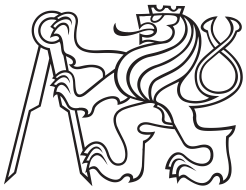
Bachelor's Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Real-time Public Transport Vehicle Delay Prediction

Emmanuil Vaganov

Supervisor: Ing. Martin Schaefer
January 2023

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Vaganov Emmanuil** Personal ID number: **453450**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Real-Time Public Transport Vehicle Delay Prediction

Bachelor’s thesis title in Czech:

Real-time predikce zpoždění vozů hromadné dopravy

Guidelines:

Research the literature on the delay prediction of public transport vehicles.
1. Study and describe the accessible data sources of real-time and historical locations of public transport vehicles.
2. Propose a pipeline for real-time delay prediction of public suburban transport vehicles in Prague.
3. Evaluate the quality of the prediction on the real-world data, compare with a reasonable baseline solution

Bibliography / sources:

[1] Ma, J., Chan, J., Ristanoski, G., Rajasegarar, S. and Leckie, C., 2019. Bus travel time prediction with real-time traffic information. Transportation Research Part C: Emerging Technologies, 105, pp.536-549.
[2] Yin, T., Zhong, G., Zhang, J., He, S. and Ran, B., 2017. A prediction model of bus arrival time at stops with multi-routes. Transportation research procedia, 25, pp.4623-4636.
[3] Sun, F., Pan, Y., White, J. and Dubey, A., 2016, May. Real-time and predictive analytics for smart public transportation decision support system. In 2016 IEEE International Conference on Smart Computing (SMARTCOMP) (pp. 1-8). IEEE.
[4] Yu, B., Lam, W.H. and Tam, M.L., 2011. Bus arrival time prediction at bus stop with multiple routes. Transportation Research Part C: Emerging Technologies, 19(6), pp.1157-1170.

Name and workplace of bachelor’s thesis supervisor:

Ing. Martin Schaefer Self-employed person, Prague

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **24.02.2022** Deadline for bachelor thesis submission: **10.01.2023**

Assignment valid until: **30.09.2023**

Ing. Martin Schaefer	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

. | Date of assignment receipt | Student’s signature

Acknowledgements

I would like to thank my supervisor Ing. Martin Schaefer for believing in me and my thesis despite the difficulties I encountered. I am also grateful to my family for their endless support and encouragement during the whole period of my study.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, January 10, 2023

.....

Abstract

Buses are one of the most popular transportation means used in Prague, and the reliability of public transport is one of the most critical aspects for commuters. Having information about expected delays allows people to plan their journey better. This work introduces a method for real-time prediction of bus delays several stops ahead of their current position.

We propose an algorithm to create and optimize a graph of public transport routes from data in General Transit Feed Specification (GTFS) format. Information about bus travel times is saved into the graph and used to predict delays of all buses passing the same road segment. We use the Kalman filter to make predictions and introduce techniques to adapt the filter to sudden changes in the traffic situation. Although the proposed solution is tested on bus positions, it can also work for other forms of public transport when using the GTFS format.

Experiments on test data show that compared to the prediction model currently used in Prague, our solution reduces prediction error by up to 30%, with the most significant improvements when comparing predictions for bus trips with widely varying delays at stops.

Keywords: delay prediction, real-time prediction, Kalman filter, public transport, GTFS

Abstrakt

Autobusy jsou jedním z nejpoužívanějších dopravních prostředků v Praze a spolehlivost veřejné dopravy je pro cestující jedním z nejpodstatnějších faktorů. Informace o očekávaném zpoždění umožňují lidem lépe plánovat cestu. Tato práce představuje metodu predikce zpoždění autobusů v reálném čase na několik zastávek dopředu.

Navrhujeme algoritmus pro vytvoření a optimalizaci grafu linek městské dopravy z dat ve formátu General Transit Feed Specification (GTFS). Informace o dobách jízd autobusů se ukládají do grafu a používají se k predikci zpoždění všech autobusů projíždějících stejným úsekem. K vytvoření předpovědí využíváme Kalmanův filtr a používáme metody pro přizpůsobení filtru náhlým změnám v dopravní situaci. Přestože je navrhované řešení testováno na autobusech, může při použití formátu GTFS fungovat i pro jiné druhy veřejné dopravy.

Experimenty na testovacích datech ukázaly, že v porovnání s modelem predikce v současnosti využívaném v Praze, snižuje naše řešení chybu až o 30%, přičemž nejvýznamnější zlepšení nastává při porovnávání predikcí pro jízdy autobusů s výrazně proměnlivým zpožděním na jednotlivých zastávkách.

Klíčová slova: predikce zpoždění, real-time predikce, Kalmanův filtr, hromadná doprava, GTFS

Contents

Figures

Tables

Chapter 1

Introduction

Hundreds of thousands of people in Prague rely on public transport for their daily commute, and almost a third of the city's transit trips involve buses [1]. According to the research conducted by Regional Organizer of Prague Integrated Transport, service reliability is the most important aspect for passengers when using public transit [2]. As the city continues to grow, commuters want to know when to expect delays, especially for bus rides, which are prone to get held up by traffic. With reliable information about the approximate time of arrival, people can better plan their time and journey.

Accurately tracking vehicles and informing passengers of estimated arrival times is a challenge because of several factors, such as weather conditions, operational delays, or varying time required to load passengers at each stop. Over the past decades, many models for predicting bus arrival times have been developed, and many different methods have been used, such as historical and real-time approaches, statistical methods, machine learning methods, and model-based methods, and it is still a very popular area of research. Not only individuals but also large corporations, such as Google, are still looking for ways to effectively solve this problem, as can be seen from their blog dedicated to research in the field of artificial intelligence [3].

As of November 2022, we found two widely known services that provide real-time information about predicted public transport delays in Prague:

- *Google Maps* [4] is a free online map. It is possible to get information about delays from the application, although only indirectly. They are provided when planning a route from one place to another, but only for vehicles relevant to that route. In addition, there is no direct access to an Application Program Interface (API) to obtain information and compare their predictions with others.

- *Golemio* [5] is a data platform containing different kinds of informa-

tion about Prague. The transit data and schedules are provided in General Transit Feed Specification (GTFS) format [6]. It contains information about delay estimates for the first stop from the current position of a vehicle, and after the first stop, the same delay is used as an estimate for all remaining stops. The API provided by Golemio is officially used by Prague Integrated Transport to display positions of public transport and delays [7].

While predicting delay only for the next stop, as Golemio does, is certainly helpful for people waiting at that stop, people in the vehicle may find information about delays for future stops useful, as it will allow them to know their expected arrival time and plan their trip accordingly. Another problem with this strategy is that if it is known that traffic congestion has formed after the first predicted stop, this method cannot adapt to such changes and still gives the same estimate.

This thesis aims to propose a method for predicting bus delays in real-time for several future stops, while adapting to the current transit situation in case of sudden changes. We will describe how to use Prague transit data and schedules in GTFS format provided by Golemio, which gives access to a significant amount of real-time data and can be utilized in solving the defined problem. Our solution will be compared to the one provided by Golemio. Although the proposed method is tested on bus positions, it is also applicable to other forms of public transport, if the GTFS format is used.

Terminology

Since we use APIs to gather data, we need to define the terminology used to describe returned objects, which is also used throughout the thesis. We also define terms commonly used in transportation research, such as *dwell time* and *travel time* [8].

Dwell time Time a vehicle spends at a scheduled stop without moving.
Stop A location where passengers are picked up or dropped off.
Travel time Time of travel between stops (including delays) or arbitrary points (e.g., from the current vehicle position to the next stop).
Delay Time a vehicle arrived at a stop compared to the scheduled time of arrival. It is negative if the vehicle arrives earlier.
Trip A sequence of two or more stops that occur during a specific time period.
Stop time Times that a vehicle arrives at and departs from stops for each trip.

Route Describes the path taken by a vehicle as a list of GPS points that the vehicle passes in the specified order. Every trip is associated with its route. The same route can be used for multiple trips.

It is worth noting that delay prediction and travel time prediction are sometimes used interchangeably in this thesis. In this case, travel time refers to the time it takes for the bus to reach a stop from its current position.

Chapter 2

Related work

Various methods in different studies have been proposed to predict bus delays or travel times. Several existing works address the following aspects:

- *Route construction.* Because buses often share the road with other buses, information about their state and surroundings may also be shared. One such information may be a transit situation on a particular road segment. For example, we can assume that traffic congestion affects the travel time of all buses passing through the section where it occurs. Instead of viewing bus routes as a sequence of GPS coordinates, they can be viewed as a sequence of road segments, and some of them can be shared. The question of how big such segments should be is discussed in numerous studies.
- *Prediction methods.* Whether to apply one method of delay prediction or another depends on many factors, such as data availability or performance requirements. Choosing the right approach is an essential part of research trying to solve the prediction problem.
- *Impact factors.* Traffic interruption, as well as weather conditions, can affect bus travel times. Some of the existing works discuss how to deal with them and what to consider.

In the following sections, we review how related works deal with the described aspects.

2.1 Route construction

From the view of route construction, stop-based [9] and segment-based [10] route construction solutions are among the more popular. Stop-based

models divide bus routes by stop locations, and segment-based models additionally divide sections between stops into segments of varying lengths. Chien et al. [11] suggests that the stop-based model is preferred when there are multiple intersections between stops, while the segment-based one is more suitable for pairs of stops with few intersections. There also is a method based on identifying shared road segments (illustrated in Figure 2.1), which makes use of overlapping bus routes to predict delays [12]. Ma et al. [13] indicates that the shared segment-based method outperforms the stop-based and segment-based methods that do not use information about overlapping routes.

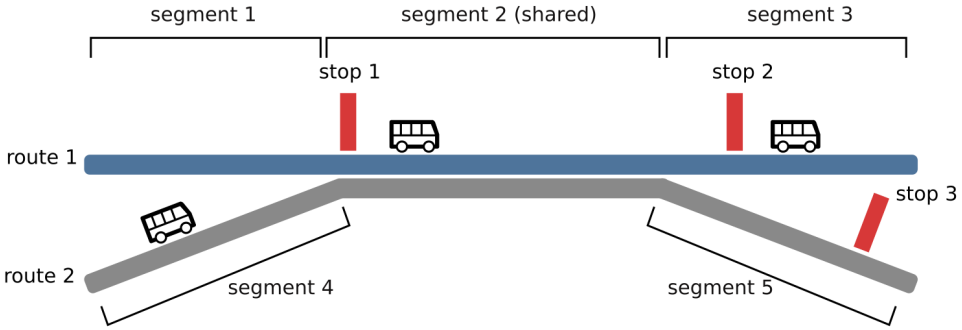


Figure 2.1: Schematic illustration of the shared segment-based method, adopted from [12]

2.2 Prediction methods

Model-based techniques use models that capture the dynamics of a system by establishing a mathematical relationship between variables. Many model-based studies use Kalman Filtering Technique (KFT) for estimation. Kalman filter is an efficient recursive procedure that estimates the future states of dependent variables. The advantages of this approach are its limited data requirement and suitability for real-time implementation. Cathey and Dailey [14] applied Kalman filtering methods for bus travel time prediction using real-time data, and used them to estimate the future state of vehicles, including speed, which is then used to predict travel times. Vanajakshi et al. [15] discussed the possibility of adopting a Kalman filtering model to directly forecast the travel time and indicated that KFT is better than simple averaging method. Sun et al. [12] utilized clustering analysis and KFT with a segment-based route construction in their study. It shows that KFT can be successfully used for delay prediction in the near future, and historical delays can be used to predict later. Shalaby and Farhan [16] use KFT to predict both travel times and dwell times. They discovered that KFT could outperform regression and neural network models in terms of accuracy.

Machine learning methods such as Artificial Neural Networks (ANN) and Support Vector Machines (SVM) are commonly used to predict travel time because of their ability to solve complex non-linear relationships. However, these methods are limited by extremely long training times. Chien et al. [11] proposed two ANN models to predict bus arrival time using link-based and stop-based route construction methods. Kumar et al. [17] compared model-based approaches and machine learning methods, and found that the ANN model is better than Kalman filtering with a large database. To verify the feasibility of SVM in time prediction field, Bin et al. [18] compared SVM with ANN with three inputs: time, weather and route, and showed that SVM provide better results than ANN. SVM models have also been shown to outperform the k-Nearest Neighbors (k-NN), historical average, and time series methods in [10, 9]. Yu et al. [19] compared four different models, namely, SVM, ANN, k-NN and linear regression, and found that the SVM model performs the best among the four proposed models for predicting the bus arrival times at bus stop with multiple routes. Yin et al. [20] compared SVM with Radial basis function kernel and ANN with 3 inputs, and used a genetic algorithm to optimize the Back Propagation Artificial neural network, which includes structure determination of ANN, optimization by the genetic algorithm, and prediction by ANN. The results of the proposed model indicated that both SVM and ANN models have high accuracy, while the ANN model is better than the SVM model comparatively.

Other methods include a non-linear time series model used by D'Angelo et al. [21] to predict bus travel time. Whereas, the poor accuracy and real-time performance make it an unsatisfactory model. Yu et al. [22] suggested that Random forest based on the near neighbor method shows very good prediction results but is slower and harder to implement, and advises to use parallel computing. Kumar et al. [23] focused on identifying the possible travel time patterns in the bus trajectories by developing a k-NN classifying algorithm that uses historical travel time trajectories and developed a prediction method that uses identified patterns to predict the bus travel time.

2.3 Impact factors

One of the main reasons existing approaches cannot achieve perfect accuracy is that public transport is affected by various factors. For example, weather conditions [24], rush hours [10], and passenger demand [25] are all shown to affect travel times. Ramakrishna et al. [26] analyzed the impact of vehicle speed and passenger flow (number of people entering and leaving the bus) on delays and showed that taking these data into account helps to improve the performance of the model.

2.4 Summary

The number of factors affecting travel times of public transport is large, and having information about some of the factors allows making more accurate predictions. Most studies use any data available, and it can be observed that the amount of data correlates with the quality of predictions. In addition, having information that can be exchanged between different vehicles, such as a sudden change in travel time on a particular segment of road, has a positive effect on the results.

If the available data is limited, the Kalman filter has been shown to give good predictions based on the current state of the system, making it a good candidate for real-time applications. Otherwise, if an extensive set of historical data is provided, machine learning techniques can be used in combination to produce the best results.

Chapter 3

Solution

In this chapter, we describe the approaches that we use to solve the defined problem, as well as discuss the factors that we considered when implementing them.

After a brief introduction to GTFS, we present a method of identifying road segments shared by different routes and describe how it is used in our solution. As discussed in the previous chapter, this approach outperforms the ones not accounting for overlapping road sections of different routes. Moreover, it was presented in [12], where GTFS data were also used, and had a positive effect on the result.

Next, we describe a Kalman filter-based model to predict travel times of buses on different road segments based on data from preceding buses, and how it can be modified to react to sudden changes in the transit situation.

We also propose additional methods to improve the robustness of the model, such as generation of missing data and automatic parameter estimation using the Expectation-Maximization algorithm.

3.1 General Transit Feed Specification

In this section, we give a very brief overview of the already mentioned General Transit Feed Specification (GTFS). Using this format is an important requirement for our solution to work, because we assume some of the parameters provided by it. The reference documentation can be found at [6].

GTFS provides information about particular aspects of public transit. Two types of data are available:

- *Static data.* Persistent aspects, such as stops, routes, or trips, are considered static information. They generally do not change much.

- *Real-time data.* Vehicle positions, trip updates (temporary changes), and service alerts are assumed to be subject to change at any time. They are often provided as a feed of data instead of static files.

Some of the essential properties needed for our solution and provided in GTFS are described below:

- *Vehicle position updates.* Every bus or another vehicle sends information about its current position at regular intervals. Every position update contains information about the time of update (timestamp), speed, trip, relevant stops, and other important information.
- *GPS coordinates.* Actual physical positions of routes, stops, and vehicle positions are described using GPS coordinates.
- *Distance traveled along route.* Other than GPS coordinates, GTFS contains information about the distance it takes to reach a certain point from the start of the route. This is a very important property for our solution and is used extensively. For example, if we want to check if a vehicle has already reached a specific stop, we can simply compare their distances.
- *Scheduled departures and arrivals.* For every stop, the scheduled time of departure and arrival is indicated.

3.2 Route graph construction

Since buses share the road with other transportation means, such as personal cars, bicycles, or other buses, their travel time is more responsive to various road conditions. Traffic congestion, accidents, and weather can all affect the resulting bus delay. Although reliably predicting such conditions is out of the scope of this thesis, we can use information about preceding buses to react to sudden changes in travel time on a road segment. Travel time does not directly indicate a traffic jam or road closure, but a quick increase in travel time on a section of road may indicate that something is happening.

If we have a road segment that is being used by multiple routes, we will call such a segment *shared*. Otherwise, if a road segment is only used by one route, we call it *simple*. We can store information about bus travel times on shared segments to use it in travel time prediction of subsequent buses.

We can describe the route graph as a directed multigraph. Since GTFS describes bus routes using a sequence of GPS coordinates, we can set

them as nodes in the route graph, and every two consecutive coordinates as edges. Thus, initially, we assume that if two edges share the same source and target nodes, then they form a shared segment represented by these edges. Then these edges can be merged into one edge that contains information about both merged edges.

Every node contains the following information:

- *Arrival and departure time for every bus passing the node.* Departure times are only calculated if the node is a stop; otherwise, it is equal to the arrival time.

Every edge contains the following information:

- *Routes passing the edge.* Shared edges contain information about multiple routes, and simple edges about only one route.
- *Latest travel time for every trip.* Calculated as the difference between the departure time from the start node and the arrival time to the end node of the edge.

After creating the route graph, we apply the following steps:

Step 1: *Merge nearby nodes.* There can be many nodes lying very close to each other in the initialized route graph, depending on the number of decimal places used to describe their coordinates. In the case of public transportation, we can consider nodes that are only a few centimeters or even meters apart to be the same. More specifically, in our solution, we round all GPS coordinates to four decimal places, which is accurate to 11.1 meters (±5.55 meters) at the equator.

Step 2: *Generate missing points.* After merging nearby nodes, there may still be shared segments left to merge, which can be created from overlapping edges, as shown on Figure 3.1a and Figure 3.1b. It should be mentioned that these figures are illustrative, and the route graph edges are shorter and more "straight" at the time of performing Step 2. To make our algorithm aware of overlapping edges, we generate missing points between every two nodes of the graph. In other words, we fill the space between every two nodes with other nodes as much as the specified GPS accuracy allows. The process of generating missing points is described by Algorithm 1.

Step 3: *Merge consecutive segments.* Two consecutive edges with the same routes passing through them can be merged into one edge by removing the common node. We do this for all edges except those that start or end at stops or intersections; the reasoning is described in the next section. This allows us to combine very small edges created by the previous steps into the longest possible shared and simple segments.

Algorithm 1 Missing points generation

Input:

$p_1^{lon}, p_1^{lat} \leftarrow$ first point longitude and latitude
$p_2^{lon}, p_2^{lat} \leftarrow$ seconds point longitude and latitude
$m \leftarrow$ required GPS precision

Output:

List of generated points r

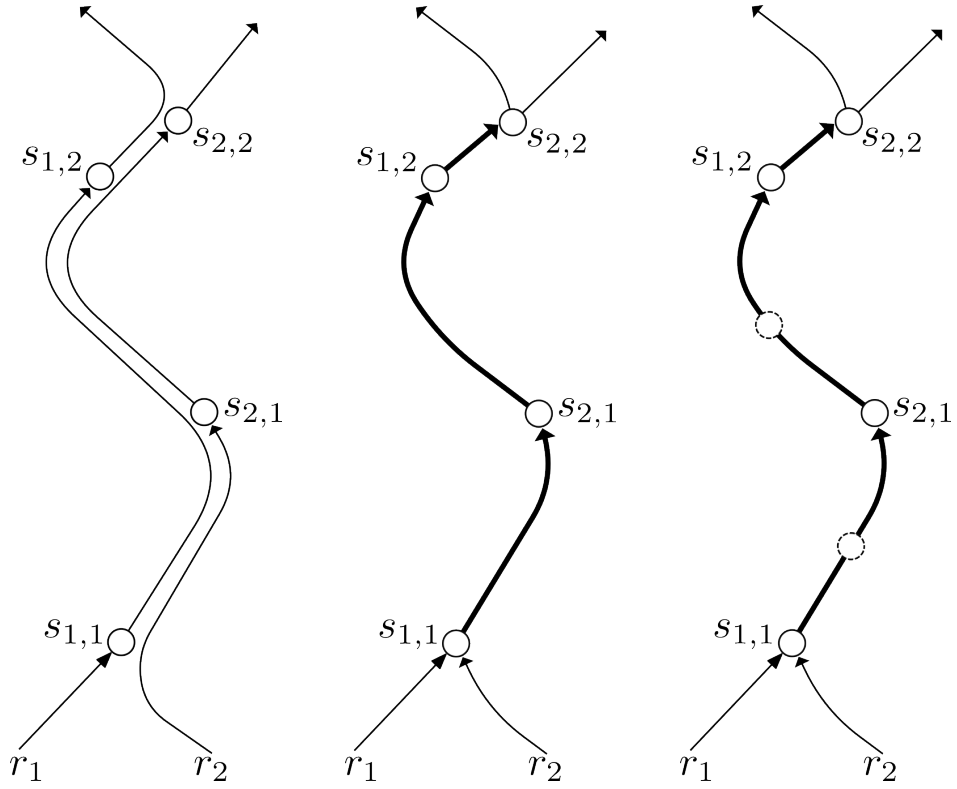
$r = []$
$dlon \leftarrow abs(p_2^{lon} - p_1^{lon})$
$dlat \leftarrow abs(p_2^{lat} - p_1^{lat})$
$step \leftarrow 10^{-m}$
$n \leftarrow \lfloor max(\frac{dlon}{step}, \frac{dlat}{step}) \rfloor - 1$ ▷ number of points to generate
for $i = 1, ..., n$ **do**
    $r \mathrel{+}= (p_1^{lon} + i \cdot step, p_1^{lat} + i \cdot step)$ ▷ add new points to the list
end for

Step 4: *Split too long segments.* We want to split long segments into shorter ones to account for traffic lights and accidents that occur on the smaller parts of long road sections. All long segments are divided into segments with a *preferred* length of 500 meters, which means that those slightly longer than 500 meters (e.g., 505 meters) are kept as-is, but very long ones (e.g., 1000 meters and more) are divided into segments with a *maximum* length of 500 meters. This allows us to prevent the creation of smaller segments while still having most of them about 500 meters long, which roughly corresponds to the average distance between bus stops (572 meters) according to [1].

The schematic illustration of the route graph changes after applying the described steps can be seen on Figure 3.1.

Stops and intersections

Although, as can be seen in Figure 3.1a, one shared segment may contain multiple stops, different stops may have significantly varying dwell times. Moreover, one stop may be used by buses from different routes and, thus, have a varying number of passengers boarding and unboarding, which can also affect the resulting dwell time. As a consequence, in such a case, including dwell times in the predicted travel time can make the prediction less relevant to some subsequent buses. This problem is also addressed in [13], which uses division of routes into dwelling and transit segments, but GPS-only data are used instead of GTFS data as proposed in this thesis. Similar logic applies to intersections, which may have varying travel times depending on the traffic conditions and state of traffic lights. In our solution, shared segments can start and end at stops and intersections



(a) : Before merging shared segments

(b) : Merged shared segment

(c) : Merged and split shared segments

Figure 3.1: Schematic illustration of route graph changes after applying the proposed methods with shared segments bolded; routes r_1 and r_2 have their i-th stops marked as $s_{1,i}$ and $s_{2,i}$, respectively

but cannot have them in between to prevent the described problems.

Usage in travel time prediction

In order to calculate the travel time of a bus along the given edge, we first need to calculate the arrival and departure time in its start and end nodes. Since actual arrival and departure times are not known, we need to estimate them from the information about the bus positions. In the equations below, we assume one specific bus within one trip, with no loss of generality. First, we need to aggregate position updates of the bus into a list of tuples containing timestamps and distances traveled along the route. We can represent this list as $[(t_1, d_1), ..., (t_n, d_n)]$, where n is the index of the last position, and t_i, d_i are the timestamp and the distance traveled for a position with index $i = 1, ..., n$. We assume that $t_{i-1} < t_i$ and $d_{i-1} \leq d_i$ for $i > 1$. Then, to calculate the *estimated arrival time* at a stop we use the following equation:

$$\tilde{s}_m^{arr} = \begin{cases} t_k & \text{if } d_{k-1} = d_k \\ t_{k-1} + (t_k - t_{k-1}) \frac{s_m^{dist} - d_{k-1}}{d_k - d_{k-1}} & \text{otherwise} \end{cases} \tag{3.1}$$

where $\tilde{s}_m^{arr}$ is the estimated arrival time of stop s_m, s_m^{dist} is the distance traveled along the route to reach the stop, and k is the maximum index such that $d_{k-1} \leq s_m^{dist} \leq d_k$. Note that we assume the bus speed is average between two consecutive bus positions.

We can apply this formula not only for stops, but for any node of the route graph. On the other hand, calculating the *estimated departure time* only makes sense for stops, and can be done by assigning it to the timestamp of the closest bus position after the stops within some threshold. Or formally:

$$\tilde{s}_m^{dep} = t_k \tag{3.2}$$

where $\tilde{s}_m^{dep}$ is the estimated departure time of stop s_m, k is the maximum index such that $d_k - s_m^{dist} < \epsilon$, assuming $d_k > s_m^{dist}$, and ϵ is a threshold in kilometers. In our solution, we have this threshold set to 0.04, or 40 meters, which means that the departure time of a stop is defined by the timestamp of the last bus position within 40 meters from the stop.

Knowing the departure and arrival times of two consecutive nodes, we can calculate the *travel time* for their edge by taking the difference between the departure time from the start node and the arrival time to the end node. The calculated travel time can be used as the input for the other methods described in the next chapters.

Also, we can define the *estimated actual delay*:

$$\tilde{s}_m^{del} = \tilde{s}_m^{arr} - s_m^{arr} \tag{3.3}$$

where $\tilde{s}_m^{del}$ is the estimated actual delay at stop s_m, and s_m^{arr} is the scheduled arrival.

Another possibility that opens up to us when we know the arrival and departure times of a stop is that we can calculate the *estimated dwell time* by subtracting the arrival time from the departure time. Defined formally as:

$$\tilde{s}_m^{dwell} = \tilde{s}_m^{dep} - \tilde{s}_m^{arr} \tag{3.4}$$

where $\tilde{s}_m^{dwell}$ is the estimated dwell time of stop s_m, and $\tilde{s}_m^{arr} \leq \tilde{s}_m^{dep}$.

3.3 Kalman filter

Kalman filtering is a method that can be used to predict future states when working with observations that are expected to be noisy, such as GPS measurements or vehicle speed estimates; it was originally introduced in [27]. As we discussed in Chapter 2, it is a very popular algorithm for real-time public transport delay prediction.

In order to estimate the internal state of a process given noisy observations, we must specify the following matrices for each time step k:

- $\mathbf{Q}_k$ is the covariance of the process noise
- $\mathbf{R}_k$ is the covariance of the observation noise

The process model defines the evolution of the true state from $k-1$ to k as following:

$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k \tag{3.5}$$

where:

- $\mathbf{F}_k$ is the state transition matrix applied to the previous state vector $\mathbf{x}_{k-1}$
- $\mathbf{B}_k$ is the control-input model which is applied to the control vector $\mathbf{u}_k$
- $\mathbf{w}_k$ is the process noise that is assumed to be zero-mean Gaussian white noise with the covariance $\mathbf{Q}_k$, i.e., $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$

The process model is paired with the observation model that describes the relationship between the true state $\mathbf{x}_k$ and the observed value at the time step k:

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{3.6}$$

where:

- $\mathbf{H}_k$ is the observation matrix which maps the true state space into the observed space
- $\mathbf{v}_k$ is the observation noise that is assumed to be zero-mean Gaussian white noise with the covariance $\mathbf{R}_k$, i.e., $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$

The goal of the Kalman filter is to provide estimate of $\mathbf{x}_k$ given the initial estimate $\mathbf{x}_0$, the sequence of observations $\mathbf{z}_1, ..., \mathbf{z}_k$ and the matrices defined above. The algorithm is implemented through two sets of equations.

First step: *Prediction.* The current state is predicted based on the previous state:

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{F}_k \mathbf{x}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k \\ \hat{\mathbf{P}}_{k|k-1} &= \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^{\mathsf{T}} + \mathbf{Q}_k \end{aligned} \tag{3.7}$$

where:

- $\hat{\mathbf{x}}_{k|k-1}$ is the predicted state estimate of $\mathbf{x}_k$ given observations up to time $k-1$
- $\hat{\mathbf{P}}_{k|k-1}$ is the predicted estimate covariance matrix given observations up to time $k-1$

Second step: *Update.* The observation $\mathbf{z}_k$ is used to adjust the predicted state using the following set of equations:

$$\begin{aligned} \tilde{\mathbf{y}}_k &= \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \\ \mathbf{S}_k &= \mathbf{H}_k \hat{\mathbf{P}}_{k|k-1} \mathbf{H}_k^{\mathsf{T}} + \mathbf{R}_k \\ \mathbf{K}_k &= \hat{\mathbf{P}}_{k|k-1} \mathbf{H}_k^{\mathsf{T}} \mathbf{S}_k^{-1} \\ \mathbf{x}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \\ \mathbf{P}_{k|k} &= \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \hat{\mathbf{P}}_{k|k-1} \\ \tilde{\mathbf{y}}_{k|k} &= \mathbf{z}_k - \mathbf{H}_k \mathbf{x}_{k|k} \end{aligned} \tag{3.8}$$

where:

- $\tilde{\mathbf{y}}_k$ is the observation pre-fit-residual
- $\mathbf{S}_k$ is the pre-fit residual covariance
- $\mathbf{K}_k$ is the optimal Kalman gain
- $\mathbf{x}_{k|k}$ is the updated state estimate of $\mathbf{x}_k$ given observations up to and including time k
- $\mathbf{P}_{k|k}$ is the updated estimate covariance matrix given observations up to and including time k
- $\tilde{\mathbf{y}}_{k|k}$ is the observation post-fit residual

Proposed model

In the proposed solution, we use the following process model for each edge:

$$x_k = x_{k-1} + w_k, \quad w_k \sim \mathcal{N}(0, \mathbf{Q}_k) \tag{3.9}$$

where x_k is the predicted travel time along the given edge of the route graph. Therefore, we set the state transition matrix $\mathbf{F}_k$ to 1 and the control-input model to $\mathbf{B}_k$ to 0, since our model does not take into account any external influences other than the observed and predicted travel times of the preceding buses.

The observation model is defined by the equation:

$$z_k = x_k + v_k, \quad v_k \sim \mathcal{N}(0, \mathbf{R}_k) \tag{3.10}$$

where z_k represents observed travel time along the given edge of the route graph at time k. Same as with the process model, the observation model does not consider any external influences.

The values of w_k and v_k are discussed in Section 3.4.

Once we know the travel time for all edges to a predicted stop n, we can predict the arrival time $\hat{s}_n^{arr}$ at that stop. We can calculate it as follows:

$$\hat{s}_n^{arr} = t_b + q \cdot \hat{e}_b^{tr} + \sum_{i \in I} \hat{e}_i^{tr} + \sum_{j \in J} \tilde{s}_j^{dwell} \tag{3.11}$$

where:

- t_b is the timestamp of a bus before the predicted stop
- q is the percentage of distance the bus traveled along the current edge
- $\hat{e}_b^{tr}$ is the predicted travel time for the current edge
- I is a set of edge indices from the current position of the bus to the predicted stop, $b \notin I$
- J is a set of stop indices from the current position of the bus to the predicted stop, $n \notin J$
- $\hat{e}_i^{tr}$ is the predicted travel time along the i-th edge
- $\tilde{s}_j^{dwell}$ is the estimated dwell time for the j-th stop, calculated using Equation (3.4)

Predicted delay is calculated in the following way:

$$\hat{s}_n^{del} = \hat{s}_n^{arr} - s_n^{arr} \tag{3.12}$$

where s_n^{arr} is the scheduled arrival time of the predicted stop, and $\hat{s}_n^{arr}$ is calculated using Equation (3.1).

Fallback solution

The model we proposed assumes that the travel times of preceding buses are present. However, this is not always the case, and the most trivial example is the first bus of the day. Also, it is possible that only some edges have predicted travel times present. When no information is available, we assume that the bus travels in the *scheduled travel time*. To calculate the scheduled travel time along an edge e_i with the last stop before the edge being s_{prev} and the first stop after the edge s_{next}, we use the following formula:

$$e_i^{tr} = (s_{next}^{arr} - s_{prev}^{dep}) \frac{e_i^{len}}{s_{next}^{dist} - s_{prev}^{dist}} \tag{3.13}$$

where:

- e_i^{sch} is the calculated scheduled travel time of e_i
- s_{next}^{arr} is the scheduled arrival time of stop s_{next}
- s_{prev}^{dep} is the scheduled departure time of stop s_{prev}
- e_i^{len} is the length of e_i
- s_{next}^{dist} is the distance from the start of the route to s_{next}
- s_{prev}^{dist} is the distance from the start of the route to s_{prev}

We assume that the scheduled travel times of edges between two stops are equally distributed.

This solution is very similar to the one used by Golemio. When we set our algorithm to always use it instead of the Kalman filter, the result is equal to $\pm 2\%$ of Golemio results, which also has some logic for handling dwell times, hence the difference.

3.4 Adaptive Kalman filter

The Kalman filter performs best when the process behaves according to the defined process model. On the other hand, when the process might

be suddenly affected by external influences, such as traffic accidents, it can take some time before the filter catches up with these changes. Moreover, after the filter catches up, the mentioned external influences might no longer affect the process, and the problem repeats. In the proposed solution, we try to make the filter adapt to sudden changes when it detects dynamics that the process model cannot account for.

Varying noise

When predicting travel times for edges of different lengths, we do not use a fixed number to define the process and observation noise. Longer edges are assumed to have longer travel times. Rather than trying to find the ideal noise values, we use a percentage of scheduled travel time and provide it to the Kalman filter as expected noise. In other words, we multiply the scheduled travel time as defined in Equation (3.13) by a multiplier.

As a side note, we can define the process noise as confidence in our model, and the observation noise as a measure of the accuracy of sensors (e.g., speedometer or GPS). Therefore, not only the absolute values of these parameters are important, but also their ratio (i.e., how much more or less we trust our model than the sensors). The automatic setup of noise is also discussed in Section 3.6.

Default multipliers are set to 0.05 for the process noise (i.e., 5% of the scheduled travel time) and 0.25 for the observation noise, which means that we do not want to instantly assume that the travel time of the last preceding bus is the absolute truth. Instead, we are being slightly suspicious of it and waiting for the situation to develop with further buses arriving.

Fading memory factor

Fading memory filters are not normally classified as adaptive since they do not adapt to the input, but they do provide good performance with suddenly changing processes [28].

According to [29], we can make the filter gradually forget older measurements by adding a multiplying term α to the calculation of $\hat{\mathbf{P}}_{k|k-1}$ in Equation (3.7):

$$\hat{\mathbf{P}}_{k|k-1} = \alpha^2 \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^{\mathsf{T}} + \mathbf{Q}_k \tag{3.14}$$

where $\alpha > 1$. Otherwise, when α is equal to 1, we get a regular Kalman filter.

Initially, we set α to 1.05, since [28] recommends having it close to 1 before changing to a higher value. But the value of this parameter and how it affects the filter performance is a subject to discussion in Chapter 5.

3.5 Kalman smoother

The Kalman smoother is a backward algorithm that is run after the Kalman filter, and allows to refine estimates of previous states to produce "smoothed" estimates of state variables. We define it according to [30].

We calculate the *smoothed* estimates for $\hat{\mathbf{x}}_{k|n}$ and covariances $\mathbf{P}_{k|n}$, with $k < n$, using observations $\mathbf{z}_1, ..., \mathbf{z}_n$. The filtered predicted and updated state estimates $\hat{\mathbf{x}}_{k|k-1}$, $\hat{\mathbf{x}}_{k|k}$, and covariances $\mathbf{P}_{k|k-1}$, $\mathbf{P}_{k|k}$ are saved for use in the backward pass, where we compute $\hat{\mathbf{x}}_{k|n}$ and $\mathbf{P}_{k|n}$. Starting from the last time step, we proceed backward in time using the following recursive equations:

$$\begin{aligned} \mathbf{C}_k &= \mathbf{P}_{k|k}\mathbf{F}_{k+1}^{\mathsf{T}}\mathbf{P}_{k+1|k}^{-1} \\ \hat{\mathbf{x}}_{k|n} &= \hat{\mathbf{x}}_{k|k} + \mathbf{C}_k\left(\hat{\mathbf{x}}_{k+1|n} - \hat{\mathbf{x}}_{k+1|k}\right) \\ \mathbf{P}_{k|n} &= \mathbf{P}_{k|k} + \mathbf{C}_k\left(\mathbf{P}_{k+1|n} - \mathbf{P}_{k+1|k}\right)\mathbf{C}_k^{\mathsf{T}} \end{aligned} \tag{3.15}$$

Note that the smoother is not used directly for prediction, as it is more of a post-processing step.

We also define *lag one covariance* as a recursion for $j = k, k-1, ..., 2$ [31]:

$$\mathbf{P}_{j-1,j-2|k} = \mathbf{P}_{j-1|j-1}\mathbf{C}_{j-2}^{T} + \mathbf{C}_{j-1}\left(\mathbf{P}_{j,j-1|k} - \mathbf{F}_k\mathbf{P}_{j,j-1|j-1}\right)\mathbf{C}_{j-2}^{T} \tag{3.16}$$

with initial value $\mathbf{P}_{k,k-1|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\mathbf{F}_k\mathbf{P}_{k-1|k-1}$. The lag one covariance is used in Section 3.6.

Generation of missing data

We can use the filter and the smoother to generate missing data (e.g., missing bus positions). When calculating the updated state estimate $\mathbf{x}_{k|k}$ and covariance $\mathbf{P}_{k|k}$, we need to know the value of observation $\tilde{\mathbf{y}}_k$, but even in the case it is not available, we can still use the transition equation. Therefore, when $\tilde{\mathbf{y}}_k$ is missing during the update step, we modify Equation (3.8) in the following way:

$$\begin{aligned} \mathbf{x}_{k|k} &= \hat{\mathbf{x}}_{k|k-1}K \\ \mathbf{P}_{k|k} &= \hat{\mathbf{P}}_{k|k-1} \end{aligned}$$

After that, once the Kalman filter and the smoother have been applied for all observations, we have smoothed estimates of state $\mathbf{x}_i$ and covariances $\mathbf{P}_i$, for $i = 1, ..., k$. The values for all time steps, including the generated ones, can be determined as follows:

$$\mathbf{z}_i = \mathbf{H}_i \mathbf{x}_i$$

Proposed model

Since the rate at which bus positions are updated is not fixed and can vary from several seconds to several minutes, we use the Kalman smoother to generate data according to the following process model:

$$\begin{pmatrix} \hat{d}_k \\ \hat{v_k} \end{pmatrix} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} d_{k-1} \\ v_{k-1} \end{pmatrix} + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k) \tag{3.17}$$

where $\hat{d}_k$ is the predicted distance traveled by bus along the route at time k, and $\hat{v_k}$ is its velocity. In the transition matrix, $\Delta t = t_k - t_{k-1}$ is the interval of update.

The observation model is defined as:

$$\begin{pmatrix} \tilde{d}_k \\ \tilde{v_k} \end{pmatrix} = \begin{pmatrix} d_k \\ v_k \end{pmatrix} + \mathbf{v}_k, \quad \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k) \tag{3.18}$$

where $\tilde{d}_k$ is the observed distance traveled by bus along the route at time k, and $\tilde{v_k}$ is its velocity.

3.6 Automatic parameter estimation

Because our proposed algorithm can be used for multiple buses, it can be difficult to find suitable parameters that work for each route. For example, process noise ($\mathbf{Q}_k$) and observation noise ($\mathbf{R}_k$) may be different for buses from different routes, and these parameters are known to be hard to estimate correctly [28]. We can use the Expectation-Maximization (EM) algorithm to automatically estimate these parameters. In simple words, the idea of this algorithm is that it tries to estimate the most likely parameters for a given set of observations.

In the case of the Kalman filter, it consists of two steps applied iteratively [31, 32]:

1. Calculate $\hat{\mathbf{x}}_{k|n}$ and $\mathbf{P}_{k|n}$ using Equation (3.15).

2. Estimate process and observation noise using the following equations:

$$\mathbf{Q}_k^{(r+1)} = \frac{1}{k}\left(\mathbf{W} - \mathbf{U}\mathbf{V}^{-1}\mathbf{U}\right)$$

$$\mathbf{R}_k^{(r+1)} = \frac{1}{k}\sum_{i=1}^{k}\left[\left(\hat{\mathbf{y}}_{i|i} - \mathbf{C}\hat{\mathbf{x}}_{i|n}\right)\left(\hat{\mathbf{y}}_{i|i} - \mathbf{C}\hat{\mathbf{x}}_{i|n}\right)^T + \mathbf{C}\mathbf{P}_{i|n}\mathbf{C}^T\right]$$

$$\mathbf{U} = \frac{1}{k}\sum_{i=1}^{k}\left[\mathbf{P}_{i-1|k} + \hat{\mathbf{x}}_{i-1|k}\hat{\mathbf{x}}_{i-1|k}^T\right]$$

$$\mathbf{V} = \frac{1}{k}\sum_{i=1}^{k}\left[\mathbf{P}_{i,i-1|k} + \hat{\mathbf{x}}_{i|k}\hat{\mathbf{x}}_{i-1|k}^T\right]$$

$$\mathbf{W} = \frac{1}{k}\sum_{i=1}^{k}\left[\mathbf{P}_{i|k} + \hat{\mathbf{x}}_{i|k}\hat{\mathbf{x}}_{i|k}^T\right]$$

Continue for a specified r number of operations or until the difference between the calculated parameters is smaller than some defined threshold.

In our solution, we set r to 5 because we do not want the algorithm to run for too long. In addition, although we use the EM algorithm only for noise estimation, it can also be used to estimate other parameters.

3.7 Prediction parameters

In this section, we describe parameters that can be changed by the end-user without having to understand the algorithm.

Past data time window

In order to not consider data about preceding buses from several hours ago for the currently predicted bus, we introduce a concept named *past data time window*. It controls how old data can be before it gets ignored. In the case of travel time, the timestamp determining its age is equal to the time of arrival at the starting node of the edge for which the travel time is calculated.

Setting this parameter to 0 means that no data is passed to the Kalman filter, and it always reverts to the fallback solution.

Number of predicted stops

We are not limited to predicting delay only for the next stop from the current bus position, because Equation (3.11) can be used to calculate

the travel time to any future stop from the current bus position. Even if the number of predicted stops is set 1, our algorithm can make use of information from shared segments and react to possible changes in traffic.

Chapter 4

Data collection and analysis

Raw static GTFS data for Prague can be obtained from the open data page managed by Prague Integrated Transport (PID, from its Czech acronym) [33]. The provided ZIP archive can be freely downloaded by anyone and contains data for the next 14 days with daily updates. The archive contains files in CSV format according to the GTFS specification. Also, the page contains a brief description and clarification on the specifics of the data provided.

Real-time GTFS data are provided by the Golemio API developed by Operator ICT, and can be accessed after generating an API key [5]. Using the API, it is possible to get information about bus positions, schedules, stops, routes, and other information relevant to the currently active trips. Although the API can also be used to retrieve static GTFS data instead of using the archive from PID, it has a maximum limit of 10,000 objects returned per request. Initially, we tried to use only the API because of its advantage of returning all data in JSON format [34], but that idea was discarded after realizing that some records, such as stop times, are stored in millions of instances. The API documentation containing information about the returned JSON objects and their properties can be found in [5].

As of November 2022, there is no easy way to access historical data on bus positions in Prague. When contacting Operator ICT, we were informed that other than the data for the next 14 days, no historical information is stored. Our proposed solution does not use any historical data beyond what was collected on the day of prediction.

4.1 Aggregation of datasets

Aggregation of static data, such as routes, stops, or schedules, can be easily done by downloading the archive from the PID page. Since it is valid for two weeks, we can download it just once at the start of the

prediction day without having to update it periodically during the day.

On the other hand, the real-time positions of buses are mostly valid for the duration of their active trip and are removed soon afterwards. We have implemented a program to get bus positions from the API at a given interval and for a specified amount of time. For example, to collect a dataset, we can fetch data every 15 seconds from 00:01 to 23:59. After that, the program saves the data into JSON files, which can be used for analysis or testing different solutions.

Below is an example of a response with a bus position from the Golemio API.

```
"last_position": {
  "bearing": 34,
  "delay": {
    "actual": 213,
    "last_stop_arrival": 23,
    "last_stop_departure": -84
  },
  "last_stop": {
    "id": "331",
    "sequence": 11,
    "arrival_time": "2019-03-18T11:00:00.000Z",
    "departure_time": "2019-03-18T11:04:00.000Z"
  },
  "next_stop": {
    "id": "334",
    "sequence": "12",
    "arrival_time": "2019-03-18T11:10:00.000Z",
    "departure_time": "2019-03-18T11:14:00.000Z"
  },
  "is_canceled": false,
  "origin_timestamp": "2019-03-18T11:00:00.000Z",
  "speed": 34,
  "shape_dist_traveled": "12.3",
  "tracking": false
}
```

As mentioned earlier, *gtfs_shape_dist_traveled* is one of the most important parameters for us. It specifies the actual distance traveled along the bus route from the first stop to the position specified in the record. This allows us to make predictions based on distances rather than GPS coordinates, although coordinates are still used in our solution to create the route graph.

The following datasets were created using the process described above:

- *Friday, May 27, 2022 (all routes).* Contains bus positions and other related data from 03:00 to 23:59 of the specified day. This dataset is interesting because there were numerous API outages during its creation, and some of the data is missing. It allows us to test prediction methods on incomplete data, which can also happen during real-time operation. Also, Friday might be an interesting day in terms of prediction because many people try to get home earlier, which can create peak hours at a different time from the next dataset.
- *Monday, December 5, 2022.* Collected from 02:00 to 23:59 without any significant outages. In contrast with Friday, Mondays can be busier in the morning when many people are trying to get to work.
- *Friday, May 27, 2022 (most used routes).* According to the Prague Transportation Yearbooks for years 2018-2020, the busiest segment of bus routes in the city is Nemocnice Krč-U Labutě (67,020 passengers carried from 06:00 to 20:00), which is also near the busiest stop Kačerov (49 860 passengers from 06:00 to 20:00) [1, 35, 36]. We extracted bus trips passing the busiest points from the first dataset. Due to its relatively small size (about 150 Megabytes) compared to the other two datasets (about 2 Gigabytes), this dataset can be found in the attachments of the thesis.

4.2 Analysis

After collecting the datasets, we can investigate their properties to gain more insights for interpreting the tested predictors' results.

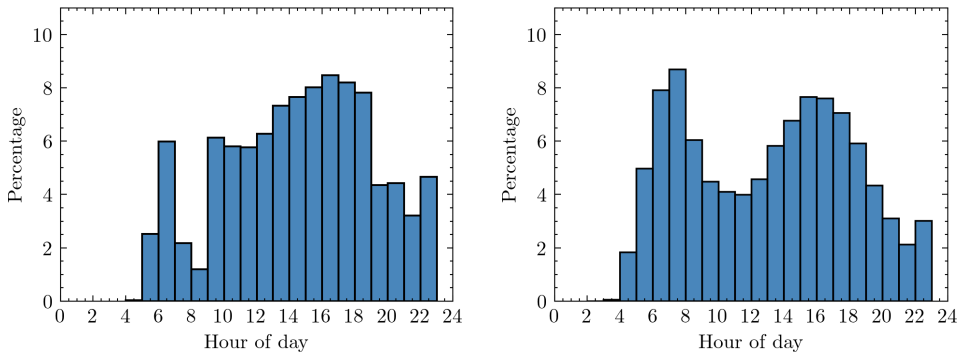
Table 4.1 provides a brief overview of the collected datasets. We see that a relatively small amount of routes services tens of thousands of trips, which is consistent with our idea of reusing information about buses passing the same road segment to make predictions.

Table 4.1: Dataset statistics

	May 27, 2022 (all routes)	December 5, 2022 (all routes)	May 27, 2022 (most used routes)
Routes	800	799	19
Shapes	6,533	6,574	115
Stops	16,420	17,258	551
Stop times	1,505,054	1,414,618	76,495
Trips	75,761	68,069	3,802
Bus positions	1,160,878	1,406,420	266,300

Figure 4.1 combines timestamps when bus positions were collected into a histogram, and we can see that there are indeed gaps in the May

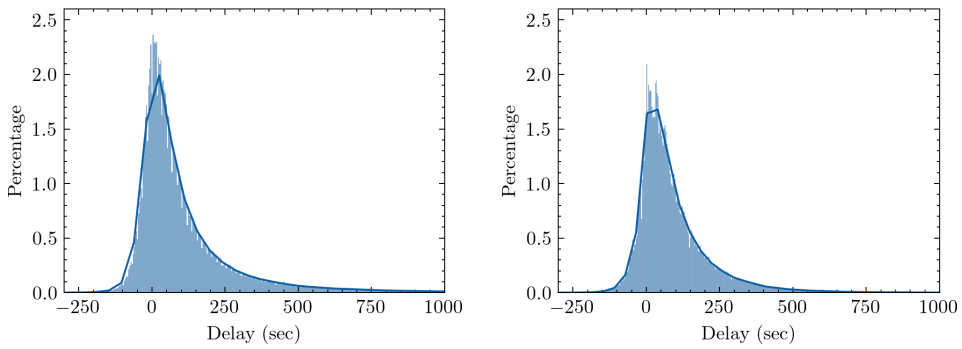
dataset during the morning hours and around 19:00. Although we do not see the state in the morning, it seems that there is no such steep fall around noon when comparing to the second graph. The histogram for December also supports our assumption that there is a stronger demand on Monday morning, which results in more buses traveling around the city.



(a) : Dataset for May 27, 2022 **(b) :** Dataset for December 5, 2022

Figure 4.1: Distribution of bus position timestamps

While on first sight from Figure 4.1 it may seem that a significant chunk of delays lies near zero, it should be noted that for both datasets, around 50% of delays exceed 1 minute, 30% exceed 2 minutes, and 11% exceed 5 minutes. Also, even though we cannot directly map the result to Prague, a study conducted in Columbia shows that only one-third (33%) of people are willing to wait up to 5 minutes beyond the scheduled arrival time of buses [37]. Recall that actual delays are estimated using Equation (3.3).

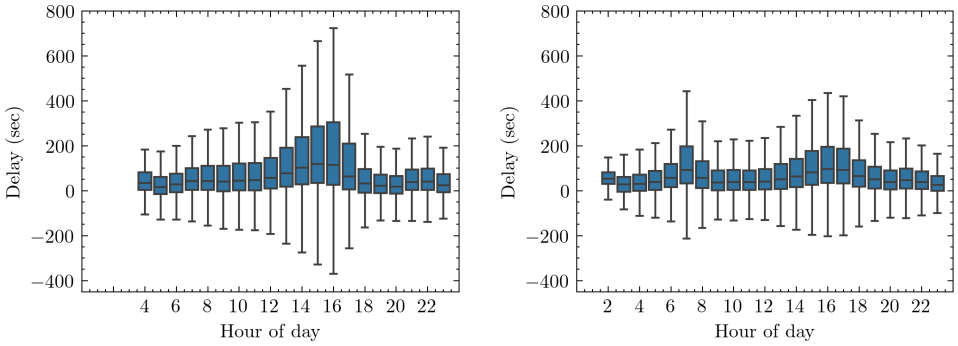


(a) : Dataset for May 27, 2022 **(b) :** Dataset for December 5, 2022

Figure 4.2: Distribution of actual delays

Figure 4.3 provides a valuable, albeit predictable, insight into the distribution of delays by time. Along with the information from Figure 4.1, we can say that the increase in the number of buses on the road might correlate with an overall increase in the number of other vehicles and

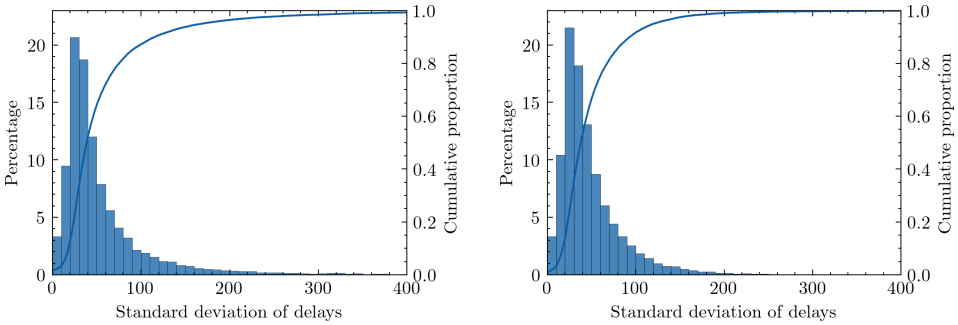
people on the road in general, which negatively affects travel times. We clearly see the morning and evening rush hours and can expect real-time predictors to perform worse during these times.



(a) : Dataset for May 27, 2022 **(b) :** Dataset for December 5, 2022

Figure 4.3: Actual delays for different hours of day

Although absolute values of delays are very important, Figure 4.4 gives us information about how much the delays vary during the same trip. For example, a bus that departed from the depot 10 minutes later but otherwise ran "on schedule" has big absolute delays at every stop, but if we compare all delays between each other, they will not differ much. Such delays are very easy to predict for simple algorithms, which do not have a lot of information about the surrounding state of the bus and try to act safely by not changing their predictions too often. In contrast, such algorithms will perform worse when a bus has widely varying delays at each stop (and thus, a bigger standard deviation of delays). The cumulative distribution curve in the figure shows that the May dataset has more trips with larger standard deviations of delays.

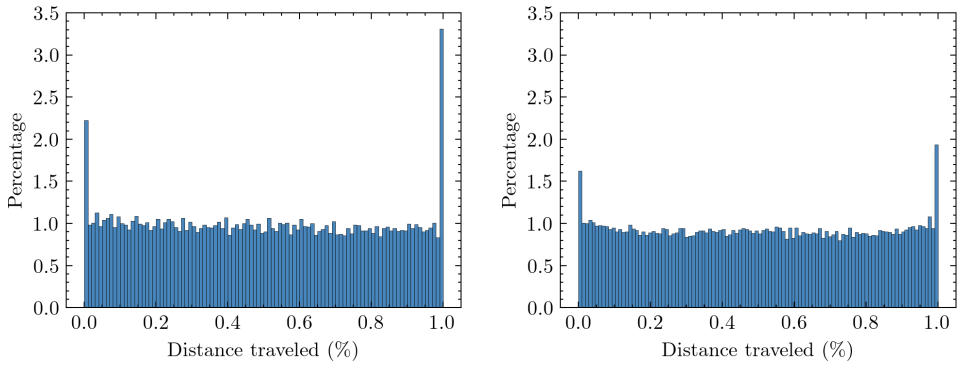


(a) : Dataset for May 27, 2022 **(b) :** Dataset for December 5, 2022

Figure 4.4: Distribution of standard deviations of actual delays calculated for every trip

Figure 4.5 show that buses are distributed over the routes relatively

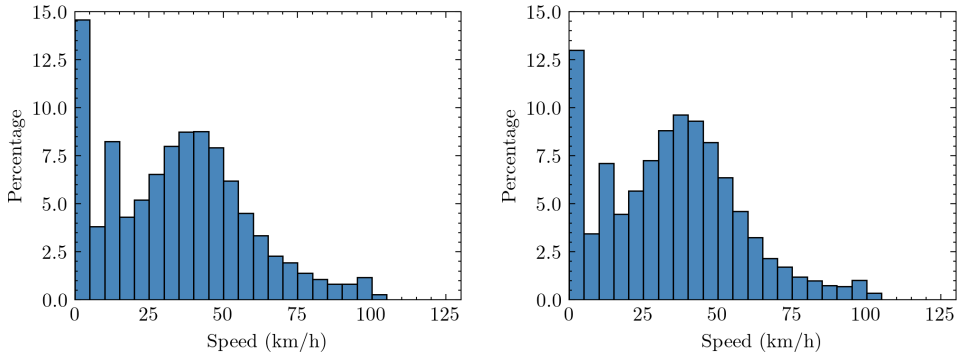
uniformly. The sudden jumps at the start and end of routes likely represent buses that have finished their trips and are waiting at the depot.



(a) : Dataset for May 27, 2022 **(b) :** Dataset for December 5, 2022

Figure 4.5: Distribution of the percentage of distance traveled along the route for all bus positions

In Figure 4.6 we see that bus speeds are distributed very similarly for both datasets. Zero-speed buses are most probably staying at stops or depots.



(a) : Dataset for May 27, 2022 **(b) :** Dataset for December 5, 2022

Figure 4.6: Distribution of bus speeds

4.3 Handling of invalid data

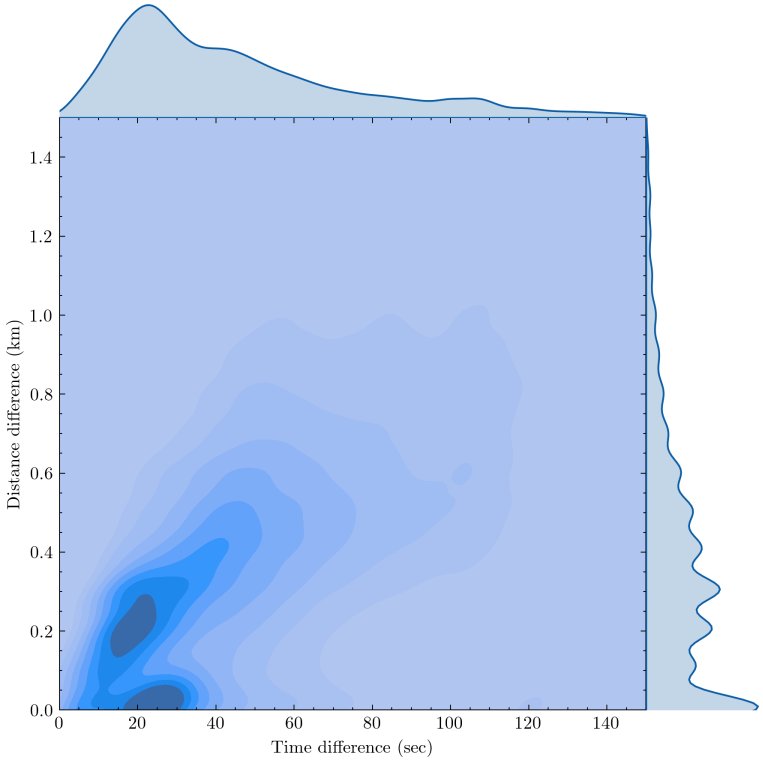
Data collected from the described sources might contain errors that may influence the correctness of prediction algorithms by invalidating some assumptions. For example, Equation (3.1) used to calculate estimated arrival at a stop cannot be computed if the subsequent bus position has a smaller distance traveled than the previous one.

Before providing data to prediction algorithms, we do the following:

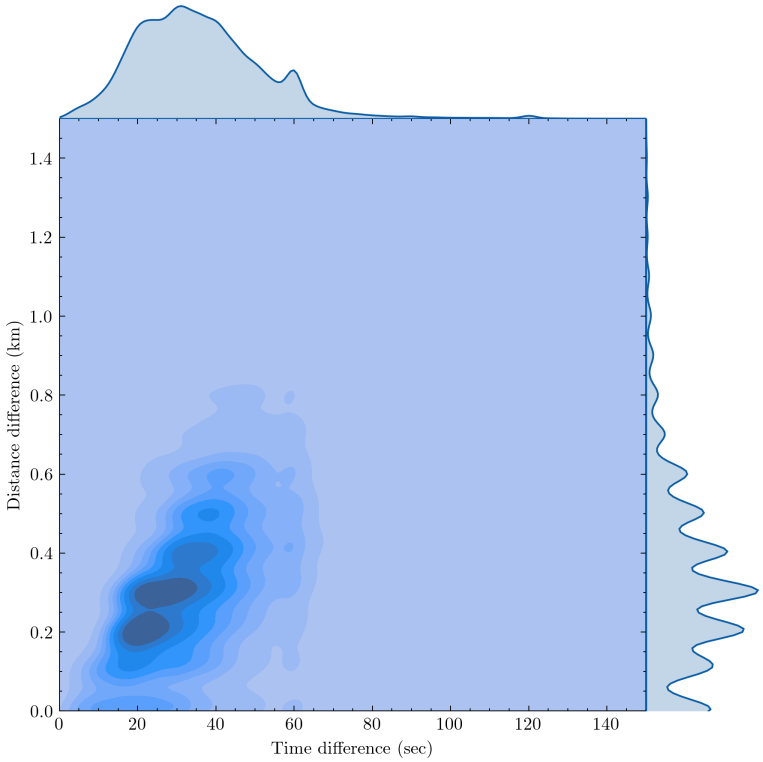
- Duplicated bus positions are filtered out.
- Sometimes, positions are provided so that a bus appears to go backwards (the previous position has a lower distance but higher time). These positions are considered to be invalid and are removed.
- Some stops can also be considered invalid when we have two different stops assigned to the same route having the same distance traveled assigned to them. Positions at such stops are also removed.

There are less than 2% of all positions that we can consider invalid in both datasets. Therefore, we can simply ignore them without affecting the performance of the proposed algorithms too much.

One of the most critical parts of our solutions is the estimation of arrival times at stops and nodes of the route graph as described in Section 3.2. In short, we are trying to calculate arrival times by looking at the two closest bus positions before and after the estimated point. There are two problems that we are trying to solve: the bus positions closest to the calculated point may actually be quite a long distance apart, or too much time may have passed between them. Figure 4.7 visualizes how far apart (both time- and distance-wise) are every two consecutive bus positions. According to the graphs, most bus position updates are sent out quite often, and we can limit the maximum allowed distance without losing too much information. We calculated that for both datasets, around 5% of all pairs of consecutive positions are more than both 2 minutes and 1 kilometer apart. Also, in about 1% of all pairs, only one of the limits is exceeded. As a result, in our solution, we ignore positions that are more distant than 2 minutes or 1 kilometer apart.



(a) : Dataset for May 27, 2022



(b) : Dataset for December 5, 2022

Figure 4.7: Relationship between the time difference and the distance difference of two consecutive bus positions for the same trip

Chapter 5

Evaluation

In this chapter, we discuss the outputs of the proposed method given the datasets introduced in the previous chapter and compare it to the baseline solution.

First, we present the results after applying the proposed route graph creation algorithm. After that, we describe the process of simulating real-time updates using the collected datasets, and introduce error metrics used to quantify the performance of prediction models. Then we discuss the impact of different parameter setups on the implemented prediction model. And finally, our solution is compared to the baseline solution provided by Golemio, which predicts delays by simply calculating the difference between the timestamp of the bus position at a given distance traveled and the scheduled time for that distance. Data from Golemio are officially used by Prague Integrated Transport to display positions of public transport and delays [7].

For the evaluation and implementation of the core parts of our solution, we used Python 3 [38], simdkalman [39], FilterPy [40], and NetworkX [41].

5.1 Route graph

Figure 5.1 showcases the edges of the created route graph superimposed on the map of Prague. The dataset is for December 5, 2022, and all the proposed optimization methods are applied. Every two edges lying next to each other have different colors and represent route segments with a preferred length of 500 meters.

Graphs in Figure 5.2 show segment length distributions before and after applying the proposed optimization steps. Initially, the route graph containing nodes with rounded GPS coordinates has many short edges. After combining them into the longest possible segments, we can see a

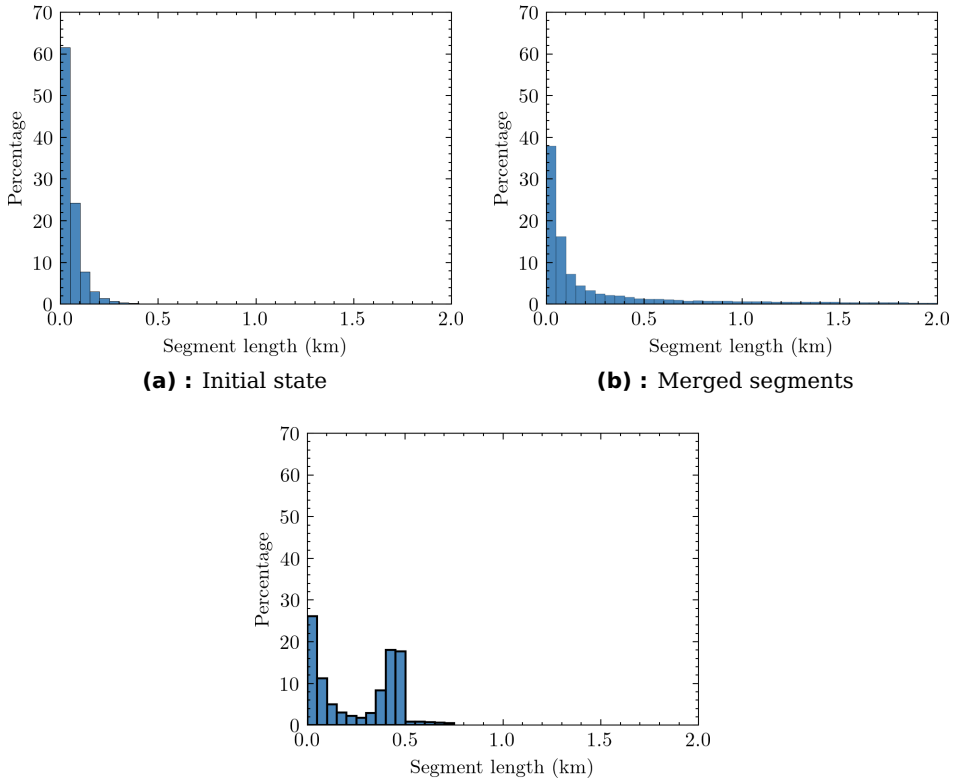
Figure 5.1: Route graph created using the proposed methods superimposed on the map of Prague

significant decrease in the number of edge lengths lying close to zero, with more values distributed in the right part of the graph shown in Figure 5.2b. When the long edges are broken up into smaller segments, a significant portion of all edges begins to be about the preferred 500 meters long. Note that edges slightly longer than the preferred length are kept as-is, as described in Section 3.2, to prevent the creation of more small edges.

Table 5.1 presents statistics about the state of the route graph before and after optimizing it. Overall we can clearly see that the proposed algorithm greatly reduces the graph size in terms of the number of nodes and edges while keeping other characteristics the same.

Table 5.1: Route graph state before and after applying optimizations

	Nodes	Edges	Shared segments	Length (km)	Stops
Before	176,085	321,114	280,639	110,337	12,578
After	40,716	72,215	63,196	110,337	12,578



(a) : Initial state

(b) : Merged segments

(c) : Merged and split segments

Figure 5.2: Distribution of route graph segment lengths before and after optimizations

5.2 Simulation of real-time process

The real-time process of getting data from the API and processing it on the fly is simulated using the collected datasets in the following way:

1. Bus positions are sorted by their timestamps.
2. The sorted list of positions is split into chunks in such a way, that the last item's timestamp in each chunk is at most 15 seconds after the timestamp of the first item in the same chunk.
3. The created chunks are provided to the predictors one by one.

5.3 Performance metrics

Prediction results are evaluated in terms of the performances of two measures, namely, the mean absolute error (MAE) and the root mean square

error (RMSE). The RMSE gives a relatively high weight to large errors; therefore, it is more useful when outliers are particularly undesirable. The MAE is a linear score, which means that all the individual differences are weighted equally in the average; thus, it will give less weight to outliers. These measures are calculated as follows:

$$\begin{aligned} MAE &= \frac{1}{n}\sum_{i=1}^{n}\left|\tilde{s}_i^{del} - \hat{s}_i^{del}\right|, \\ RMSE &= \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\tilde{s}_i^{del} - \hat{s}_i^{del}\right)^2}, \end{aligned} \tag{5.1}$$

where $\tilde{s}_i^{del}$ represents the estimated actual delay at stop s_i, and $\hat{s}_i^{del}$ represents the predicted delay at the same stop, and n is the number of predictions.

5.4 Fine-tuning of parameters

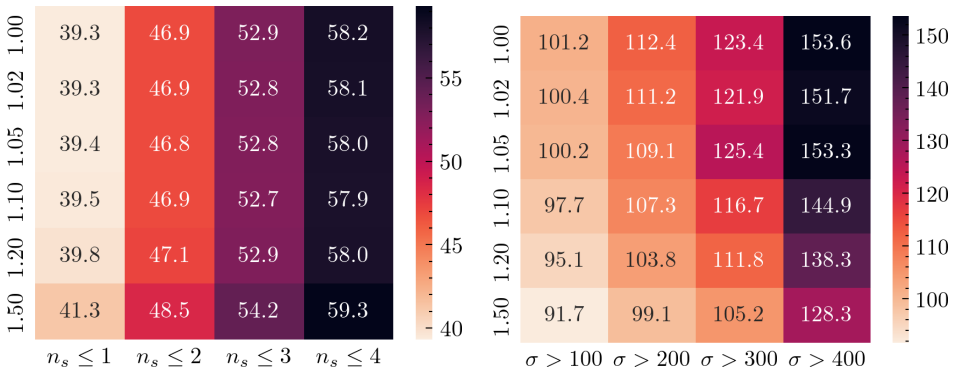
We want to know how different parameters introduced in Chapter 3 affect the performance of our solution. We do so by running simulations for different parameter values and comparing the resulting RMSE. Every simulation is also parameterized by the maximum number of stops to be predicted (n_s) and the minimum standard deviation of delays (σ_{del}) in the tested trips. The simulations are done on the smaller dataset from May 27, 2022, containing the trips passing along the most used road segment, as described in Section 4.1. This allows us to test parameters more efficiently because a complete simulation cycle for all trips of a day can take several hours, and testing all parameters can take days or even weeks. In addition, it gives readers the opportunity to run simulations themselves with the same dataset, as it is attached to this thesis.

Fading memory factor

In Figure 5.3a it can be seen that increasing value of α (fading memory factory of the Kalman filter) affects the prediction performance rather negatively, although the differences are very insignificant.

The situation changes completely in Figure 5.3b, where increasing α clearly reduces the error. It makes sense, because with delays widely varying during the day, recent information about travel times of preceding buses is expected to provide more value than older information.

We choose 1.10 as the value of α, since it performs well enough in both cases. Larger values may perform worse for trips with low σ_{del}.

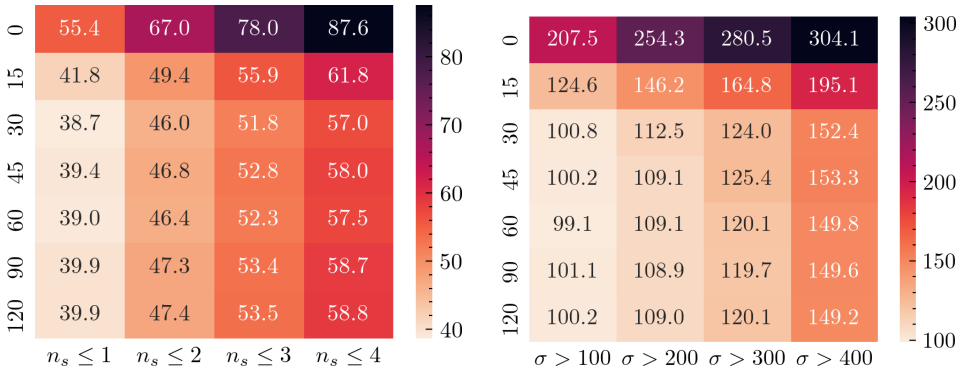


(a) : Limited number of predicted stops **(b) :** Minimum σ of delays

Figure 5.3: RMSE for different values of α

Past time window

The past time window has a very good effect on the performance of prediction, as shown in Figure 5.4. Setting it to 0 basically means disabling all features of the proposed solution and switching to the fallback method. In our simulation, the value of 60 minutes performs best in both test cases, and we use this value in the final evaluation. While increasing the past time window value may be tempting in a real-time setting, because we want to collect as much information about the state of the transit situation as possible, it increases the error when older information starts to be less relevant (e.g., information about traffic congestion from 2 hours ago is not relevant for the current transit situation).

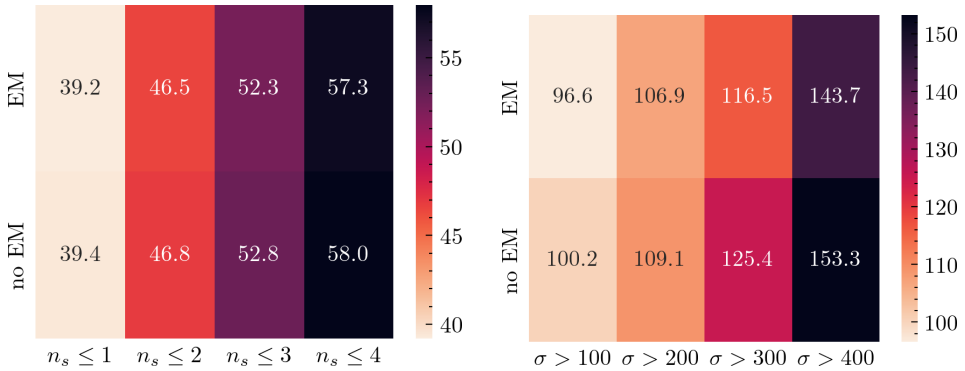


(a) : Limited number of predicted stops **(b) :** Minimum σ of delays

Figure 5.4: RMSE for different settings of the past time window

Expectation-Maximization

Although using the EM algorithm reduces prediction error, albeit not by much, its disadvantage is a very long running time. Enabling EM causes Kalman smoothing to be used multiple times for each prediction, and the simulation program takes 3-5 times longer to run than when EM is disabled, which is not feasible for real-time prediction. Therefore, we do not use EM in the final solution. Figure 5.5 shows the results of the simulation with the number of EM iterations set to 5.



(a) : Limited number of predicted stops **(b) :** Minimum σ of delays

Figure 5.5: RMSE with EM enabled and disabled

5.5 Results

In this section, we show how our proposed solution compares to the baseline solution provided by Golemio. According to the tests in the previous section and the discussions in Chapter 3, we use parameter values as shown in Table 5.2 for the final simulation. Simulations are divided into three groups: by the number of predicted stops, the hour of day, and the minimum standard deviation of delays in tested trips. The datasets used are described in Chapter 4.

Table 5.2: Parameters used in the final simulation

Parameter	Value
Max. number of predicted stops	6
Past time window	60
Fading memory factor (α)	1.10
EM algorithm	No
Observation noise multiplier	0.25
Process noise multiplier	0.05

Number of predicted stops

Figure 5.6 shows a simple but effective comparison of the tested prediction models. With an increasing number of predicted stops, our model shows more and more significant error reduction. The biggest improvement can be seen in the May 27, 2022 dataset for trips traveling along the busiest bus segment in Prague. Table 5.3 shows average error reduction for every dataset.

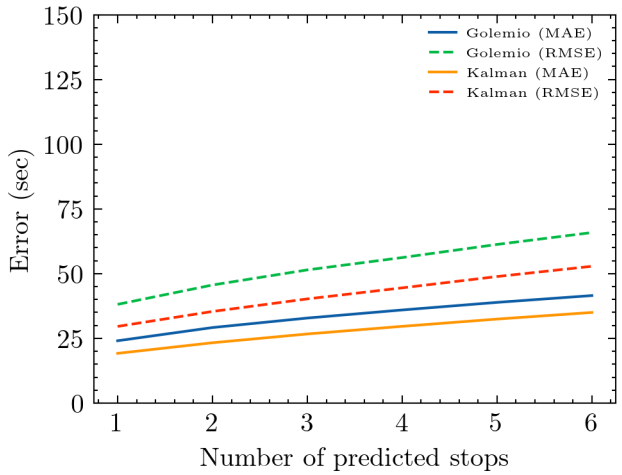
Table 5.3: Number of predicted stops - average error reduction in comparison with the baseline solution

Dataset	MAE	RMSE
December 5, 2022 (all routes)	18.1%	23.2%
May 27, 2022 (all routes)	12.8%	12.7%
May 27, 2022 (most used segment)	21.5%	33.4%
Mean	17.5%	23.1%

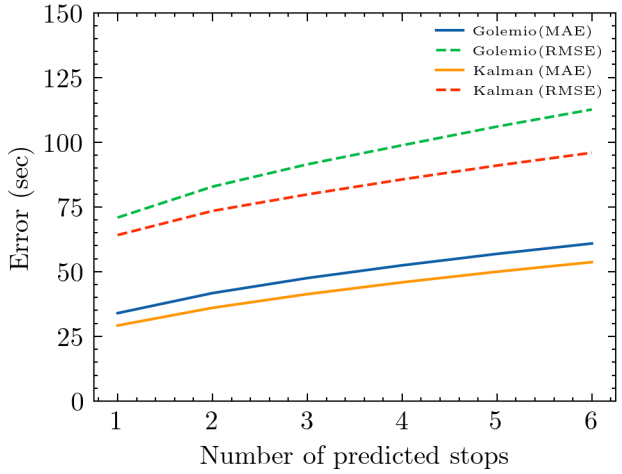
Hour of day

Figure 5.7 shows a greater and less consistent difference between the errors of the predictors. We can clearly see that both prediction models are less accurate during peak hours, but the jump in error of our model is much less drastic than that of the baseline model. From the sharp change in RMSE of the baseline solution, we can assume that it makes more significant errors during the prediction process in rush hours. While the error reduction in the previous simulation was noticeable, the current simulation demonstrates the main advantage of the proposed solution – the ability to adapt to sudden changes in the current transit situation.

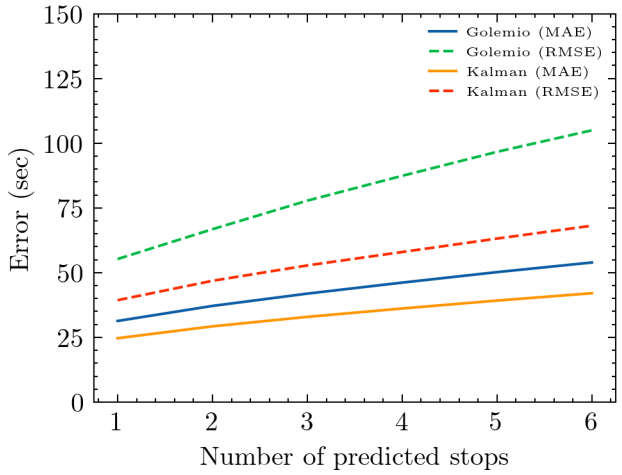
From Table 5.4, we see that for the May 27, 2022 dataset (for the most used segment), the difference in error during peak hours was big enough to shift the mean value of RMSE sufficiently to take first place for the highest error reduction among all simulations. Worse performance of the other dataset for the same day shown in Figure 5.7b can be explained by missing data about bus positions for some hours of the day as discussed in Section 4.2. In addition, the initial high error in the morning hours may be the result of only a few night buses having significant delays and not driving often enough for our model to capture the travel times of the preceding buses.



(a) : Dataset for December 5, 2022 (all routes)

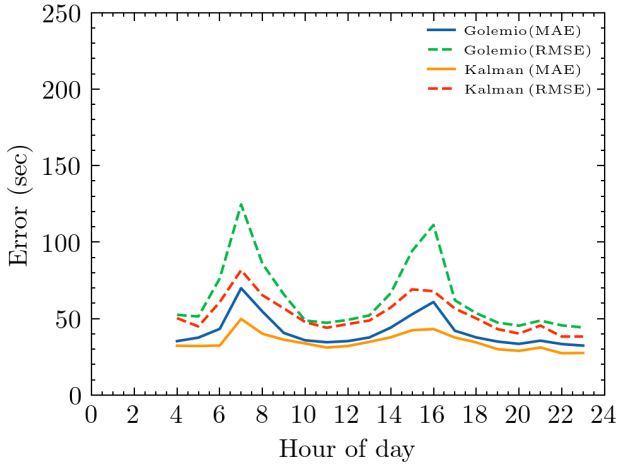


(b) : Dataset for May 27, 2022 (all routes)

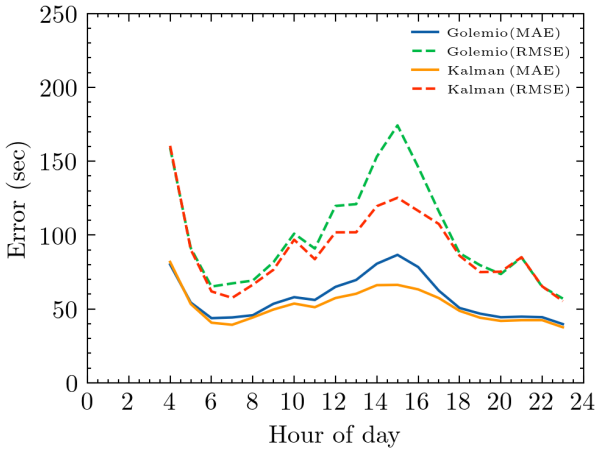


(c) : Dataset for May 27, 2022 (most used segment)

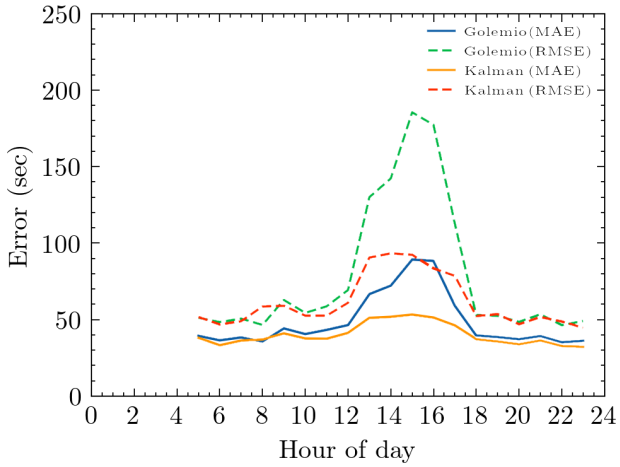
Figure 5.6: MAE and RMSE when limiting the number of predicted stops



(a) : Dataset for December 5, 2022 (all routes)



(b) : Dataset for May 27, 2022 (all routes)



(c) : Dataset for May 27, 2022 (most used segment)

Figure 5.7: MAE and RMSE for different hours of day

Table 5.4: Hour of day - average error reduction in comparison with the baseline solution

Dataset	MAE	RMSE
December 5, 2022 (all routes)	15.1%	14.5%
May 27, 2022 (all routes)	8.5%	8.0%
May 27, 2022 (most used segment)	13.7%	61.4%
Mean	12.4%	28.0%

Minimum standard deviation of delays

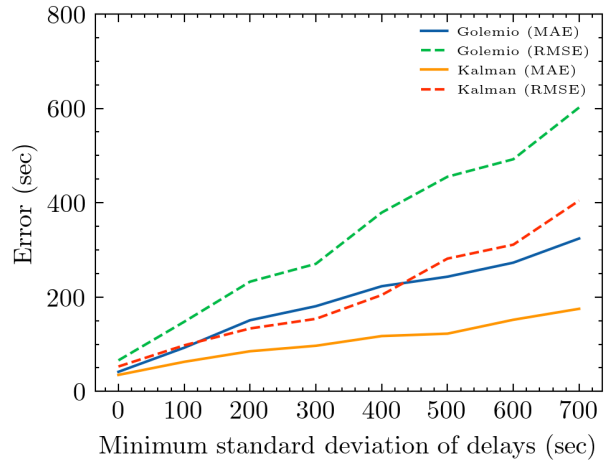
In Section 4.2 we introduced the cumulative distribution curve of standard deviations of delays (σ_{del}) within the same trip. It may seem that there are not many trips having large enough σ_{del} to consider it an appropriate use case, and that is true if we look at the trips for the whole day.

For example, we can look at the dataset for May 27, 2022 (for the most used segment). Only 4% of all trips have σ_{del} greater than 100 when taking the entire day into account. But if we look at the same statistics during peak hours (15:00-18:00, as can be seen from the graphs), then this number increases to 12% of all trips. Moreover, 8% of all trips have their σ_{del} greater than 200 and 5% greater than 300.

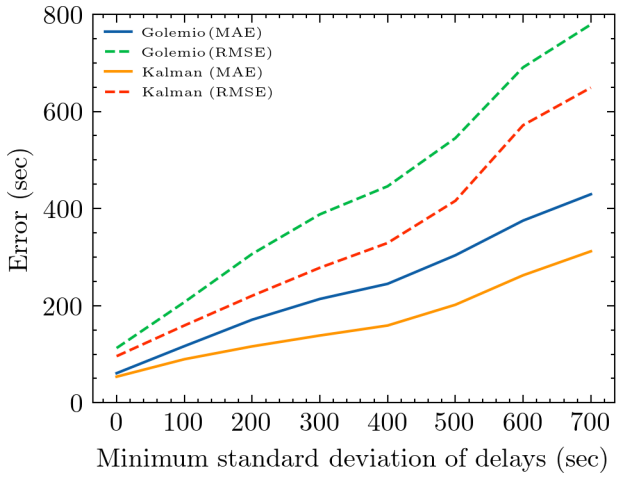
Figure 5.8 show how increase of minimum σ_{del} affects the prediction errors. They grow for both models, but our model grows at a slower rate, with the exception of the dataset for May 27, 2022 (all trips). Starting with the minimum standard deviation of 600, the error suddenly increases. But apart from that, our model compares with the baseline model best in this test case and has the highest average reduction of error.

Table 5.5: Minimum standard deviation of delays - average error reduction in comparison with the baseline solution

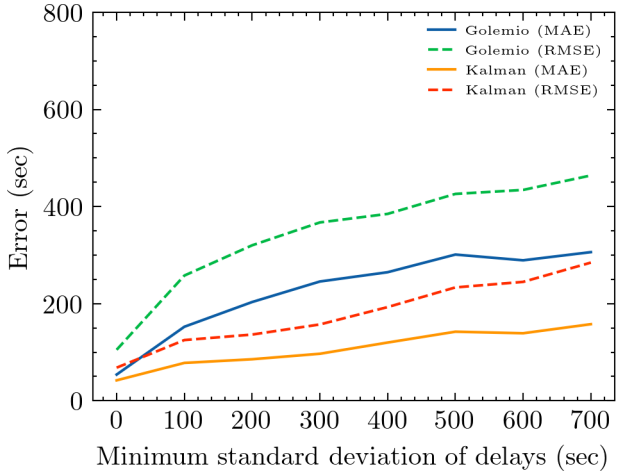
Dataset	MAE	RMSE
December 5, 2022 (all routes)	40.6%	38.6%
May 27, 2022 (all routes)	28.5%	25.3%
May 27, 2022 (most used segment)	49.6%	48.3%
Mean	39.5%	37.4%



(a) : Dataset for December 5, 2022 (all routes)



(b) : Dataset for May 27, 2022 (all routes)



(c) : Dataset for May 27, 2022 (most used segment)

Figure 5.8: MAE and RMSE when setting the minimum σ of delays

Chapter 6

Conclusion

In this thesis, we proposed a solution for predicting bus delays in real-time. Our solution is based on the Kalman filter algorithm, which we use to predict travel times of buses from their current position to future stops.

The essential part of the proposed method is the creation of a graph containing information about all bus routes, and using it to determine the road segments shared by multiple buses. The graph is optimized so that consecutive segments used by the same routes are merged into one segment, but no longer than 500 meters, allowing us to capture the transit situation among segments of reasonable length. Information about bus travel times along such segments is saved into the graph and used by the Kalman filter to predict travel times of subsequent buses.

One of the key ideas of our algorithm is to adapt to sudden changes in the current transit situation. In order to achieve this, we introduce minor modifications to the process of the Kalman filter algorithm, such as adding a fading memory effect, which gives less weight to older measurements. Initially, we also proposed to use the Expectation-Maximization algorithm for the automatic estimation of filter parameters, but its calculation turned out to be very slow for real-time purposes and led to only slightly better results, so it was not included in the final solution.

The primary parameters for configuring the algorithm by the end users are the number of future stops to predict and the past data time window, which allows limiting the age of data from preceding buses used to predict travel times.

As we expect bus position updates to come at irregular intervals, we use the Kalman smoother to generate missing data.

The implemented solution requires data in GTFS format, which specifies the distance traveled along the route to reach a specific stop. This information is also included in every bus position update, allowing us to determine whether the bus has reached the stop by comparing their

distances. One of the main advantages of GTFS is that although the proposed method is described on buses, it can work on other forms of public transport as well, as long as the same format is used.

We compare our solution with the baseline solution provided by Golemio API, which is officially used by Prague Integrated Transport. When using test datasets based on real data, the proposed method resulted in an average error reduction of up to 30%, with the most impressive results (up to 61%) achieved when applied for bus trips with widely varying delay times.

6.1 Future work

Future research can address the following issues:

- *Historical data.* Currently, our solution fallbacks to a solution similar to the baseline when no data is available about the travel time of preceding buses. The usage of historical data containing information about recurring public transit patterns at the predicted location can be considered. Such data might reduce prediction errors for less frequent routes.
- *Support of distributed prediction systems.* The current implementation assumes similar parameters for all predicted routes. A way to efficiently use the model independently for different routes or even trips could allow more fine-grained control of the predicted system (e.g., city or neighborhood) and handle known exceptions.
- *Expectation–Maximization.* Although we abandoned the EM algorithm for automatic parameter estimation because of its slow performance and unimpressive results, its usage may still be examined for occasional automatic parameter re-estimation.

Bibliography

1. TSK HMP. *Prague Transportation Yearbook 2018*. 2018. Tech. rep. Úsek dopravního inženýrství. Available also from: https://www.tsk-praha.cz/static/udi-rocenka-2018-en.pdf.
2. MAGISTRÁT HL. M. PRAHY. *Průzkum v MHD: 87% cestujících je spokojených* [online]. 2014. [visited on 2022-05-01]. Available from: https://www.praha.eu/jnp/cz/doprava/mhd/jaka_je_spokojenost_s_mhd_ukazal_pruzkum.html.
3. FABRIKANT, Alex. *Predicting Bus Delays with Machine Learning* [online]. 2019. [visited on 2022-05-01]. Available from: https://ai.googleblog.com/2019/06/predicting-bus-delays-with-machine.html.
4. GOOGLE. *Google Maps* [online]. [visited on 2022-12-07]. Available from: https://www.google.com/maps.
5. OPERATOR ICT. *Golemio API* [online]. [visited on 2022-12-07]. Available from: https://api.golemio.cz/v2/pid/docs/openapi/.
6. GOOGLE. *General Transit Feed Specification Reference* [online]. [visited on 2022-12-07]. Available from: https://developers.google.com/transit/gtfs/reference.
7. PRAŽSKÁ INTEGROVANÁ DOPRAVA. *Mapa PID* [online]. [visited on 2022-12-07]. Available from: https://mapa.pid.cz/.
8. *Transportation Research Thesaurus* [online]. [N.d.]. [visited on 2022-09-07]. Available from: https://trt.trb.org/.
9. XU, Haitao and YING, Jing. Bus arrival time prediction with real-time and historic data. *Cluster Computing*. 2017, vol. 20, no. 4, pp. 3099–3106. Available from DOI: 10.1007/s10586-017-1006-1.
10. YANG, Ming, CHEN, Chao, WANG, Lu, YAN, Xinxin and ZHOU, Liping. Bus arrival time prediction using support vector machine with genetic algorithm. *Neural Network World*. 2016, vol. 26, no. 3, pp. 205–217. Available from DOI: 10.14311/nnw.2016.26.011.

11. CHIEN, Steven I-Jy, DING, Yuqing and WEI, Chienhung. Dynamic Bus Arrival Time Prediction with Artificial Neural Networks. *Journal of Transportation Engineering*. 2002, vol. 128, no. 5, pp. 429–438. Available from DOI: 10.1061/(asce)0733-947x(2002)128:5(429).

12. SUN, Fangzhou, PAN, Yao, WHITE, Jules and DUBEY, Abhishek. Real-Time and Predictive Analytics for Smart Public Transportation Decision Support System. In: *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2016. Available from DOI: 10.1109/smartcomp.2016.7501714.

13. MA, Jiaman, CHAN, Jeffrey, RISTANOSKI, Goce, RAJASEGARAR, Sutharshan and LECKIE, Christopher. Bus travel time prediction with real-time traffic information. *Transportation Research Part C: Emerging Technologies*. 2019, vol. 105, pp. 536–549. Available from DOI: 10.1016/j.trc.2019.06.008.

14. CATHEY, F.W. and DAILEY, D.J. A prescription for transit arrival/departure prediction using automatic vehicle location data. *Transportation Research Part C: Emerging Technologies*. 2003, vol. 11, no. 3-4, pp. 241–264. Available from DOI: 10.1016/s0968-090x(03)00023-8.

15. VANAJAKSHI, L., SUBRAMANIAN, Shankar and SIVANANDAN, R. Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses. *Intelligent Transport Systems, IET*. 2009, vol. 3, pp. 1–9. Available from DOI: 10.1049/iet-its:20080013.

16. SHALABY, Amer and FARHAN, Ali. Prediction Model of Bus Arrival and Departure Times Using AVL and APC Data. *Journal of Public Transportation*. 2004, vol. 7, no. 1, pp. 41–61. Available from DOI: 10.5038/2375-0901.7.1.3.

17. KUMAR, Vivek, KUMAR, B Anil, VANAJAKSHI, Lelitha Devi and SUBRAMANIAN, Shankar C. Comparison of Model Based and Machine Learning Approaches for Bus Arrival Time Prediction. In: *Proceedings of the 93rd Annual Meeting*. Transportation Research Board, 2014, pp. 14–2518.

18. BIN, Yu, YANG, zhong and BAOZHEN, Yao. Bus Arrival Time Prediction Using Support Vector Machines. *Journal of Intelligent Transportation Systems - J INTELL TRANSPORT SYST*. 2006, vol. 10, pp. 151–158. Available from DOI: 10.1080/15472450600981009.

19. YU, Bin, LAM, William H.K. and TAM, Mei Lam. Bus arrival time prediction at bus stop with multiple routes. *Transportation Research Part C: Emerging Technologies*. 2011, vol. 19, no. 6, pp. 1157–1170. Available from DOI: 10.1016/j.trc.2011.01.003.

20. YIN, Tingting, ZHONG, Gang, ZHANG, Jian, HE, Shanglu and RAN, Bin. A prediction model of bus arrival time at stops with multi-routes. *Transportation Research Procedia*. 2017, vol. 25, pp. 4623–4636. Available from DOI: 10.1016/j.trpro.2017.05.381.

21. D'ANGELO, Matthew P., AL-DEEK, Haitham M. and WANG, Morgan C. Travel-Time Prediction for Freeway Corridors. *Transportation Research Record: Journal of the Transportation Research Board*. 1999, vol. 1676, no. 1, pp. 184–191. Available from DOI: 10.3141/1676-23.

22. YU, Bin, WANG, Huaizhu, SHAN, Wenxuan and YAO, Baozhen. Prediction of Bus Travel Time Using Random Forests Based on Near Neighbors. *Computer-Aided Civil and Infrastructure Engineering*. 2017, vol. 33, no. 4, pp. 333–350. Available from DOI: 10.1111/mice.12315.

23. KUMAR, B. Anil, JAIRAM, R., ARKATKAR, Shriniwas S. and VANAJAKSHI, Lelitha. Real time bus travel time prediction using k-NN classifier. *Transportation Letters*. 2017, vol. 11, no. 7, pp. 362–372. Available from DOI: 10.1080/19427867.2017.1366120.

24. ALAM, Omar, KUSH, Anshuman, EMAMI, Ali and POULADZADEH, Parisa. Predicting Irregularities in Arrival Times for Transit Buses with Recurrent Neural Networks Using GPS Coordinates and Weather Data. *Journal of Ambient Intelligence and Humanized Computing*. 2021, vol. 12. Available from DOI: 10.1007/s12652-020-02507-9.

25. ZHOU, Yuyang, YAO, Lin, CHEN, Yanyan, GONG, Yi and LAI, Jianhui. Bus arrival time calculation model based on smart card data. *Transportation Research Part C: Emerging Technologies*. 2017, vol. 74, pp. 81–96. Available from DOI: 10.1016/j.trc.2016.11.014.

26. RAMAKRISHNA, Y., RAMAKRISHNA, P., LAKSHMANAN, V. and SIVANANDAN, R. Use of GPS Probe Data and Passenger Data for Prediction of Bus Transit Travel Time. *Transportation Land Use, Planning, and Air Quality*. 2008. Available from DOI: 10.1061/40960(320)13.

27. KALMAN, Rudolph Emil. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*. 1960, vol. 82, no. Series D, pp. 35–45.

28. LABBE, Roger. *Kalman and Bayesian Filters in Python* [https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python]. GitHub, 2014.

29. SIMON, Dan. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. USA: Wiley-Interscience, 2006. ISBN 0471708585.

30. RAUCH, H. E., TUNG, F. and STRIEBEL, C. T. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*. 1965, vol. 3, no. 8, pp. 1445–1450. Available from DOI: 10.2514/3.3166.

31. SHUMWAY, R. H. and STOFFER, D. S. *An Approach To Time Series Smoothing And Forecasting Using The EM Algorithm*. Vol. 3. Wiley, 1982. No. 4. Available from DOI: 10.1111/j.1467-9892.1982.tb00349.x.

32. MADER, Wolfgang, LINKE, Yannick, MADER, Malenka, SOMMERLADE, Linda, TIMMER, Jens and SCHELTER, Björn. *A numerically efficient implementation of the expectation maximization algorithm for state space models*. Vol. 241. Elsevier BV, 2014. Available from DOI: 10.1016/j.amc.2014.05.021.

33. PRAŽSKÁ INTEGROVANÁ DOPRAVA. *Otevřená data PID* [online]. [visited on 2022-12-07]. Available from: https://pid.cz/o-systemu/opendata/.

34. PEZOA, Felipe, REUTTER, Juan L, SUAREZ, Fernando, UGARTE, Martin and VRGOC, Domagoj. Foundations of JSON schema. In: *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 263–273.

35. TSK HMP. *Prague Transportation Yearbook 2019*. 2019. Tech. rep. Úsek dopravního inženýrství. Available also from: https://www.tsk-praha.cz/static/udi-rocenka-2019-en.pdf.

36. TSK HMP. *Prague Transportation Yearbook 2020*. 2020. Tech. rep. Úsek dopravního inženýrství. Available also from: https://www.tsk-praha.cz/static/udi-rocenka-2020-en.pdf.

37. ARHIN, Stephen A., PTOE, P. E., GATIBA, Adam, ANDERSON, Melissa, RIBBISSO, Melkamsew and MANANDHAR, Babin. *Patron Survey of Acceptable Wait Times at Transit Bus Stops in the District of Columbia*. Vol. 09. Scientific Research Publishing, Inc., 2019. No. 04. Available from DOI: 10.4236/ojce.2019.94019.

38. VAN ROSSUM, Guido and DRAKE, Fred L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

39. SEISKARI, Otto. *simdkalman Documentation*. 2022. Version 1.0.2. Available also from: https://simdkalman.readthedocs.io/.

40. LABBE, Roger. *FilterPy Documentation*. 2022. Version 1.4.5. Available also from: https://filterpy.readthedocs.io/.

41. NETWORKX DEVELOPERS. *NetworkX Documentation*. 2022. Version 2.7.1. Available also from: https://networkx.org/documentation/networkx-2.7.1/.

Appendix A

Attachments

The following files are attached to the electronic version of this thesis:

- **data/**
  Dataset for May 22, 2022 (most used segment) described in Chapter 4
- **delay_prediction/**
  Source code
- **README.md**
  Brief guide to the project
- **Pipfile, Pipfile.lock, pyproject.toml, setup.cfg**
  Project dependency and configuration files