \left[\mathbb{E}(X_k|C_k)\right]$ where C_k is the random variable of the observed cell and X_k is the random variable for the position. The distribution $\mathbb{P}(X_k|C_k)$ is sampled in the same way as it is done in Section 3.1. The slight difference is that this time positions are not mapped to road sections in drivetraces since we do not require discretization anymore.

Similarly, measurement error is given as $\mathbf{R}_k = \left[Var(X_k|C_k)\right]$. In order to avoid the errors arising from the biased sampling in the drivetraces, we set a lower-bound r_{min} on the variance of the measurement noise. In particular, we set $\mathbf{R}_k = \left[min\left(Var(X_k|C_k), r_{min}\right)\right]$. With this setting of $\mathbf{R}_k$, we are fitting a Gaussian distribution on top of $\mathbb{P}(s|c)$. Mean of this Gaussian is $\mathbf{z}_k$ and its variance is $\mathbf{R}_k$.

We set initial values $\hat{\mathbf{x}}_0 := [\mathbf{z}_0 \quad \mathbb{E}[v_0]]^T$ and

$$\mathbf{P}_0 := \begin{bmatrix} \mathbf{R}_0 & 0 \\ 0 & Var(v_0) \end{bmatrix}$$

where v_0 is the random variable of the speed of the driver initially.

As we discussed in Appendix B, $\hat{\mathbf{x}}_0 := [\mathbf{z}_0 \quad \mathbb{E}[v_0]]^T$ guarantees that our filter is unbiased under the assumption that the measurement noise is Gaussian. We select the entries in $\mathbf{P}_0$ with values reflecting variability of the position and the speed of the agent. Unfortunately, we do not have access to the statistics of v_0 directly. Here, we propose two options: the first is to assume that, given the speed limit L, for the road v_0 has a normal distribution with mean $L/2$ km/h and a variance value that would imply that with 95% confidence, the driver travels with a speed between 0-L km/h.

The second and more advanced approach would be to infer statistics of v_0 using the last speed estimation produced in previous iterations of the algorithm for the section covering the point $\mathbf{z}_0$. For instance, consider the following scenario: we get our first observation for a car. There exists a section s such that $\mathbf{z}_0 \in s$ holds. Assuming that there have been other cars that are estimated to be located at s in last 15 minutes, we can use estimates for those cars to get some idea about the level of congestion in the section.

5.3 Subsequent Section Interpolation

Kalman filter requires nothing other than the underlying physical principles of the system observed. In cellular localization case, we model the underlying physics of the system with two equations: one for the position and one for the speed of the driver. This is, however, a very simplistic view on the actual dynamics of the cars moving across the highway. As our observations contain only the position, and have high variability, there is no evidence

that replacing this simplistic model in Kalman filter can yield better estimations. In fact, we can deploy a more advanced model using the output of the Kalman filter to describe people's driving pattern.

A *spline function* is a piecewise function that consists of polynomial pieces joined together. Spline functions satisfy continuity in zeroth, first, second and so on depending on the degree of the spline. For example, the piecewise linear function is a spline of degree 1 which is linear polynomials joined together to achieve continuity in zeroth derivative (the function itself). Spline of degree 3 is the most used function in the family of spline functions.

The points $t_0 < t_1 < \ldots < t_k$ we are fitting a function to is called **knots**. The explicit form of a spline is,

$$S(x) = \begin{cases} S_{(0)}(x) & x \in [t_0, t_1] \\ S_{(1)}(x) & x \in [t_1, t_2] \\ \vdots & \\ S_{(k-1)}(x) & x \in [t_{k-1}, t_k] \end{cases} \tag{5.6}$$

In nth order spline, each of these pieces $S_{(i)}$ are at most nth order polynomials:

$$S_{(0)}(x) = a_0^{(0)} + a_1^{(0)} x + a_2^{(0)} x^2 + \cdots + a_{n-1}^{(0)} x^{n-1} + a_n^{(0)} x^n \tag{5.7}$$

and $S, S', S'', \cdots, S^{(k-1)}$ is continuous in the domain $[t_0, t_k]$.

Spline interpolation is a form of interpolation in which, interpolated curve (of 3 or higher) is a spline and curvature of the curve is minimized (one can imagine that the curve always follows the minimum gradient in the curvature formula). The curvature of the curve $y = S(x)$ is given by:

$$k = \frac{\ddot{y}}{(1 + \dot{y}^2)^{3/2}} \tag{5.8}$$

Spline interpolation is a useful approximation to driver behavior in the road because:

- 3 or higher order spline produces continuous speed and acceleration profile.

- It gives the smoothest curve.

We compute cubic spline (degree three) on the data points collected from Kalman algorithm (See Figure 5.1). The x-axis of the spline curve is the position of a probe. The y-axis of the spline is velocity of the probe (See Figure 5.2). Due to the requirement that knots must be in increasing order, we compute longest increasing subsequence of driver's recent driving history (Note that we cannot change the order of the data points as it has to be sorted by time as well). For each driver, Algorithm 1 computes speed map for the sections the driver covered in last 150 seconds. Cubic spline allows us computing speeds for the sections that does not cover any data point in drivers' history. Speed map for each user is aggregated in the last stage — and beyond the scope of the Algorithm 1—. The final speed estimates for each section is determined based on this aggregated speed maps.

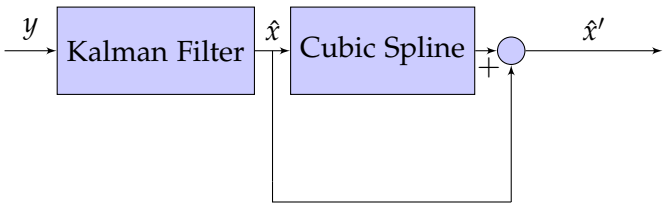

Figure 5.1: Block diagram showing how interpolation is performed

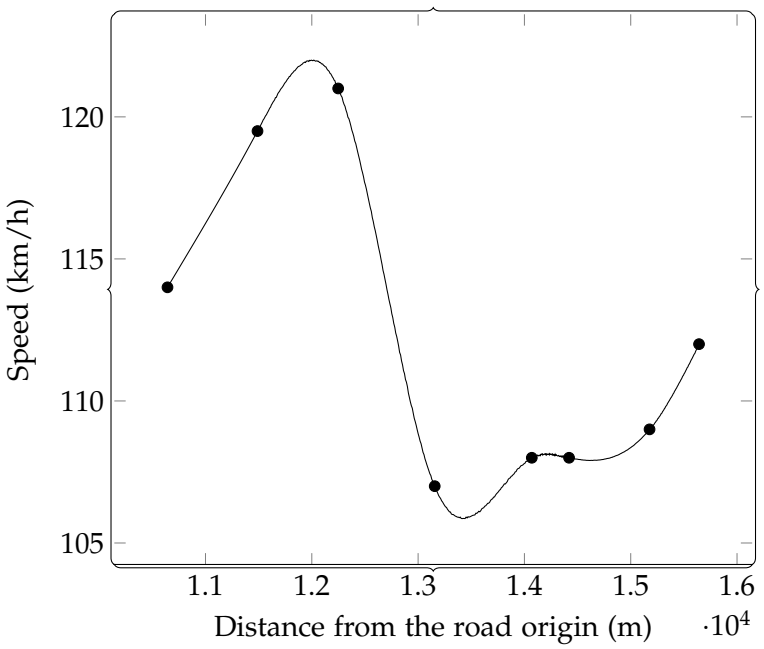

Figure 5.2: Visualisation of spline interpolation on an example drive on the road.

Algorithm 1 Cubic Spline Subsequent Section Interpolation

Input: *history*: $[\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_k]$ ▷ Old Kalman states
Input: *batch*: $[\hat{\mathbf{x}}_{k+1}, \ldots, \hat{\mathbf{x}}_n]$ ▷ New Kalman states
Input: *road*: $[s_1, s_2, \ldots, s_m]$ ▷ Array of sections in the road
Output: *speedMap*: $[s_1 \to v_1, \ldots, s_m \to v_m]$ ▷ Map from section to velocity

1: **function** estimateSpeeds
2: Compute Longest Increasing Subsequence w.r.t. x (a.k.a position) on combined *history* and *batch* sequences. Replace *history* and *batch* with their monotonic subsequences.
3: **if** *batch* is empty or *batch* and *history* have less then 3 elements in total **then**
4: *batchSections* ← compute the sections corresponding to x component in every $\hat{\mathbf{x}}_i$ in *batch*
5: **return** the mapping between *batchSections* and $\dot{x}$ component in every $\hat{\mathbf{x}}_i$ in *batch*
6: **else**
7: *cubicSpline* ← compute position-versus-velocity cubic spline for all states in *history* and *batch* combined
8: $x_0 \leftarrow x$ component of the first state in *history* and *batch* combined
9: $x_e \leftarrow x$ component of the last state in *batch*
10: *spannedSections* ← find all sections in *road* between x_0 and x_e
11: *spannedSectionsSpeeds* ← compute interpolated speeds for the mid-point of every section in *spannedSection* using *cubicSpline*
12: *speedMap* ← the map from *spannedSections* to *spannedSectionsSpeeds*
13: *mergedMap* ← combine *speedMap* and all items in *history* and *batch* giving priority to their $\dot{x}$ components.
14: **return** unpublished estimates of *mergedMap* ▷ The estimates for last 150 seconds
15: **end if**
16: **end function**

Chapter 6

Real-time Processing System Architecture

Swisscom is the leading mobile service provider in Switzerland with 6.612.000 subscribers (Swisscom, 2016). Swisscom also provides data-driven solutions and insights through the analysis of the data created by this mobile network. For this purpose, Swisscom built a big data infrastructure that allows reliable access to the high throughput data streaming through its national telecommunication network. This infrastructure, which is based on technologies such as Kafka, Spark, Spark Streaming, and Scala allows processing millions of event coming from mobile network. In this chapter, we are going to present this infrastructure and its components.

6.1 Swisscom Big Data Architecture

6.1.1 Kafka

Apache Kafka is a distributed streaming platform that is run as a cluster on one or more services. It stores streams of records in categories called topics. 4G events in Swisscom telecommunication network are published to these topics.

6.1.2 Spark Streaming

Spark is an open-source cluster computing engine for Big Data. The engine provides high speed, fault tolerant computation. Spark Streaming is an extension of Spark API that processes live data streams. Spark Streaming job

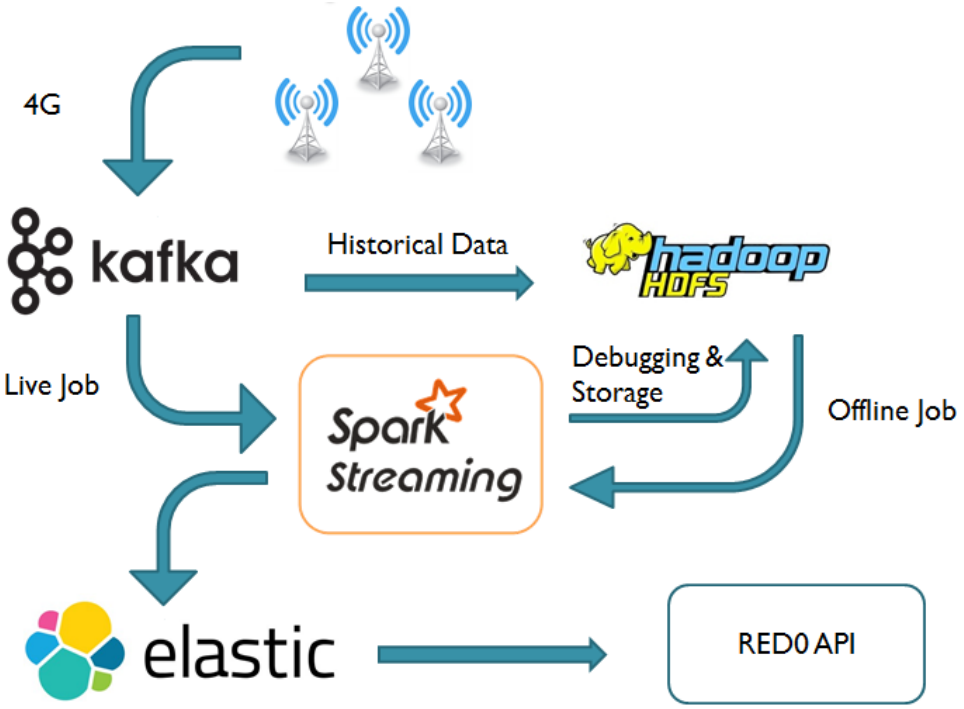

Figure 6.1: Swisscom Big Data Platform System Architecture.

connects to the Kafka stream which consists of events in the form of:

$$E = (user\_id, (timestamp, cell\_id, event\_type)) \tag{6.1}$$

Even though Kafka stream is continuous, the incoming data is partitioned by Spark Streaming into 150 seconds long chunks that are called batches. Batches are the main operational unit of Spark Streaming applications and in the context of cellular localization and speed estimation, this batching mechanism is the reason why estimations are published with 150 seconds of delay. Spark Streaming application for traffic speed estimation is structured as a pipeline of four stages. These stages are seen in Figure 6.1.2 and can be summarized as follows:

1. Not all the mobile phone users in the network is traveling in a particular highway. There is no point in performing complex computations on the events that are not coming from any of the cell towers covering some portion of the highway of interest. In addition, some residents may get connected to the base stations that is visible from a particular highway if they are working or living nearby the highway. These

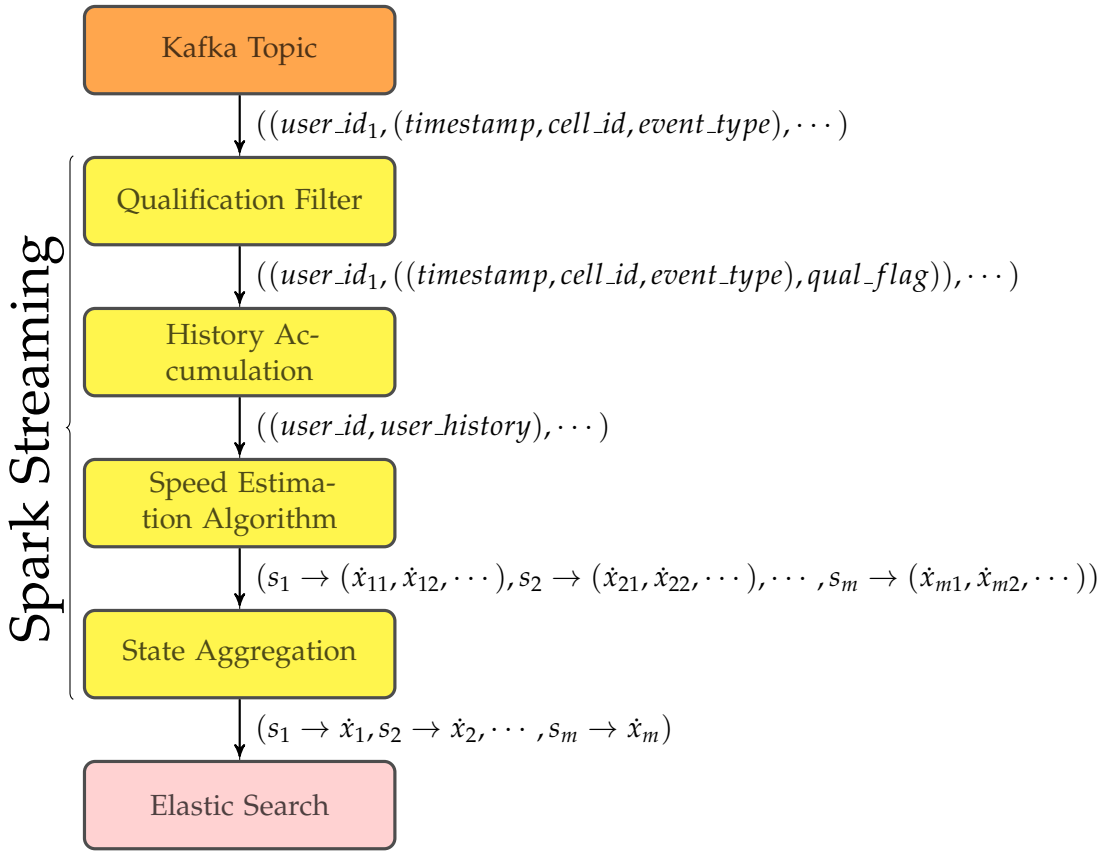

Figure 6.2: Spark Streaming Processing Pipeline. Pipeline consists of 5 stages.

events need to be filtered out as well. In short, qualification filter aims to discard all the events that are definitely not coming from drivers in the highway.

2. For all qualified users, the event history is stored in a non-persistent data structure (which is deleted after a certain period of inactivity) for speed estimation algorithms.

3. Position and speed estimation algorithms perform their computations using history of events and any static data that could be needed for estimation. The algorithms we propose in this report run in this step of the pipeline. The output of this stage is a mapping from road sections to a list of traffic speed estimates for that section.

4. The list of state estimates coming from different users are aggregated to a one to one mapping from road sections to traffic flow speed in that section. Usually, one section receives multiple estimates from mul-

tiple users. Since every section has to be mapped into a final speed, a fusion/selection algorithm reduces the list of estimates into a single estimate. One simple selection algorithm is to return the upper quartile (a.k.a. third quartile or 75th percentile) of the list of estimates for a section as the final estimation.

6.1.3 Hadoop/YARN

The Hadoop Distributed File System (HDFS) is an broadly fault-tolerant distributed file system designed to operate on commodity machines, scale horizontally, and provide high throughput access to application data. The set of computers on which HDFS is deployed is called Hadoop cluster. HDFS is suitable for applications that have large data sets: a typical file in HDFS is gigabytes to terabytes in size. YARN is the next generation of Hadoop data operating system providing cluster resource management, running on a Hadoop cluster. It acts like an interface between HDFS processing units and various data processing applications such as Spark.

In Swisscom Big Data Architecture, applications operate on a Hadoop cluster. Kafka stream of historical network events are kept in HDFS. Also, the Spark Streaming Application we present in section 6.1.2, which runs on Hadoop/YARN cluster, can be run on offline data by reading old network events from the HDFS and can write debugging information into there.

6.1.4 Elasticsearch

Elasticsearch is an open-source, broadly-distributable, readily-scalable, enterprise-grade search engine (Vanderzyden, 2015). Accessible through an extensive and elaborate API, Elasticsearch can power extremely fast searches that support data discovery applications. In big data system architecture given in Figure 6.1, speed estimates produced by Spark Streaming application is written to an Elasticsearch index. Any client application accesses this data and visualizes it through RED0API.

6.2 Discussion

In this chapter, we give details of Swisscom Big Data Platform system architecture. The primary goal for explaining this architecture is to give an

insight as to how any road traffic estimation algorithm we propose can be implemented in a Swisscom product. We believe that putting the methods we propose in Chapter 4 and 5 into a context of a real, sophisticated traffic state estimation application will be beneficial for the reader in a great extent.

Chapter 7

Experiments and Results

We face several challenges in evaluating the algorithms we propose in previous chapters. The first and the biggest challenge is lack of availability of *ground truth* for the real traffic state. There is no reliable method for getting the ground truth information in such a real-time system we operate: posing reliable estimations is our primary goal in the first place. Fortunately, there are several things we can still do: (1) comparing estimates against other real-time estimation services, (2) running algorithms on offline data as if it was real-time, and comparing the results against GPS drive traces. The key point in the latter approach is that those traces must be separate from the traces we use for generation of prior distributions; otherwise we may fall in the trap of over-fitting. The second challenge is our incapability of testing our algorithms in different scenarios that may take place in traffic; because, as we are not operating on a simulation, we have no control over the what is happening in the highway. One sometimes need to wait a couple of days in order to see a congestion in the highway.

7.1 The setup

7.1.1 Algorithms in comparison

In our experiments, we benchmark three methods against each other. Those methods are as follows:

Kalman filter & spline interpolation: This is the implementation of the approach we propose in Chapter 5. We will call this algorithm as kspline in

the rest of this chapter.

HMM based estimation: This is the implementation of the approach we propose in Chapter 4. We will call this algorithm as hmm.

Swisscom Redzero: Swisscom Redzero is the traffic speed estimation solution that has been developed by Swisscom and currently used in their product. It is based on the same system architecture that we mention in Chapter 6. However, position and speeds of the vehicles are determined by a heuristic algorithm. We can summarize the logic heuristic algorithm uses in two parts:

- Vehicle positioning is performed using a static discrete probability distribution $\mathbb{P}(s|c)$. This distribution is a combination of $\tilde{\mathbb{P}}(s|c)$ and a prior distribution.
- Vehicle speed is estimated by comparing consequent cell events c_i, c_{i+1} and probability distributions they induce. The distribution of vehicle's displacement can be computed using convolution of two distributions $\mathbb{P}(s|c_i)$ and $\mathbb{P}(s|c_{i+1})$. As the time difference between two cell events is known, the distribution of vehicle's speed can be computed. If the distribution has high variance, the algorithm goes back in history and compares c_{i-1}, c_{i+1}. This continues until a confident estimation can be done.

Aside from these three methods, we use the following 3rd party services for comparison:

- Google Maps Api allows us to query for estimated drive time between two points in real-time.
- Here Traffic is another service providing a similar kind of functionality.
- Tomtom is a telematics company specialized in traffic, navigation, and mapping products. We use their Live Traffic Api service.

7.1.2 System Configuration

The specs of the Hadoop cluster these algorithms run is:

- **VCores total:** 1152

- **Memory total:** 5.06 TB
- **Maximum Allocation:** memory:221184, vCores:48
- **Number of Nodes:** 36

7.1.3 The monitored highway

Experiments have been performed on Bern-Zurich highway. The sections referred in these experiments extend between the following geographical coordinates:

- **Section 2000:** (47.4332260N, 8.3580800E), (47.4306180N, 8.3633180E)
- **Section 2432:** (47.4288930N, 8.3684140E), (47.4309710N, 8.3628500E)
- **Section 3294:** (47.2491710N, 7.6664580E), (47.2518780N, 7.6721930E)
- **Section 3683:** (47.2519770N, 7.6721320E), (47.2492810N, 7.6663790E)
- **Section 3284:** (47.2241120N, 7.6112610E), (47.2254170N, 7.6140340E)
- **Section 3693:** (47.2263210N, 7.6160310E), (47.2241890N, 7.6111950E)

7.2 Experiments

7.2.1 Comparison of Time Series

Figure 7.1 shows time series of Google, Tomtom, Here, and Algorithm 1 during the moderate congestion that took place in Section 3693 on 12 August 2017. In the early morning and in the evening, average speeds reach 120 km/h (speed limit) in all algorihms, which is the usual flow speed when there is no traffic. All methods respond to the congestion that started around 11:00 AM.

Figure 7.2 shows the comparison between speed estimations from kspline, redzero, and hmm algorithms for Section 3693 and Section 3284 within the same time-frame as that of Figure 7.1. We observe that the algorithm hmm fails to detect the congestion. Kspline and redzero display similar results in these two sections.

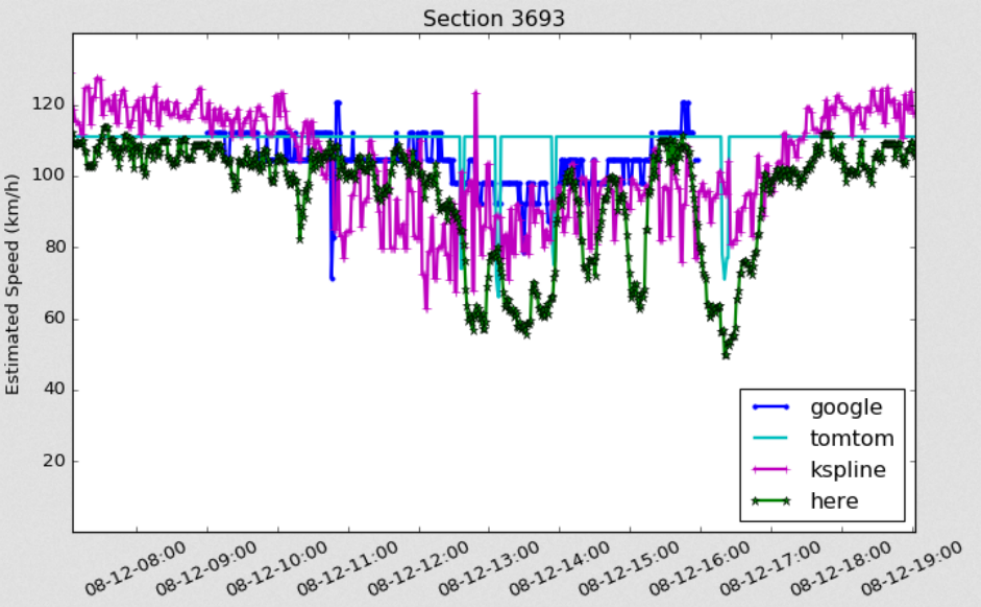

Figure 7.1: Speed estimations from Google, Tomtom, Here, and Algorithm 1 (referred as kspline) for Section 3693 is monitored during a day with moderate congestion. All of these methods predict the congestion.

7.2.2 Coverage Analysis

In the ideal scenario, traffic state estimation is expected to cover all parts of the highway: that is, it is supposed to output speed estimates for all the sections. Coverage of an algorithm is defined as below:

$$100 \cdot \frac{\text{the number of sections some estimate is output}}{\text{the total number of sections in the highway}}$$

Figure 7.3 and Table 7.1 show the analysis of Bern-Zurich highway coverage for redzero, kspline, and hmm algorithms, measured on 09 August 2017. Kspline has superior coverage compared to redzero algorithm, almost reaching to 100% in average (see Table 7.1).

Method	Coverage in avg.
kspline	96.857
hmm	95.001
redzero	83.566

Table 7.1: Coverage values have been averaged over time.

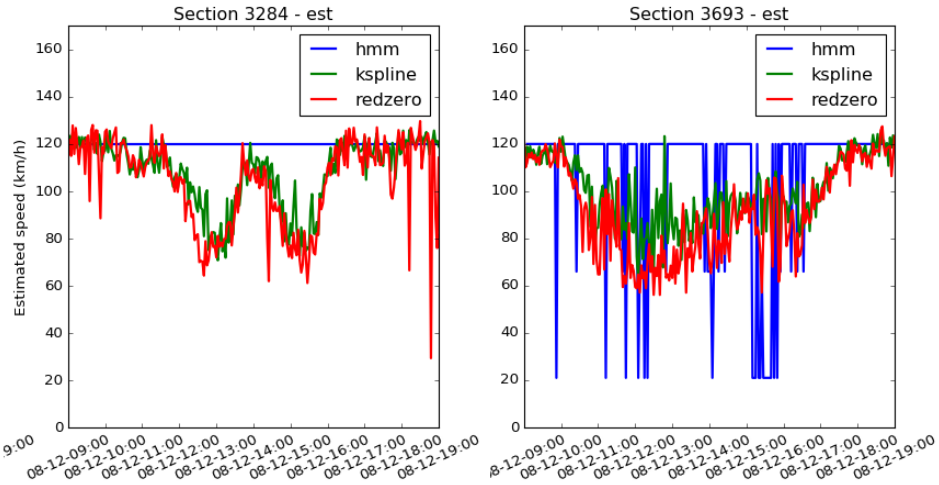

Figure 7.2: Speed estimations from kspline, redzero, and hmm algorithms for Section 3693 and Section 3284 is monitored during a day with moderate congestion. HMM fails to detect the congestion.

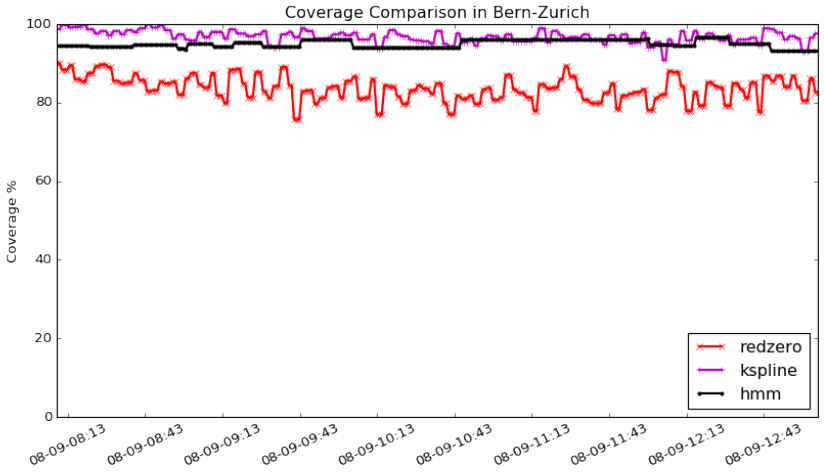

Figure 7.3: Coverage of an algorithm indicates the fraction of sections whose level of congestion is output. Bern-Zurich highway coverage comparison between hmm, kspline, and redzero algorithms shows that kspline is very close to having full coverage over the highway.

7.2.3 Speed Aggregation - Percentiles

In the last step of the Spark streaming pipeline (see Figure 6.1.2), the state estimates coming from different users are aggregated to a one to one mapping from road sections to traffic flow speed in that section. When one section receives multiple estimates from multiple users, a simple selection algorithm for determining the final estimation is to use percentiles values.

We employ 50th, 75th, and 85th percentile strategies on top of kspline algorithm. Kspline algorithm with different percentile strategies are put into a test in which the speed estimation output is compared against each other and against *external reference*. We monitor the highway using three 3rd party services, Google, Here, and Tomtom as well. We call the average of the expected drive times across the section estimated by those services as external reference. That corresponds to the harmonic average of speed estimates those services provide.

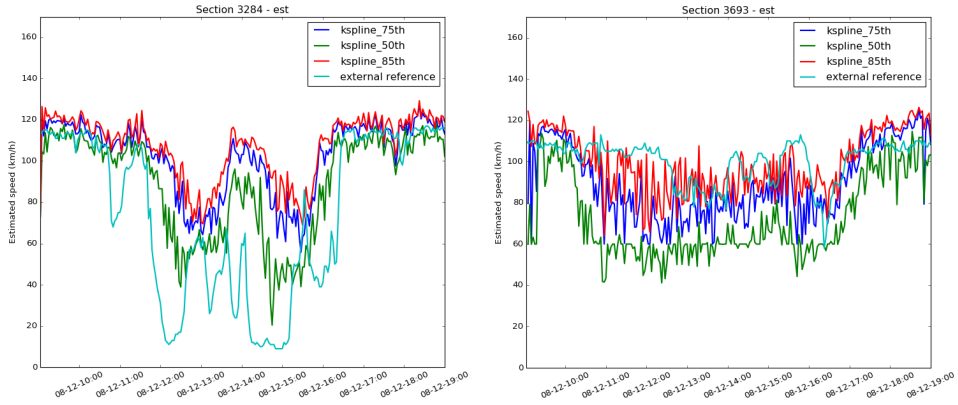

Figure 7.4: The list of speed estimates coming from different users are aggregated to a one to one mapping from road sections to traffic flow speed in that section. 50th, 75th, and 85th percentile of the group of individual estimations selected as the final estimation for the road section. We observe that in section 3693, 85th percentile approximates the external reference while in section 3284, we observe the opposite behavior.

Figure 7.4 illustrates the outcome of the test sections 3284 and 3693 was monitored for 9 hours. In section 3284, 50th percentile (median) approach gives the best approximation to external reference. On the other hand, in section 3693, 85th percentile gives the best approximation until 17:00. This might be the result of the followings:

- $\mathbb{P}(s|c)$ might be sampled with a bias in cells covering either of sections 3284 and 3693. Differences in skewness of $\mathbb{P}(s|c)$ may result in underestimating or overestimating the actual traffic flow in a section.

- The external reference might be biased in one of these sections.

7.3 Discussion

The biggest challenge in performing experiments with quantitative results is the absence of ground truth data. Without ground truth information, it is impossible to compute an error rate for different algorithms. However, evidenced from the first experiment, cellular data based methods successfully detect the level of congestion in some sections. The Kalman filter based algorithm, kspline, and redzero algorithm overshadow HMM based approach. HMM based algorithm we propose does not react to any level of congestion. However, kspline allows us to reach near 100% coverage across Bern-Zurich highway, which is not achievable by Swisscom Redzero. This illustrates that better coverage is achievable without sacrificing precision.

Until we get access to some sort of ground truth data, sections can be separately asdjusted to use the percentile that reduces the distance between output of 3rd party solutions and kspline, and thus the bias can be reverted (or be matched with that of 3rd party services) in some degree.

Chapter 8

Conclusion

In this report, we examined algorithms for extracting real-time traffic information on highways based on noisy cellular localization data. We gave an example of a pattern we confronted with in drivetraces: the distribution $\mathbb{P}(s|c)$ was not following a normal distribution. We proposed two algorithms for speed estimation: each had their own advantages and disadvantages. Even though Hidden Markov model is not good at dealing with asynchronous observations, we handled that constraint by discretizing time and introducing erasures. Other disadvantages of HMM based solution, (1) time complexity, (2) requirement for a Markov chain which imitates drivers' behavior in the highway, remained. The experiments revealed that those disadvantages impaired the quality of the estimation.

On the other hand, Kalman filter based solution, when combined with spline interpolation, produces high quality estimates with a high coverage across the highway. We advised to reduce the bias of Kalman algorithm by (1) increasing volume of offline training using drivetraces, (2) picking the optimal percentile for each section in speed aggregation stage. However, correct interpretation of the events from cells with skewed or bi-modal probability distribution using Kalman or another filter is a future work. Likewise, using a more complex Kalman filter model, for instance, introducing process error and user inputs to the current model is a future work.

We proposed two methods for estimating v_0 with minimum amount of bias. Implementing the second and more complex method and comparing it against the first method is a future work.

Appendix A

The Markov Chain Modeling of Active Drivers in the Highway

We start with quantizing the possible range of speeds a driver can reach. Three speed profiles we use correspond into following range of speeds:

1. $SLOW$ = 0-42 km/h

2. MID = 42-90 km/h

3. $FAST$ = 90-150 km/h

We want to remind our assumption that road R is divided into continuous equal length segments, given in the sorted order $s_1, s_2, \cdot, s_m$. Each of these segments have the length l (in kilometers). We call the time difference between a state transition $\delta(S_i, S_{i+1})$ as Δt (in hours). The transition

probabilities are given as follows:

$$
\begin{aligned}
Pr((s_{i+k}, SLOW)|(s_i, SLOW)) &= 4/Z \quad \forall k : 0 \leq kl \leq 48\Delta t \\
Pr((s_{i+k}, SLOW)|(s_i, MID)) &= 1/Z \quad \forall k : 0 \leq kl \leq 48\Delta t \\
Pr((s_{i+k}, SLOW)|(s_i, FAST)) &= 1/Z \quad \forall k : 0 \leq kl \leq 48\Delta t \\
Pr((s_{i+k}, MID)|(s_i, SLOW)) &= 1/Z \quad \forall k : 48\Delta t < kl \leq 90\Delta t \\
Pr((s_{i+k}, MID)|(s_i, MID)) &= 4/Z \quad \forall k : 48\Delta t < kl \leq 90\Delta t \\
Pr((s_{i+k}, MID)|(s_i, FAST)) &= 1/Z \quad \forall k : 48\Delta t < kl \leq 90\Delta t \\
Pr((s_{i+k}, FAST)|(s_i, SLOW)) &= 1/Z \quad \forall k : 90\Delta t < kl \leq 120\Delta t \\
Pr((s_{i+k}, FAST)|(s_i, MID)) &= 1/Z \quad \forall k : 90\Delta t < kl \leq 120\Delta t \\
Pr((s_{i+k}, FAST)|(s_i, FAST)) &= 4/Z \quad \forall k : 90\Delta t < kl \leq 120\Delta t \\
Pr((s_{i+k}, FAST)|(s_i, SLOW)) &= 1/2Z \quad \forall k : 120\Delta t < kl \leq 150\Delta t \\
Pr((s_{i+k}, FAST)|(s_i, MID)) &= 1/2Z \quad \forall k : 120\Delta t < kl \leq 150\Delta t \\
Pr((s_{i+k}, FAST)|(s_i, FAST)) &= 2/Z \quad \forall k : 120\Delta t < kl \leq 150\Delta t
\end{aligned}
$$

where Z is a normalizing constant whose value depends on variables such as $i, \Delta t, l$ and the speed mode.

This Markov chain follows principles we give in Chapter 4 because:

Conservation of speed: Transition probability within the same speed profile has the weight $4/Z$, while switching into another speed profile has the weight $1/Z$.

Obeying the speed limits: Transition probability to the sections too far away that it would imply that your speed is above 150 km/h is zero. The range from 120 km/h to 150 km/h is possible, but has half of the weight of the range 90 km/h - 120 km/h.

Appendix B

Simple Derivation of Kalman Filter

Kalman filter is an optimal —optimal in a sense that it minimizes the mean square error (MSE) of the estimated parameters— , unbiased, linear, and a recursive filter. The goal of this section is to provide the reader with a simple and intuitive derivation of Kalman filter. We feel the necessity to write this tutorial since the other tutorials and derivations sacrifice either completeness or simplicity. The reader can refer to the algorithm description in Section 5.1 for the meaning of the variables we use in this section. (The derivation of we present in this section is based on the following works Grewal and Bass, 1995; Reid, 2001; Cazan, 2011).

A minimum variance unbiased estimator (MVUE) is a MSE estimator which is unbiased. In other terms,

$$\hat{\mathbf{x}}_k = \underset{\hat{\mathbf{x}}_k}{\operatorname{argmin}} \quad \mathbb{E}\left[\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|_{l^2}^2\right] \tag{B.1}$$

$$\mathbb{E}[\hat{\mathbf{x}}_k] = \mathbb{E}[\mathbf{x}_k] \tag{B.2}$$

Right hand side of the Equation B.1 is called variance of error. Bias-variance tradeoff for MSE is (for simple derivation, see Lebanon, 2010):

$$MSE(\theta) = Var(\theta) + [Bias(\theta)]^2 \tag{B.3}$$

Therefore among the unbiased estimators, the estimator that minimizes the variance is the optimal MSE estimator. For this reason, as long as Equation B.1 and Equation B.2 holds, Kalman filter is a MSE estimator.

In Kalman filter, the evolution of the state and observation is modeled by the following linear equations:

$$\mathbf{x}_k = \mathbf{A}_{k-1} \cdot \mathbf{x}_{k-1} + \mathbf{w}_{k-1} \tag{B.4}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{B.5}$$

where $\mathbf{w}_k$ is Gaussian process noise with distribution $N(0, \mathbf{Q}_k)$. and $\mathbf{v}_k$ is Gaussian measurement noise with distribution $N(0, \mathbf{R}_k)$. For sake of a simple and easy-to-read analysis, we drop $\mathbf{B}\mathbf{u}_k$ term (control input) from the state evolution equation. We assume that the $\mathbf{w}_i$ and $\mathbf{v}_i$ are i.i.d. processes, and $\mathbf{w}_k$ and $\mathbf{v}_l$ are uncorrelated for every k and l.

We assume that $\mathbb{E}[\hat{\mathbf{x}}_0] = \mathbb{E}[\mathbf{x}_0]$ holds; because, later on we will show that this assumption is required for proving unbiasedness of Kalman filter.

Our first observation is that the filter we are trying to derive is linear, which provides the intuition that *a posteriori* state estimate can be written as linear combination of *a priori* state estimate and the observation value:

$$\hat{\mathbf{x}}_k = \mathbf{K}'_k \hat{\mathbf{x}}_k^- + \mathbf{K}_k \mathbf{z}_k \tag{B.6}$$

where $\mathbf{K}_k$ and $\mathbf{K}'_k$ are weighing matrices (for the proof that optimal estimator in linear form is equivalent to optimal nonlinear estimator, the reader can refer to Grewal and Bass (1995)). So our main goal is to find values of these gain matrices which satisfy Equation B.1.

Theorem B.1 *In linear MVUE,* $\boldsymbol{K}'_k + \boldsymbol{K}_k \boldsymbol{H}_k = I$

Proof As inductive step, assume that $\mathbb{E}[\hat{\mathbf{x}}_{k-1}] = \mathbb{E}[\mathbf{x}_{k-1}]$ holds.

$$\hat{\mathbf{x}}_k = \mathbf{K}'_k \hat{\mathbf{x}}_k^- + \mathbf{K}_k \mathbf{z}_k \tag{B.7}$$

$$= \mathbf{K}'_k \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k) \tag{B.8}$$

We take expectation of both sides,

$$\mathbb{E}[\hat{\mathbf{x}}_k] = \mathbf{K}'_k \mathbb{E}[\hat{\mathbf{x}}_k^-] + \mathbf{K}_k \mathbf{H}_k \mathbb{E}[\mathbf{x}_k] + \mathbf{K}_k \mathbb{E}[\mathbf{v}_k] \tag{B.9}$$

$$= \mathbf{K}'_k \mathbb{E}[\hat{\mathbf{x}}_k^-] + \mathbf{K}_k \mathbf{H}_k \mathbb{E}[\mathbf{x}_k] \tag{B.10}$$

From Equation B.4 and the inductive step,

$$\mathbb{E}[\mathbf{x}_k] = \mathbb{E}[\mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{w}_{k-1}] \tag{B.11}$$

$$= \mathbb{E}[\mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1}] = \mathbb{E}[\hat{\mathbf{x}}_k^-] \tag{B.12}$$

Therefore,

$$\mathbb{E}[\hat{\mathbf{x}}_k] = (\mathbf{K}_k' + \mathbf{K}_k\mathbf{H}_k)\mathbb{E}[\mathbf{x}_k] \tag{B.13}$$

$\mathbf{K}_k' + \mathbf{K}_k\mathbf{H}_k = I$ ensures that $\hat{\mathbf{x}}_k$ unbiased provided that the base case for the induction holds. By assuming $\mathbb{E}[\hat{\mathbf{x}}_0] = \mathbb{E}[\mathbf{x}_0]$, we conclude the proof. □

Corollary B.2 *By substituting $\boldsymbol{K}_k'$ with $I - \boldsymbol{K}_k\boldsymbol{H}_k$, we get that our unbiased estimator takes the form:*

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}_k\hat{\boldsymbol{x}}_k^-) \tag{B.14}$$

The matrix $\mathbf{K}_k$ in this equation is called Kalman gain. We need to identify optimal Kalman gain to minimize variance error. We introduce variables for ease of calculations,

$$\mathbf{e}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k \tag{B.15}$$

$$\mathbf{e}_k^- = \hat{\mathbf{x}}_k^- - \mathbf{x}_k \tag{B.16}$$

A priori and a posteriori estimation error become,

$$\mathbf{P}_k^- = \mathbb{E}[\mathbf{e}_k^-(\mathbf{e}_k^-)^T] \quad \mathbf{P}_k = \mathbb{E}[\mathbf{e}_k\mathbf{e}_k^T] \tag{B.17}$$

Using these variables, we can express the variance of error too,

$$\mathbb{E}\left[\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|_{l^2}^2\right] = \mathbf{P}_k = \mathbb{E}[\mathbf{e}_k^T\mathbf{e}_k] = tr(\mathbf{P}_k) \tag{B.18}$$

Theorem B.3 *The optimal value of the Kalman gain $\boldsymbol{K}_k$ that minimizes $tr(\boldsymbol{P}_k)$ is,*

$$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_k^-\boldsymbol{H}^T + \boldsymbol{R})^{-1} \tag{B.19}$$

Proof We start with finding the relation between $\mathbf{e}_k$ and $\mathbf{e}_k^-$,

$$\mathbf{e}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k \tag{B.20}$$

$$= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-) - \mathbf{x}_k \tag{B.21}$$

$$= \hat{\mathbf{x}}_k^- - \mathbf{x}_k + \mathbf{K}_k(\mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-) \tag{B.22}$$

$$= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{e}_k^- + \mathbf{K}_k\mathbf{v}_k \tag{B.23}$$

Then the value of $\mathbf{e}_k\mathbf{e}_k^T$ becomes,

$$\mathbf{e}_k\mathbf{e}_k^T = \Big((\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbf{e}_k^- + \mathbf{K}_k\mathbf{v}_k\Big)\Big((\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbf{e}_k^- + \mathbf{K}_k\mathbf{v}_k\Big)^T \tag{B.24}$$

$$\begin{aligned}&= (\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbf{e}_k^-(\mathbf{e}_k^-)^T(\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)^T - \\ &(\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbf{e}_k^-\mathbf{v}_k^T\mathbf{K}_k^T - \mathbf{K}_k\mathbf{v}_k(\mathbf{e}_k^-)^T(\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)^T + \\ &\mathbf{K}_k\mathbf{v}_k\mathbf{v}_k^T\mathbf{K}_k^T\end{aligned} \tag{B.25}$$

We take expectation of both sides. Note that $\mathbf{e}_k^-$ and $\mathbf{v}_k$ are uncorrelated.

$$\begin{aligned}\mathbb{E}[\mathbf{e}_k\mathbf{e}_k^T] &= (\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbb{E}[\mathbf{e}_k^-(\mathbf{e}_k^-)^T](\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)^T + \\ &\quad \mathbf{K}_k\mathbb{E}[\mathbf{v}_k\mathbf{v}_k^T]\mathbf{K}_k^T\end{aligned} \tag{B.26}$$

$$\mathbf{P}_k = (\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \tag{B.27}$$

We derived the equation relating the a posteriori estimation error covariance to a priori estimation error covariance, Kalman gain and measurement covariance.

Lemma B.4 *For a symmetric matrix $\boldsymbol{B}$, a differentiable function $F(\boldsymbol{A})$ with its scalar derivative $f(\cdot)$,*

$$\frac{\partial tr\big(F(\boldsymbol{A})\boldsymbol{B}F(\boldsymbol{A})^T\big)}{\partial \boldsymbol{A}} = 2F(\boldsymbol{A})\boldsymbol{B}f(\boldsymbol{A}) \qquad \textit{(see Petersen et al., 2008)} \tag{B.28}$$

Using Lemma B.4, we differentiate Equation B.27 with respect to Kalman gain and set equal to zero in order to find the Kalman gain matrix which minimize $tr(\mathbf{P}_k)$,

$$\frac{\partial\Big((\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T\Big)}{\partial\mathbf{K}_k} = 0 \tag{B.29}$$

$$-2(\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-\mathbf{H}_k^T + 2\mathbf{K}_k\mathbf{R}_k = 0 \tag{B.30}$$

After making necessary substitutions, we find,

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1} \tag{B.31}$$

□

As we now know the optimal Kalman gain, we can revisit a posteriori error covariance equation (Equation B.27), which is so-called *Joseph form*, for the purpose of getting a simpler version.

$$
\begin{aligned}
&(\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)^T+\mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T=\mathbf{P}_k^- - \\
&\mathbf{K}_k\mathbf{H}_k\mathbf{P}_k^- - \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{K}_k^T+\mathbf{K}_k\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T\mathbf{K}_k^T+\mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T && \text{(B.32)}\\
&=(\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- - \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{K}_k^T+ \\
&\mathbf{K}_k(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T+\mathbf{R}_k)\mathbf{K}_k^T && \text{(B.33)}\\
&=(\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- - \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{K}_k^T+\mathbf{P}_k^-\mathbf{H}_k^T\mathbf{K}_k^T && \text{(B.34)}\\
&=(\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- && \text{(B.35)}
\end{aligned}
$$

This gives us the last one of update stage equations of Kalman filter. We derived all update stage equations. We also know how to compute a priori estimates based on Bayesian approach (although we didn't mention explicitly, but used in our analysis on Equation B.12),

$$\hat{\mathbf{x}}_k^- = \mathbf{A}_{k-1}\cdot\hat{\mathbf{x}}_{k-1} \tag{B.36}$$

The last piece of the puzzle is an equation relating a priori error covariance in time-step k to a posteriori error covariance in time-step $k-1$. Luckily, deriving this equation is simple,

$$
\begin{aligned}
\mathbf{P}_k^- &= \mathbb{E}[(\mathbf{x}_k-\hat{\mathbf{x}}_k^-)(\mathbf{x}_k-\hat{\mathbf{x}}_k^-)^T] && \text{(B.37)}\\
&= \mathbb{E}[(\mathbf{A}_{k-1}\mathbf{x}_{k-1}+\mathbf{w}_{k-1}-\mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1}^-)(\mathbf{A}_{k-1}\mathbf{x}_{k-1}+\mathbf{w}_{k-1}-\mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1}^-)^T] && \text{(B.38)}\\
&= \mathbb{E}[(\mathbf{A}_{k-1}\mathbf{e}_{k-1}+\mathbf{w}_{k-1})(\mathbf{A}_{k-1}\mathbf{e}_{k-1}+\mathbf{w}_{k-1})^T] && \text{(B.39)}\\
&= \mathbf{A}_{k-1}\mathbb{E}[\mathbf{e}_{k-1}(\mathbf{e}_{k-1})^T]\mathbf{A}_{k-1}^T+\mathbf{A}_{k-1}\mathbb{E}[\mathbf{e}_{k-1}\mathbf{w}_{k-1}^T]+ \\
&\mathbb{E}[\mathbf{w}_{k-1}(\mathbf{e}_{k-1})^T]\mathbf{A}_{k-1}^T+\mathbb{E}[\mathbf{w}_{k-1}\mathbf{w}_{k-1}^T] && \text{(B.40)}\\
&= \mathbf{A}_{k-1}\mathbf{P}_{k-1}\mathbf{A}_{k-1}^T+\mathbf{Q}_{k-1f} && \text{(B.41)}
\end{aligned}
$$

Notice that we exploited the fact that $\mathbf{e}_{k-1}$ and $\mathbf{w}_{k-1}$ are uncorrelated.

This equation concludes the derivation of the Kalman filter.

Bibliography

Saurabh Amin, Steve Andrews, Saneesh Apte, Jed Arnold, Jeff Ban, Marika Benko, Re M Bayen, Benson Chiou, Christian Claudel, Coralie Claudel, et al. Mobile century using gps mobile phones as traffic sensors: A field experiment. 2008.

Raffaele Bolla and Franco Davoli. Road traffic estimation from location tracking data in the mobile cellular network. In *Wireless Communications and Networking Confernce, 2000. WCNC. 2000 IEEE*, volume 3, pages 1107–1112. IEEE, 2000.

Irina Cazan. Kalman filters, 2011. URL https://www.colby.edu/math/program/honorsprojects/2011-Cazan-Honors.pdf.

Peng Cheng, Zhijun Qiu, and Bin Ran. Particle filter based traffic state estimation using cell phone network data. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 1047–1052. IEEE, 2006.

Christian G. Claudel and Alexandre M. Bayen. Guaranteed bounds for traffic flow parameters estimation using mixed lagrangian-eulerian sensing. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, sep 2008. doi: 10.1109/allerton.2008.4797618. URL https://doi.org/10.1109/allerton.2008.4797618.

Mohinder S Grewal and RW Bass. Kalman filtering: theory and practice. *IEEE Transactions on Automatic Control*, 40(11):1983, 1995.

Ke Han, Tao Yao, and Terry L Friesz. Lagrangian-based hydrodynamic model: Freeway traffic estimation. *arXiv preprint arXiv:1211.4619*, 2012.

Juan C. Herrera and Alexandre M. Bayen. Incorporation of lagrangian measurements in freeway traffic state estimation. *Transportation Research Part B: Methodological*, 44(4):460–481, may 2010. doi: 10.1016/j.trb.2009.10.005. URL https://doi.org/10.1016/j.trb.2009.10.005.

Simon J Julier, Jeffrey K Uhlmann, and Hugh F Durrant-Whyte. A new approach for filtering nonlinear systems. In *American Control Conference, Proceedings of the 1995*, volume 3, pages 1628–1632. IEEE, 1995.

Wei-Kuang Lai and Ting-Huan Kuo. Vehicle positioning and speed estimation based on cellular network signals for urban roads. *ISPRS International Journal of Geo-Information*, 5(10):181, 2016.

Guy Lebanon. Bias, variance, and mse of estimators, 2010. URL https://pdfs.semanticscholar.org/205c/483a68e8d58f8ab09c9bee7f82a46e27829d.pdf.

Michael J Lighthill and Gerald Beresford Whitham. On kinematic waves. ii. a theory of traffic flow on long crowded roads. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 229, pages 317–345. The Royal Society, 1955.

Lyudmila Mihaylova and Rene Boel. A particle filter for freeway traffic estimation. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 2, pages 2106–2111. IEEE, 2004.

Lyudmila Mihaylova, René Boel, and A Hegyi. An unscented kalman filter for freeway traffic estimation. *IFAC Proceedings Volumes*, 39(12):31–36, 2006.

Lyudmila Mihaylova, René Boel, and Andreas Hegyi. Freeway traffic estimation within particle filtering framework. *Automatica*, 43(2):290–300, 2007.

Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7:15, 2008.

Fatih Porikli and Xiaokun Li. Traffic congestion estimation using hmm models without vehicle tracking. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 188–193. IEEE, 2004.

Ian Reid. Estimation ii, 2001. URL http://www.robots.ox.ac.uk/~ian/Teaching/Estimation/LectureNotes2.pdf.

W Schneider. Mobile phones as a basis for traffic state information. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 782–784. IEEE, 2005.

Xiaotian Sun, Laura Muñoz, and Roberto Horowitz. Highway traffic state estimation using improved mixture kalman filters for effective ramp metering control. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 6, pages 6333–6338. IEEE, 2003.

Ye Sun and Daniel B Work. A distributed local kalman consensus filter for traffic estimation. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 6484–6491. IEEE, 2014.

Swisscom. Annual report, 2016. URL http://reports.swisscom.ch/en/2016/report/annual-report/introduction/kpis-of-swisscom-group.

Sha Tao, Vasileios Manolopoulos, Saul Rodriguez Duenas, and Ana Rusu. Real-time urban traffic state estimation with a-gps mobile phones as probes. *Journal of Transportation Technologies*, 2(1):22–31, 2012.

Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98. ACM, 2009.

Davide Tosi. Cell phone big data to compute mobility scenarios for future smart cities. *International Journal of Data Science and Analytics*, pages 1–20, 2017.

Danilo Valerio, Alessandro D'Alconzo, Fabio Ricciato, and Werner Wiedermann. Exploiting cellular networks for road traffic estimation: A survey and a research roadmap. In *VTC Spring 2009 - IEEE 69th Vehicular Technology Conference*. IEEE, apr 2009a. doi: 10.1109/vetecs.2009.5073548. URL https://doi.org/10.1109/vetecs.2009.5073548.

Danilo Valerio, Alessandro D'Alconzo, Fabio Ricciato, and Werner Wiedermann. Exploiting cellular networks for road traffic estimation: a survey and a research roadmap. In *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pages 1–5. IEEE, 2009b.

Danilo Valerio, Tobias Witek, Fabio Ricciato, Rene Pilz, and Werner Wiedermann. Road traffic estimation from cellular network monitoring: a hands-on investigation. In *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, pages 3035–3039. IEEE, 2009c.

John Vanderzyden. What is elasticsearch, and how can i use it?, 2015. URL https://qbox.io/blog/what-is-elasticsearch.

Yibing Wang and Markos Papageorgiou. Real-time freeway traffic state estimation based on extended kalman filter: a general approach. *Transportation Research Part B: Methodological*, 39(2):141–167, 2005.

Daniel B Work, Olli-Pekka Tossavainen, Sébastien Blandin, Alexandre M Bayen, Tochukwu Iwuchukwu, and Kenneth Tracton. An ensemble kalman filtering approach to highway traffic estimation using gps enabled mobile devices. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 5062–5068. IEEE, 2008.

Daniel B Work, Olli-Pekka Tossavainen, Quinn Jacobson, and Alexandre M Bayen. Lagrangian sensing: traffic estimation with mobile devices. In *American Control Conference, 2009. ACC'09.*, pages 1536–1543. IEEE, 2009.

Jungkeun Yoon, Brian Noble, and Mingyan Liu. Surface street traffic estimation. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 220–232. ACM, 2007.