Combining Profile Similarity and Kalman Filter for Real-World Applicable Short-Term Bus Travel Time Prediction

Felix Schwinger*, Clemens Frohnhofen†, Jonas Wernz†, Stefan Braun*, Matthias Jarke*
*Databases and Information Systems, RWTH Aachen University, Germany
Email: {schwinger, braun, jarke}@dbis.rwth-aachen.de
†IVU Traffic Technologies AG, Germany
Email: {clf, jow}@ivu.de

***Abstract*— The emergence of intelligent transportation systems and the availability of detailed past trips and real-time data enabled more accurate travel time prediction algorithms. Passengers of public transportation highly value reliable service and accurate travel information. Unexpected delays while traveling impact traveler satisfaction enormously. We present a novel approach for a travel time prediction model for public transit in an urban setting. For this, we extended the existing profile similarity model based on k-medoids clustering and augmented it for an urban regions. The algorithm integrates real-time data from preceding vehicles traveling on the same links with a Kalman filter. Hence, the developed model uses the information available in modern ITS by combining historic travel time profiles with real-time data. While the algorithm focuses on short-term predictions, we have also evaluated it for long-term forecasts. Our results show that the proposed model outperforms benchmark models regarding certain criteria and can provide accurate travel times with comparably low-volume data input. Furthermore, the results indicate that the combination of a Kalman filter with a k-medoids approach improves the quality of the predictions.**

***Index Terms*— Bus Travel Time Prediction, Intelligent Transportation Systems, Public Transit, Data Science.**

I. Introduction

OVER the last decades, intelligent transportation systems (ITS) for public transport have evolved into distributed, highly complex, and versatile systems. For passengers, the predominant task of an ITS is providing accurate information on public transportation to public displays or personal smartphones in real-time. Today, this information includes the planned arrival times as well as the predicted arrival times of vehicles. An ITS also assists public transit operators by permanently monitoring the fleet, allows for control of the fleet, and stores all operational information for later use. Therefore, with increasing ITS availability, operational data's overall availability, including past trips of vehicles, also rises. This increase of available data enables more sophisticated travel time prediction algorithms.

The need for more accurate travel time prediction grows as mobility becomes increasingly interconnected in intermodal journeys. Intermodal journeys aim to combine the flexibility and low travel time of personal transportation with the sustainability and throughput of public transit by interlinking several mobility modes in one trip. For seamless intermodal journeys, an accurate prediction of travel time for public transit becomes increasingly essential; even a tiny deviation may lead to long waits when a critical transfer is missed.

There is a growing need for more environmentally sustainable transportation, with public transportation as its backbone. The transport sector's influence on climate change is vast. It is the single largest contributing sector of greenhouse gases and the only sector in which the greenhouse gas emissions have not reduced in the European Union since 1990 [1]. A considerable amount of greenhouse gases directly relate to personal transportation. Enabling people to switch from private to public transportation or intermodal journeys with large sections based on public transportation may help the sector become more eco-friendly. Reliability of the transportation strongly influences customers' satisfaction and loyalty in their chosen transportation mode [2]. A reliable transport service consistently operates as planned. Accurate real-time travel information increases the perceived reliability, even when delays occur [3]. Unexpected delays harm customer satisfaction, whereas early communicated delays increase reliability. Customers with access to accurate real-time information feel more in control of their trip and can better plan their waiting time and possible transfers to other transportation modes [4]. Indeed, reliability influences travel mode choice and may help travelers switch to more ecological sustainable transportation modes.

Travel time prediction algorithms for public transportation have been thoroughly researched, and various solutions have been proposed [5]. Past research has rarely focused on short-term travel time prediction (i. e., predictions for events less than 10 minutes in the future) but highlighted long-term travel prediction. Short-term travel prediction is crucial as passengers time their arrival at their origin station based on the available real-time information [4]. To date, proposed prediction algorithms are either efficient or accurate but seldom both, limiting their practical large-scale applicability in ITS. Most of the previous approaches evaluate prediction algorithms only on a single line of a public transportation network; hence, scalability improvements toward multi-line predictions are necessary. Therefore, we aim to introduce an accurate short-term travel time prediction algorithm for public transportation with practical applicability in ITS.

Felix Schwinger and Matthias Jarke are additionally affiliated to Fraunhofer FIT, Germany.

Our proposed hybrid algorithm is based on the profile similarity model (PSM) [6] and a Kalman Filter (KF) [7]. As the PSM is initially developed for *long*-distance, intercity bus travel, we adapted the algorithm for *short*-distance, intracity bus travel. The PSM incorporates past trip data into the prediction, whereas the KF incorporates recent trip information from other vehicles. To check whether the proposed hybrid algorithm is both scalable and accurate on short-term travel time predictions in an urban context, we performed a simulation as a case study with data from the city of Aachen. Furthermore, the proposed hybrid algorithm will be compared with other travel time prediction approaches such as neural networks.

The remainder of this paper is structured as follows. Firstly, Section II introduces the scenario for short-term travel prediction algorithms and highlights their relevant qualities. Secondly, Section III discusses relevant related work. Section IV introduces our proposed solution in detail. In Section V, we present the algorithm's evaluation. Finally, Section VI concludes the paper and provides an overview of further research challenges.

II. Scenario

Public transportation passengers are often not interested in the travel time but the arrival or departure time of vehicles at particular stop points. However, the problem of arrival or departure time prediction can be reduced to the travel time prediction of vehicles [8]. We define bus travel time prediction as short-term if the bus's planned arrival time is less than 10 minutes in the future from a travel time request. Such a time horizon is especially interesting for urban scenarios. On the one hand, people usually estimate the time needed to reach their departure station and look at real-time information just in time before departing [4]. On the other hand, the passenger's trips in urban settings are relatively short [9]. Hence, we focus on short-term bus travel time prediction instead of long-term prediction, typically used for intercity travel with longer distinct journeys.

Predictors of any kind face multiple challenges. In general, there is often a trade-off between accuracy and complexity. The definition of the following six quality criteria regarding the practical application in ITS was inspired by the comparison of Cristóbal et al. [6]: **Q1: Accuracy.** For a prediction model, accuracy is arguably the most crucial property. It measures how large the deviation between a predicted travel time and the measured travel time is. **Q2: Data Demand.** The amount of required training data consisting of past trips is essential, as these need to be gathered before the algorithm can predict travel times. **Q3: Time Demand.** The duration of the training phase for the predictor is also crucial. If a model is not scalable, time demand is critical. However, most often, the data demand is more important than the time demand, as the time demand can be reduced by increasing the system's computation power. In contrast, operators cannot compensate for a large data demand as easily. **Q4: Scalability.** As public transit operators operate multiple lines, an algorithm not only needs to predict travel times for a single line but the whole transportation network. Scalability describes a predictor's ability to be used on a complete transportation network. **Q5: Flexibility.** Public transit operators regularly update their transportation network or adapt their timetables. Flexibility describes a predictor's ability to react to timetable changes without extensive retraining. For the practical applicability of a model in ITS, flexibility is an important property. **Q6: Interpretability.** For most predictors, the interpretability of the results is a desired property. Interpretability describes how well experts can comprehend the internal prediction process. As the travel time prediction is used to inform the passengers of expected delays and used for the bus dispatcher to react to events appropriately, the dispatcher must understand why certain travel time predictions occur.

When balancing accuracy and complexity, ITS users should prefer a low data demand (Q2) over a low computational demand (Q3). Even in urban scenarios with frequent trips, data collection takes a long time. This observation holds especially for models that are not scalable (Q4) and not flexible (Q5), i. e., require retraining once the network is slightly adapted.

III. Literature Review

In this section, we present an overview of current advances toward accurate and fast travel time prediction. We will introduce prediction models grouped by their different foundations into five distinct groups. Afterward, we will discuss each prediction model.

A. Simple Models

Simple models do not require much data and are less complicated than other algorithms. Typically, this simplicity results in poor accuracy. *Delay-based models (Delay)* offset the current delay or earliness to the scheduled arrival times while incorporating the buses' dwell time [10].

B. Machine Learning Regression Models

Machine learning regression (MLR) models learn parameters of mathematical functions to predict a value based on a set of feature variables. *Linear regression (LR)* models assume a linear relationship between the target variable and independent features [11, 12]. In contrast, *kernel regression (KR)* allows for a non-linear relationship between the target variable and the feature set. Both LR and KR reveal important factors that influence travel times [13, 14]. *Support vector machines (SVM)* classify data by separating the clusters with the broadest hyperplane possible. SVMs can also learn non-linear relationships using the kernel trick [15, 16]. Models based on SVM deliver good results. However, SVMs do not scale to larger problems [17], making a single SVM for a complete transportation network infeasible. In the domain of travel time prediction, *artificial neural networks (ANN)* are also widely used. The accuracy of these models is high; however, all regarded publications only evaluate one bus line [14, 18–21]. For travel time prediction, a particular variant of recurrent neural networks, *long short-term memory*

networks (LSTM) [8, 17], have proven to be promising. These networks are suited for identifying long-term patterns while also incorporating more dynamic short-term influences. Furthermore, they are well-suited for modeling time series data. Neural networks have a very high predictive power, but they need a lot of training data and training time [14]. Finally, models based on the *k-nearest-neighbor (k-NN)* regression algorithm deduce the value of the target variable based on the similarity to the nearest neighbor [22, 23].

C. State-Based Time Series Models

State-based time series (SBTS) models assume that the target variable's value is dependent on a state variables' past values. The *Kalman Filter (KF)* uses a two-step iterative procedure to estimate the value of state variables by including new measurements. In the first step, the state variables are estimated based on a mathematical model. In the second step, these estimations are updated with recent measurements, allowing KF models to handle inaccuracies in the data well [7, 24]. While *smoothing functions (Smoothing)* are less common in the literature than other approaches, some have been proposed and evaluated [25]. *ARIMA* models have also been proposed but are slightly less accurate than other regression or state-based approaches [26]. Both ARIMA and smoothing models need to be more thoroughly investigated for this problem area.

D. Clustering

One prediction model called *profile similarity model* (PSM) extracts historic travel profiles from clusters of past trips [6]. The model uses k-medoids clustering with the Manhattan distance. Each historic travel profile represents the prominent travel behavior of one cluster. During prediction, the model determines the historic travel profile most similar to the current driving behavior. The model then derives the travel times from this historic travel profile as predictions. The PSM was initially developed for intra-city travel time predictions. Hence, the model directly operates on a vehicle's on-board computer independent of outside information.

E. Hybrid Models

Hybrid models combine multiple prediction algorithms into a single model. Most commonly, machine learning regression algorithms are combined with a Kalman filter, e. g., SVMs have been combined multiple times with Kalman filters [27–29]. Similar approaches have also been conducted with ANNs [30, 31]. However, all considered hybrid models have only been used on a single bus line, most likely because of the significant data demand for multiple lines of SVM-based and ANN-based models. The authors conclude that, both for SVM and ANN-based models, the combined models outperform single-predictor models in terms of accuracy.

F. Discussion

In this discussion, we compare the introduced approaches using the proposed quality criteria on a coarse-grained level. For this, we describe each predictor's qualities, elucidate our proposed prediction algorithm as a hybrid model combining an adapted PSM [6] with a KF model [7] and finally motivate our decision. Table I qualitatively summarizes the results: the more filled the circle, the better a model performed according to this criteria. Unfortunately, quantitatively comparing all predictors is not possible, as often neither the algorithm's code nor the required data is publicly available.

TABLE I: Comparison of travel time prediction models.

Method		Accuracy	Data demand	Time demand	Scalability	Flexibility	Interpretability
Simple	Delay	○	●	●	●	●	●
MLR	LR [11, 12]	◔	◑	◑	◕	◕	◕
	KR [10]	◑	◑	◑	◕	◑	◑
	SVM [16]	◕	◔	◔	◔	◑	◔
	ANN [14, 18–21]	◕	◔	◔	◔	◔	◔
	LSTM [8]	●	◔	◔	◔	◔	◔
	k-NN [22]	◔	◔	◔	◑	◑	◑
SBTS	KF [7, 24]	◑	◑	◕	◕	●	◕
	Smoothing [25]	◔	◑	◕	●	◑	◕
	ARIMA [26]	◔	◑	◑	◕	◑	◑
Clustering	PSM [6]	◑	◑	◑	◕	◕	◑

Q1: Accuracy. In terms of prediction accuracy, complex models generally outperform simpler ones: In particular, SVMs, ANNs, or LSTMs outperform the state-based models [8, 27, 32]. For k-NN models, long-term predictions are problematic [22]. The more data and computational resources a model requires, the better generally the accuracy of the model. Regarding the hybrid models, all models perform better when combined with a Kalman filter as recent traffic patterns are incorporated into the forecast [27, 29].

Q2 and Q3: Complexity. Regarding the complexity, we differentiate between data demand (Q2) and time demand (Q3). The comparison shows that the more accurate models also have a longer runtime and require larger datasets for their prediction. Longer runtime can be compensated with more considerable computational power; for instance, Petersen et al. [8] show that their powerful ConvLSTM model can be trained in a short amount of time on commodity hardware utilizing GPUs. A more considerable data demand, however, remains more critical.

Q4: Scalability. Regarding scalability, there are two ways to scale up models to a transport network. The first approach creates a model for each line. This scaling method is suitable when models have low complexity (Q2 & Q3) or are very flexible (Q5). SVMs, PSM and, K-NNs are typically scaled this way. The second approach uses a single model to predict the travel times for an entire network. Neural networks are generally suitable for such a task. Nevertheless, to our knowledge, no neural network has demonstrated such an ability in a larger practical setting for travel time prediction.

Q5: Flexibility. Models that do not require prior training are entirely flexible, such as the Delay-based and KF model. They can generate predictions for different transportation networks without much adaptation. Models that require prior training and build an internal model, such as ANNs, LSTMs, SVMs, or the PSM, are not as flexible as changes in the network often necessitate retraining.

Q6: Interpretability. For prediction algorithms that build an internal model such as SVM or ANN, it is hard to deduce why the model comes to specific predictions, resulting in low interpretability (Q4). Models such as the Delay-based model, the KF, or even the PSM are usually highly interpretable. The KF adjustments are easy to follow, whereas the number of PSM clusters is low (typically lower than 6) and corresponds to typical observable states, such as rush hour traffic.

If we define a model's efficiency as a combination of Q2 to Q6, it becomes evident that most current models trade accuracy against efficiency As seen in Table I, most accurate models are not efficient, while most efficient models are not accurate. On one side is the delay-based model, which is quite efficient, while the ConvLSTM model by Petersen et al. [8], which is the most accurate model to date, is on the other side. The ConvLSTM model's accuracy is achieved by training on a large dataset and only regarding a single line with a limited number of stop points that do not diverge, hinting at limited scalability and flexibility. The ConvLSTM's ability to provide high-quality predictions only based on trip recordings is exceptional. However, as we set out to create a prediction algorithm that performs well in practice, we want to create a model that better balances accuracy and efficiency.

When looking at models with average efficiency, only models with average accuracy are left, such as PSM, KF, or KR. The PSM's strength is the clear separation of data, the efficient reduction of data, the excellent prediction quality, and its interpretability. The model itself only uses a few historic travel profiles that are computed in a previous clustering step. Compared to ANNs, the PSM requires a far smaller dataset during training. Hence, the PSM is a promising approach for travel time prediction for inter-city transportation in practice.

Like existing hybrid models, we hypothesize that the PSM's accuracy could be increased by integrating a KF. The KF requires no training, and the additional needed real-time information is already present in most ITS systems. In other words, combining average models such as PSM and KF into a hybrid model might result in an applicable model with reasonable accuracy and maintainable complexity.

IV. Profile Similarity and Kalman Filter Model

Our proposed model PSM+KF combines the profile similarity model (PSM) [6] with a Kalman Filter (KF) [7]. The PSM was developed with intra-city bus travel in mind. Therefore, the model is independent of a connection to an ITS. In urban areas, vehicles usually transmit information constantly to an ITS, allowing the PSM+KF to use more information. The original PSM predicts travel times based only on a small subset of a route's stop points. This approach has proven to work well in inter-city travel time prediction [6]. However, as urban traffic fluctuates more and is more prone to sudden unexpected events, the algorithm must consider all stop points. The inclusion of a Kalman Filter in the PSM+KF model allows the use of all available information of preceding vehicles and allows to react to sudden urban traffic events. This inclusion, however, also assumes a constant data connection between vehicles and the ITS; without this connection, the predictor can not compute any forecasts.

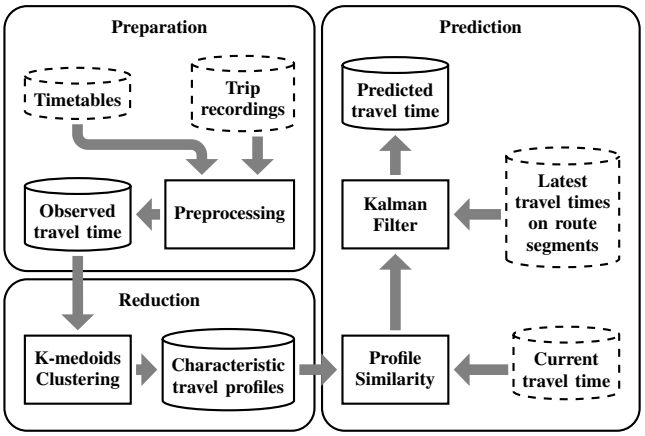


Fig. 1: Overview of the model's workflow: Dashed round boxed represent input, solid round boxes represent processed output and squared boxes represent a data processing step.

The PSM+KF's workflow consists of three distinct steps (Fig. 1): *Preparation*, *Reduction*, and *Prediction*. The preparation step is for preprocessing the data. It consists of computing observed travel times from past trips based on the respective timetables and measured travel times from the buses' on-board computers. The Reduction, which incorporates the training phase, applies the adapted PSM clustering and computes the characteristic travel profiles based on past observed travel times. As the final step, the algorithm then predicts the total travel times in two stages. First, the algorithm predicts a travel time with the most similar characteristic travel time profile based on the current travel time. The KF then finalizes this forecasted travel time by incorporating the travel times of preceding vehicles into the prediction. As modern ITS usually perform the preparation step, we are going to focus the discussion on the Reduction and Prediction steps:

A. Reduction

The reduction step's main idea is to compute k characteristic travel profiles (CTPs) from past observed travel times obtained in the Preparation step. Each CTP highlights prominent travel behaviors, i. e., reducing the amount of data required for each prediction while not losing important information. The CTPs are extracted with the k-medoids clustering approach similar to the original PSM model [6].

We adapted the algorithm of Cristóbal et al. [6] in two ways: We regard all stop points of a line when applying the k-medoids algorithm, and we switched the Manhattan to a fractional distance metric. Considering all stop points of a line is feasible as the prediction is performed in a central ITS in our urban scenario. We assume that a much-reduced bus stop set, i. e. regarding fewer stops, cannot accurately represent highly fluctuating traffic patterns. There is a slight difference in accuracy when comparing distance-based and stop-based data sets [32]. Therefore, we expect higher accuracy when the k-medoids clustering step regards all stop points of a line. Regarding the switch of the distance metric to a fractional metric: Commonly used distance metrics, such as the Manhattan or Euclidean distance, lose their meaningfulness in high dimensional spaces and thus are not suitable for clustering [33]. As the reduction step does not reduce the number of stop points of a line, the number of dimensions grows with more stop points. Therefore, we employed the fractional distance metric (1), which is proven to work well with high-dimensional data [33]. The metric defines the distance between two points x and y in a d-dimensional space with the fractional distance norm parameter f, where $f \in (0,1)$.

$$dist_f^d(x,y) = \sum_{i=1}^{d} [(x^i - y^i)^f]^{1/f} \tag{1}$$

We run the k-medoids algorithm $n-1$ times from 2 to a defined upper limit n with a different value of k clusters each time to evaluate the best number k for each dataset. Afterward, we select the cluster with the highest Silhouette score. The Silhouette score measures the consistency of a cluster [6, 34]: It represents how well a data point fits into a given cluster. The mean Silhouette score of all clusters provides a reasonable estimation of the overall quality of the clustering. We, therefore, select the cluster with k CTPs that had the highest Silhouette score for the Prediction step. Our simulation has shown that an upper limit n of 10 is sufficient for most lines, as the Silhouette score does not increase anymore. The CTPs are then computed and stored for each line.

During the implementation, the k-medoids clustering approach was prone to overfitting. This overfitting sometimes resulted in one cluster containing a handful of trips, whereas another cluster contained all remaining trips. When the small cluster adjusts to a group of outliers, often representing sporadic, extreme behavior or errors, the Silhouette score indicates a perfect fit to the data. Returning these clusters as CTPs will likely degrade the accuracy of the predictions. Therefore, we have introduced a balancing factor defined as the ratio between the number of trips in the largest and smallest cluster. The algorithm can then dismiss clustering results not complying with a given balancing factor, improving the predictions' quality.

B. Prediction

The prediction is a two-step process (Fig 1): First, the PSM gives us a prediction, and secondly, a Kalman Filter augments this prediction. The algorithm divides a bus line into segments that each lie between two stop points. Assume a bus has arrived at a stop s_1 and that a passenger requests a prediction for stop s_2 further down this bus's line. Then, the travel time to stop s_2 from the start of the trip consists of two components: The sum of observed travel times on all segments until stop s_1 and the sum of predicted travel times on all segments between stop s_1 and s_2. Each segment's travel time is split into a dwell time at the stop and a running time. Depending on whether the bus is currently at the stop, the dwell time at stop s_1 is observed or predicted. The running time of all segments after s_1 are predicted travel times, even if the bus is on the segment directly after stop s_1.

The predicted dwell and running time are derived from the most representative CTP. The decision criterion for selecting the most representative CTP is the travel time difference until the last visited stop between the CTP and the running trip. After choosing the CTP, the algorithm derives the dwell time from the CTP. The predicted dwell time at stop s_i is equal to the difference between the departure and arrival time at stop s_i from the CTP. Similarly, the predicted running time from stop s_i to s_{i+1} is computed.

The Kalman filter then updates the predicted running time by incorporating a set of observed running times from other vehicles that recently traveled on the preceding route segment. We decided not to integrate previous vehicles' dwell time in the prediction update as it is unclear how the dwell time of preceding vehicles influence the dwell times of subsequent vehicles. The algorithm performs an iteration for each element in the set of observed running times. The integration of the Kalman Filter is similar to other hybrid prediction algorithms [24]; therefore we only highlight the model's changes.

$$x_t = Fx_{t-1} + Bu_{t-1} + w_{t-1} \Rightarrow x_t = x_{t-1} + w_{t-1} \tag{2}$$

In (2), the Kalman Filter's process model, which defines the state's progression from iteration $t-1$ to t, is given. The state vector x consists of only the predicted running time. Therefore, the state transition matrix F reduces to a scalar value. As there is no relationship between the predicted running time between iteration t and $t-1$, F is set to 1. Furthermore, no external influences are modeled; hence the term Bu_{t-1} that models external influences is omitted. To still have the process noise covariance as a tuning parameter, the model still contains the process noise vector w.

$$z_t = Hx_t + v_t \Rightarrow z_t = x_t + v_t \tag{3}$$

In (3), the Kalman Filter's observation model is given. The model defines the relation between the observation z and the state x. The observation vector z contains one observed running time from the set of observed running times during each iteration step. The observation matrix H reduces to a scalar value and is set to 1 because both the observation and state models simulate the running time on the same segment. The term v_t defines the observation noise and is kept as a tuning parameter. The defined Kalman filter then

computes the travel time until stop point $s_1 + 1$ is reached. As long as the stop $s_1 + 1$ is not equal to stop s_2, the algorithm continues to predict dwell and running times. Once the algorithm computes the travel time to s_2, it can terminate and return the requested accumulated travel time prediction.

V. Evaluation

To evaluate our algorithm's performance in a real-world scenario, we first introduce our simulation methodology based on a real-world dataset. Afterward, we compare its results to other approaches.

A. Methodology

The available dataset contains 1.4 million trip recordings on 108 routes in Aachen in Germany from the year 2019. As a pre-selection, we dropped incomplete trips, selected evenly distributed recordings throughout the year, and selected lines with sufficient recordings. After the pre-selection, 28 lines remain, which correspond to 120,000 recorded trips. Models that can handle multiple lines, create predictions for all lines, whereas models that can only predict for a single line, predict on a single representative line.

We split the dataset into training and validation sets by a fixed date to mimic realistic conditions, i. e., the prediction models can use all observations before that date for training. Afterward, all further observations are treated as real-time measurements for the Kalman Filter, which matches an actual deployment's conditions. All simulations have been performed on an Intel Core i7-9700 CPU with eight cores and 16 GB of RAM without GPU acceleration.

Each simulation run contains parameters relevant during a simulation. The request time is defined as the difference between a prediction request's placement and the planned arrival time. From the passengers' perspective, high accuracy at small request times is more crucial than accuracy at very high request times. Hence, all simulations create prediction requests with request times of 1, 2, 5, 10, 15, 30, 45, 60, and 120 minutes. Thereby, the evaluation sets a focus on small request times.

For measuring the accuracy of predicted travel times, we use the Root Mean Squared Error (RMSE) as given in (4).

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (\mathrm{oTT}_i - \mathrm{pTT}_i)^2}{n}} \tag{4}$$

To compute the RMSE, we sum up for each stop s_i the difference between the observed travel time oTT_i and the predicted travel time pTT_i. The RMSE penalizes large prediction errors stronger; hence the further a travel time prediction diverges from the actual travel time, the stronger it is penalized. This evaluation does not present other metrics such as the Mean Absolute Error or Mean Absolute Percentage Error, as these metrics have shown the same trend as the RMSE.

B. Results

Table II shows the tunable parameters of the models and their optimal values. The window size determines how far back from the time of the request the KF incorporates observed running times, and the appliance limit specifies how long the KF is enabled in the prediction process. We assume that the parameters' influences on the prediction accuracy are independent of each other. Hence, in each of our simulations, only one parameter is varied while the other parameters are set to a fixed value. After computing the parameters, we verified this assumption by performing a grid search with the parameters in the found solution's vicinity. All parameters except for the process noise variance might depend on the use case, and we can define only general trends. However, all parameters significantly influence the model's accuracy.

TABLE II: Optimal parameters for the model

Parameter	Model	Optimal Value
Cluster Balancing Factor	PSM	35
Fractional Distance Norm Parameter	PSM	0.1
Number of Historic Trips	PSM	$\geq$ 1250
Process Noise Variance	KF	0 s^2
Observation Noise Variance	KF	500 s^2
Window Size	KF	45 min
Appliance Limit	KF	25 min

Next, we compare the accuracy of the different models. Fig. 2 compares the newly developed PSM+KF model to the simple delay-based and ConvLSTM models [8] for request times of 45, 10, and 2 minutes. If not otherwise stated, the ConvLSTM is trained on a single line with 1000 trips, while all other models are trained and averages over all 28 lines, as the ConvLSTM model is unable to predict for multiple lines. Furthermore, the distinct parts of the PSM+KF model are evaluated separately to see their particular influence on the prediction. Only the PSM+KF and the Kalman Filter utilize real-time data for all evaluation results, while the other approaches only predict based on past trip recordings. For large request times, the models which create a model based on recorded data such as the PSM+KF and the ConvLSTM significantly outperform the Kalman Filter. The PSM+KF model performs best here, closely followed by the PSM and ConvLSTM model. For medium request times, the accuracy of all predictors significantly increases. The KF has the most significant increase in accuracy as it is only able to utilize real-time information. Most models here perform equally well. The ConvLSTM model performs best, even without real-time information, directly followed by the PSM+KF model. For request times of 2 minutes, the KF can give even better predictions better than the delay-based model. Here, the ConvLSTM model is also the best predictor, with the PSM+KF model close behind. The data shows that the KF can substantially increase the PSM's performance for medium and short request times, enabling the model to react to sudden changes. The performance of the ConvLSTM model, even without real-time data and without the request times matching its internal time resolution of 15 minutes is exceptional. The PSM+KF model closely follows the accuracy of the ConvLSTM model, therefore we compare the performance of these two models in more detail.

For this comparison, we train the ConvLSTM and

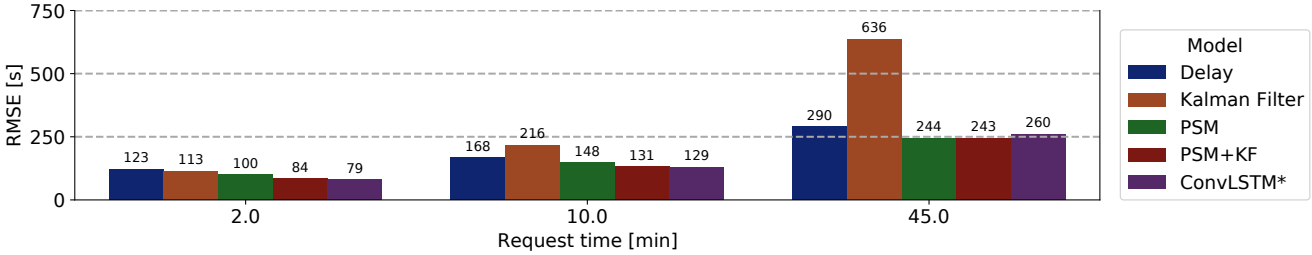
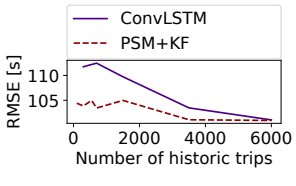
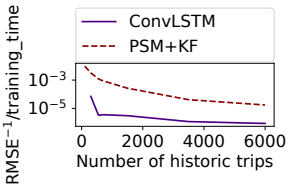


Fig. 2: Comparison of prediction accuracy of different models at request times of 45, 10, and 2 minutes. All models except the ConvLSTM model are tested on 28 lines over a month. As the ConvLSTM model can handle only one line, a ConvLSTM models trains on the line with most available data points. Average training set size: 750 trips. ConvLSTM line: 1000 trips, 38 stops, trip frequency 15 minutes, trip duration 106 minutes.

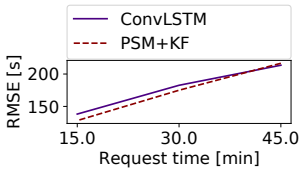
* The ConvLSTM model allows only predictions of one line. Hence, the measurements are more biased by the data.



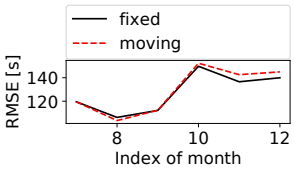
(a) Accuracy with varying training set sizes.



(b) Accuracy per training time with varying training set sizes.



(c) Accuracy comparison. Line with 30 stops and 700 trips.



(d) Stability PSM+KF. Fixed and moving training period.

Fig. 3: Performance comparison between the PSM+KF and ConvLSTM model and stability of the PSM+KF model.

PSM+KF models on the same line with the same data. The line is the ConvLSTM line from Fig. 2 as this line has the largest amount of trips available. The PSM+KF model utilizes available real-time data, while the ConvLSTM only utilizes historic data. In Fig. 3a, the impact of different training set sizes on accuracy is investigated. The PSM+KF model performs better with a low amount of training data. With a high volume of training data, both models achieve similar accuracy. The ConvLSTM model requires at least 300 data samples, whereas the PSM+KF model can already work with fewer samples.

In Fig. 3b, the accuracy per training time unit in relation to the training samples is shown. It becomes apparent that the PSM+KF model requires much less data and less time to predict in a similar quality to that of a neural network.

Fig. 3c compares the accuracy of the ConvLSTM with the PSM+KF model at request times of 15, 30, and 45 minutes, matching the time resolution of the ConvLSTM model [8]. The ConvLSTM achieves similar accuracy as the PSM+KF model at request times of 15 and 30 minutes. At request times of 45 minutes, the ConvLSTM model outperforms the PSM+KF model. In practice, the travel time's request time will rarely precisely match the neural network's resolution, giving the PSM+KF model a slight advantage: The average RMSE of all request times, including those not shown in the graph, is lower for the PSM+KF (103.44 s) than for the neural network (112.47 s).

Fig. 3a to Fig. 3c show that the accuracy of the PSM+KF and the ConvLSTM model are approximately on par, with the PSM+KF model sometimes outperforming the ConvLSTM and vice-versa. However, the comparison of the required training data shows that the PSM+KF model requires much less training data and requires much less time for the training. The reached accuracy divided by the training time is two orders of magnitude less than that of the ConvLSTM model, highlighting the small complexity of the PSM+KF model in terms of data and time demand.

Fig. 3d highlights the stability of the PSM+KF model. We compared training on a fixed dataset from June with a moving training horizon. For the moving training, the model trains on data of the previous month to the prediction month. Both models predict with roughly the same accuracy, making it unnecessary to retrain the model often. Characteristics in the data cause the fluctuations of the RMSE in October.

We were surprised by the similar accuracy between the PSM+KF and ConvLSTM models. The similar performance stems from the limitation of the training data to 700 trips. On the one hand, the average dataset available does not provide any more recorded trips: The average number of recorded trips per line decreases with a greater variety of lines. On the other hand, the lack of data also stems from the evaluation, as fair evaluation conditions set the same training data limit for all models. Moreover, most lines have a service frequency of 15 minutes or less, leading to the temporal grid's step size of at least 15 minutes. Likely, the accuracy at low request times increases drastically with a smaller time step size; however, the vehicle's trip frequency must be similar high, so that each time step has at least multiple observations. In the future, we aim to compare the PSM+KF model with a perfectly trained ConvLSTM model.

VI. Conclusion and Future Research

This paper motivated the need for an algorithm for short-term bus travel time prediction in urban areas. In contrast to related work, the scalability and the applicability of the model in practice were critically important. Our algorithm's design ensures that it can combine available recorded data of past trips and relevant real-time information from other vehicles into a high-quality travel time prediction. The prediction algorithm consists of two distinct steps. First, a prediction is computed based on information of past travel times. Secondly, this prediction is adjusted with real-time information from other vehicles that recently traveled on the same links.

Our evaluation highlights both accuracy and scalability and demonstrates the practical application of the proposed model. Even though we set out to create a scalable algorithm with average accuracy, it keeps pace with state-of-the-art neural networks while requiring only a fraction of the training data. These properties stem from the excellent prediction quality of the profile similarity model and the inclusion of real-time information through the Kalman Filter.

For future work, we are mainly interested in improving the approach's accuracy and practicability and extending the evaluation by integrating more approaches. We examine incorporating additional information sources available in ITS into the prediction, e. g., external traffic monitoring services. Moreover, we want to improve the algorithm's practical usability by handling edge cases. Currently, the k-medoids clustering, for example, does not always return any clusters due to discarding clusters based on the balancing factor. Another unhandled edge case is predicting the departure time for the first stop of a line. Accumulated delays from previous trips are also not transferred to the next trip of a bus.

With PSM+KF, we present a practical and accurate travel time prediction algorithm for public transit operators. The algorithm uses readily available data in ITS, trip recordings, and real-time data and focuses on practical applicability.

References

[1] European Comission, "A European Strategy for low-emission mobility," https://ec.europa.eu/clima/policies/transport_en/, 2016, [Online; accessed 24-February-2021].

[2] A. König and K. W. Axhausen, "The reliability of the transportation system and its influence on the choice behaviour," in *Proceedings of the 2nd Swiss Transportation Research Conference*, 2002.

[3] D. Van Lierop, *et al.*, "What influences satisfaction and loyalty in public transport? a review of the literature," *Transport Reviews*, vol. 38, no. 1, pp. 52–72, 2018.

[4] C. Brakewood and K. Watkins, "A literature review of the passenger benefits of real-time transit information," *Transport Reviews*, vol. 39, no. 3, pp. 327–356, 2019.

[5] M. Altinkaya and M. Zontul, "Urban bus arrival time prediction: A review of computational models," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 2, no. 4, pp. 164–169, 2013.

[6] T. Cristóbal, *et al.*, "Bus travel time prediction model based on profile similarity," *Sensors*, vol. 19, no. 13, p. 2869, 2019.

[7] A. Shalaby and A. Farhan, "Prediction model of bus arrival and departure times using avl and apc data," *Journal of Public Transportation*, vol. 7, no. 1, p. 3, 2004.

[8] N. C. Petersen, *et al.*, "Multi-output bus travel time prediction with convolutional lstm neural network," *Expert Systems with Applications*, vol. 120, pp. 426–435, 2019.

[9] M. E. Bouwman and H. Voogd, "Mobility and the urban-rural continuum," *Global Built Environment Review*, vol. 4, no. 3, 2005.

[10] M. Sinn, *et al.*, "Predicting arrival times of buses using real-time gps measurements," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2012, pp. 1227–1232.

[11] J. Patnaik, *et al.*, "Estimation of bus arrival times using apc data," *Journal of public transportation*, vol. 7, no. 1, p. 1, 2004.

[12] S. Arhin and R. Stinson, "Transit bus travel time prediction using avl data," *Int. J. Eng. Res. Technol*, vol. 5, pp. 21–26, 2016.

[13] W. Fan and Z. Gurmu, "Dynamic travel time prediction models for buses using only gps data," *International Journal of Transportation Science and Technology*, vol. 4, no. 4, pp. 353–366, 2015.

[14] Z. K. Gurmu and W. D. Fan, "Artificial neural network travel time prediction model for buses using only gps data," *Journal of Public Transportation*, vol. 17, no. 2, p. 3, 2014.

[15] Y. Bin, *et al.*, "Bus arrival time prediction using support vector machines," *Journal of Intelligent Transportation Systems*, vol. 10, no. 4, pp. 151–158, 2006.

[16] M. Yang, *et al.*, "Bus arrival time prediction using support vector machine with genetic algorithm," *Neural Network World*, vol. 26, no. 3, p. 205, 2016.

[17] J. Pang, *et al.*, "Learning to predict bus arrival time from heterogeneous measurements via recurrent neural network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, 2018.

[18] J. Jabamony and G. Shanmugavel, "Iot based bus arrival time prediction using artificial neural network (ann) for smart public transport system (spts)," *Int. J. Intell. Eng. Syst.*, vol. 13, no. 1, 2020.

[19] R. Jeong and R. Rilett, "Bus arrival time prediction using artificial neural network model," in *Proceedings. The 7th international IEEE conference on intelligent transportation systems (IEEE Cat. No. 04TH8749)*. IEEE, 2004, pp. 988–993.

[20] Y. Lin, *et al.*, "Real-time bus arrival time prediction: case study for jinan, china," *Journal of Transportation Engineering*, vol. 139, no. 11, 2013.

[21] T. Yin, *et al.*, "A prediction model of bus arrival time at stops with multi-routes," *Transportation research procedia*, vol. 25, 2017.

[22] C. Coffey, *et al.*, "Time of arrival predictability horizons for public bus routes," in *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, 2011, pp. 1–5.

[23] H. Chang, *et al.*, "Dynamic multi-interval bus travel time prediction using bus transit data," *Transportmetrica*, vol. 6, no. 1, 2010.

[24] F. Sun, *et al.*, "Real-time and predictive analytics for smart public transportation decision support system," in *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2016.

[25] S. Xinghao, *et al.*, "Predicting bus real-time travel time basing on both gps and rfid data," *Procedia-Social and Behavioral Sciences*, vol. 96, pp. 2287–2299, 2013.

[26] J. Ramakrishnan, *et al.*, "Performance comparison of bus travel time prediction models across indian cities," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2672, no. 18-04897, 2018.

[27] C. Bai, *et al.*, "Dynamic bus travel time prediction models on road with multiple bus routes," *Computational intelligence and neuroscience*, vol. 2015, 2015.

[28] C. Xumei, *et al.*, "Brt vehicle travel time prediction based on svm and kalman filter," *Journal of Transportation Systems Engineering and Information Technology*, vol. 12, no. 4, pp. 29–34, 2012.

[29] B. Yu, *et al.*, "Hybrid model for prediction of bus arrival times at next station," *Journal of Advanced Transportation*, vol. 44, no. 3, 2010.

[30] M. Chen, *et al.*, "A dynamic bus-arrival time prediction model based on apc data," *Computer-Aided Civil and Infrastructure Engineering*, vol. 19, no. 5, pp. 364–376, 2004.

[31] M. Zaki, *et al.*, "Online bus arrival time prediction using hybrid neural network and kalman filter techniques," *International Journal of Modern Engineering Research*, vol. 3, no. 4, pp. 2035–2041, 2013.

[32] D. Liu, *et al.*, "Bustime: Which is the right prediction model for my bus arrival time?" in *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*. IEEE, 2020, pp. 180–185.

[33] C. C. Aggarwal, *et al.*, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory*. Springer, 2001, pp. 420–434.

[34] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.