Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm

Zuduo Zheng*, Dongcai Su

aCivil Engineering & Built Environment School, Science and Engineering Faculty, Queensland University of Technology, 2 George St GPO Box 2434 Brisbane Qld 4001 Australia

Abstract: To enhance the performance of the k-nearest neighbors approach in forecasting short-term traffic volume, this paper proposed and tested a two-step approach with the ability of forecasting multiple steps. In selecting k-nearest neighbors, a time constraint window is introduced, and then local minima of the distances between the state vectors are ranked to avoid overlappings among candidates. Moreover, to control extreme values’ undesirable impact, a novel algorithm with attractive analytical features is developed based on the principle component. The enhanced KNN method has been evaluated using the field data, and our comparison analysis shows that it outperformed the competing algorithms in most cases.

Keywords: Short-term traffic volume forecasting; KNN; Kalman filter; Principle component

* Corresponding author. Tel.: +61-07-3138-9989; fax: +61-07-3138-1170.
E-mail address: zuduo.zheng@qut.edu.au.
1. Introduction

As an indispensable component of intelligent transportation systems, short-term traffic volume forecasting (STTVF) has received enormous attentions over the past two decades. Consequently, many STTVF algorithms were developed using different approaches from various perspectives. Specifically, in terms of modeling, these algorithms are either parametric or non-parametric: the former explicitly and quantitatively formulates the relationship between the input and the output (the forecasted) via a parameterized function (model), while the latter is fully data driven and explores the implicit relationship between the forecasted data and input data without providing any well-defined function.

Implementing a parametric algorithm typically consists of two basic steps: estimating the parameters and forecasting the output by inputting new data into the calibrated model. Although a rich family of parametric STTVF algorithms with promising performances was developed in the literature (Ahmed and Cook 1979; Levin and Tsao 1980; Okutani and Stephanedes 1984; Hamed et al. 1995; Williams et al. 1998; Williams and Hoel 2003; Stathopoulos et al. 2003; Xie et al. 2007), they inherently face model calibration, validation, and computational challenges, which makes them difficult to be implemented in real-time transportation systems. For example, although good performance of seasonal autoregressive integrated moving average model (SARIMA) was frequently reported (Williams and Hoel 2003; Ghosh et al. 2005; Williams et al. 1998; Chung and Rosalion 2001), estimating the parameters of SARIMA is quite computationally demanding even in the case of univariate, as indicated in Smith et al. (2002).

On the other hand, non-parametric STTVF algorithms are also extensively studied and their good performances are often reported (Smith and Demetsky 1997; Zhang et al. 1998). Compared with parametric STTVF algorithms, main advantages of non-parametric algorithms include: intuitive formulation, totally data-driven and thus free of assumptions on data distribution, high flexibility and easy extendibility (Clark 2003). For example, \( k \)-nearest neighbor (KNN) algorithms can be easily extended to handle multivariate by simply adding data from multiple locations into the search space. More importantly, nonparametric algorithms are theoretically grounded. As an asymptotically optimal forecaster, when applied to a state space with \( m \) members, KNN approach will asymptotically be at least comparable to any \( m^{th} \) order parametric model (Smith et al. 2002). Motivated by this attractive property, there is a steady stream of refining and extending KNN in the literature. This paper is along this line.

Most existing KNN algorithms are single-step (Smith and Demetsky 1997; Smith et al. 2002; Davis and Nihan 1991), which has two main disadvantages: i) generating overlapping nearest neighbors when the method is extended to multiple-step forecasting as demonstrated later; ii) sensitive to noisy neighbors. To remedy these serious limitations, this study develops an enhanced KNN algorithm (i.e., KNN-LSPC)
with the ability of forecasting multiple steps. We have evaluated the algorithm’s performance using loop detector data. Our analysis shows that the enhanced \textit{KNN} algorithm outperformed the competing algorithms in most cases.

Note that for the convenience of discussion, this paper focuses on short-term volume forecasting. However, the algorithm can be easily adapted for forecasting other traffic flow measures (e.g., speed). The remaining of the paper is organized as follows. Section 2 defines the \textit{STTVF} problem, and then introduces \textit{KNN}; Section 3 presents the enhanced \textit{KNN} algorithm; Section 4 evaluates the enhanced \textit{KNN} algorithm’s performance; Finally, Section 5 summarizes the main findings and discusses future research.

2. Background

2.1 Problem Description

Without loss of generality, we define \textit{STTVF} as follows:

For a given traffic volume series \( \{\text{vol}^{(j)}(t), t_0 \leq t \leq t_c, 1 \leq j \leq m\} \), where \( t_0, t_c \) are the indices of the beginning and the current time intervals (note that for simplicity, time interval will be shortened as time unless it is otherwise stated), \( m \) is the number of locations (i.e., the number of loop detectors), and \( \text{vol}^{(j)}(t) \) represents the traffic volume collected from the \( j^{th} \) loop detector at time \( t \), the problem is to forecast volume \( L \) steps ahead for the target location (denoted as \( j^* \)). More specifically, our task is to forecast the following vector \( f(t) \):

\[
f(t) = \text{vol}^{(j^*)}(t_c + 1: t_c + L), \text{ with } f(t, i) = \text{vol}^{(j^*)}(t_c + i), 1 \leq i \leq L \text{ represents its } i^{th} \text{ element. Note that we use the notation } \text{vol}(a: b), b > a, a, b \in Z^+ \text{ to denote a sub-time series from time } a \text{ to time } b \text{ of a traffic volume time series. For notational simplicity, we denote } \text{vol}(t) = \text{vol}^{(j^*)}(t) \text{ herein.}\]

2.2 \textit{KNN} Algorithms

Like other data-driven approaches, \textit{KNN} algorithm’s performance is dependent on the representativeness and extensiveness of the data. The fundamental assumption of \textit{KNN} algorithms is that future states to be forecasted are more or less similar to a neighborhood of the past. Smith et al. (2002) provided an excellent review on \textit{KNN} forecasting algorithms.

A typical \textit{KNN} framework consists of three basic elements: defining the state vector; measuring distance between two state vectors; and forecasting future state vectors by utilizing a collection of \( k \)-nearest neighbors (candidates).

For \textit{STTVF}, a typical state vector can be defined as:

\[
x(t) = [\text{vol}(t - 2), \text{vol}(t - 1), \text{vol}(t)]
\]

The nearness of a state vector to another is commonly measured by the Euclidean
distance, according to which neighbors are ranked and selected. Out of $k$ neighbors that are nearest to the current state vector, future states can be forecasted using various methods. The simplest forecasting approach is to directly compute the average of the $k$ nearest neighbors, while more sophisticated approach in the literature generate forecasts by weighting the $k$ nearest neighbors according to their distances to the current state vector.

For illustration purpose, the KNN algorithms developed in Smith et al. (2002) are summarized below. These KNN algorithms are also used as part of the benchmark models in evaluating the new algorithm’s performance.

The KNN forecasting algorithms proposed in Smith et al. (2002) were designed to predict one step ahead and its state vector and forecasted vector at time $t$ are defined as in Equation (2) and (3), respectively.

$$x(t) = [vol(t - 2), vol(t - 1), vol(t), vol_{hist}(t), vol_{hist}(t + 1)]$$

$$f(t) = vol(t + 1)$$

where $vol_{hist}(t)$ is the historical average traffic volume of $vol(t)$; $vol_{hist}(t + 1)$ can be similarly interpreted. Compared with the state vector in Equation (1), the one in Equation (2) incorporates historical averages, which may help to find more similar neighbors and thus improve the forecasting accuracy (Mulhern and Caprara 1994). Additionally, Smith et al. (2002) reported that information contained in state vectors with only lagged observations was insufficient.

The $i^{th}$ selected candidate vector is defined as

$$Nb_i = \begin{bmatrix} vol^{(j^i)}(t^s_i - 2), vol^{(j^i)}(t^s_i - 1), vol^{(j^i)}(t^s_i), vol_{hist}^{(j^i)}(t^s_i), vol_{hist}^{(j^i)}(t^s_i + 1) \end{bmatrix},$$

where $j^i_s, t^s_i$ denote the detector id and time location.

For notational simplicity, we denote $vol_i(d) = vol^{(j^i)}(t^s_i + d), d \geq 0$, which corresponds to the forecast generation that is provided by $Nb_i$; similarly, $vol_{hist,i}(d)$ is the average historical value of $vol_i(d)$.

In Smith et al. (2002), six forecast generation methods were defined as in equations (4-9), respectively. Note that $Dist_j$ stand for the Euclidean distance between the $j^{th}$ neighbor and the current state vector.

$$\bar{vol}(t_c + 1) = \frac{1}{k} \sum_{j=1}^{k} vol_j(1)$$

$$\bar{vol}(t_c + 1) = \sum_{j=1}^{k} \frac{vol_j(1)}{Dist_j} \sqrt{\frac{1}{\sum_{j=1}^{k} 1}}$$

$$\bar{vol}(t_c + 1) = \left( \frac{1}{k} \right) \sum_{j=1}^{k} vol_j(1) \left( vol(t_c) / vol_j(0) \right)$$
\[ \hat{\text{vol}}(t_c + 1) = \frac{1}{k} \sum_{j=1}^{k} \text{vol}_j(1) \left( \frac{\text{vol}_{\text{hist}}(t_c + 1)}{\text{vol}_{\text{hist},j}(1)} \right) \] (7)

\[ \hat{\text{vol}}(t_c + 1) = \frac{1}{k} \sum_{j=1}^{k} \text{vol}_j(1) \left( \frac{\text{vol}(t_c)}{\text{vol}_j(0)} + \frac{\text{vol}_{\text{hist}}(t_c + 1)}{\text{vol}_{\text{hist},j}(1)} \right) / 2 \] (8)

\[ \hat{\text{vol}}(t_c + 1) = \frac{1}{k} \sum_{j=1}^{k} \frac{\text{vol}_j(1) \left( \frac{\text{vol}(t_c)}{\text{vol}_j(0)} + \frac{\text{vol}_{\text{hist}}(t_c + 1)}{\text{vol}_{\text{hist},j}(1)} \right) / 2}{\sum_{j=1}^{k} \frac{1}{\text{Dist}_j}} \] (9)

The forecast generation method in Equation (4) is the simple average of the outputs from the \( k \)-nearest neighbors. The methods presented in equations (5-9) are improvements of Equation (4) by incorporating the nearness of the neighbors and the historical traffic volume information.

To enable the KNN algorithms in Smith et al. (2002) to predict \( L \) steps ahead, we modify the state vector and the forecasted vector as shown in equations (10-11), and naturally extend the six forecast generation methods (i.e., equations (4-9)) to equations (12-17).

\[ x(t) = [\text{vol}(t - 2L: t), \text{vol}_{\text{hist}}(t: t + L)] \] (10)

\[ f(t) = \text{vol}(t + 1: t + L) \] (11)

\[ \hat{\text{vol}}(t_c + i) = \frac{1}{k} \sum_{j=1}^{k} \text{vol}_j(i) \] (12)

\[ \hat{\text{vol}}(t_c + i) = \frac{\sum_{j=1}^{k} \text{vol}_j(i) / \text{Dist}_j}{\sum_{j=1}^{k} \frac{1}{\text{Dist}_j}} \] (13)

\[ \hat{\text{vol}}(t_c + i) = (1/k) \sum_{j=1}^{k} \text{vol}_j(i) \left( \frac{\text{vol}(t_c)}{\text{vol}_j(0)} \right) \] (14)

\[ \hat{\text{vol}}(t_c + i) = (1/k) \sum_{j=1}^{k} \text{vol}_j(i) \left( \frac{\text{vol}_{\text{hist}}(t_c + i)}{\text{vol}_{\text{hist},j}(i)} \right) \] (15)

\[ \hat{\text{vol}}(t_c + i) = (1/k) \sum_{j=1}^{k} \text{vol}_j(i) \left( \frac{\text{vol}(t_c)}{\text{vol}_j(0)} + \frac{\text{vol}_{\text{hist}}(t_c + i)}{\text{vol}_{\text{hist},j}(i)} \right) / 2 \] (16)

\[ \hat{\text{vol}}(t_c + i) = \frac{\sum_{j=1}^{k} \text{vol}_j(i) \left( \frac{\text{vol}(t_c)}{\text{vol}_j(0)} + \frac{\text{vol}_{\text{hist}}(t_c + i)}{\text{vol}_{\text{hist},j}(i)} \right) / 2}{\sum_{j=1}^{k} \frac{1}{\text{Dist}_j}} \] (17)

where \( i = 1, 2, \ldots, L \) and other variables are similarly defined as in equations (4-9).

The execution procedure in Smith et al. (2002) is used to implement the extended KNN algorithms. For notational convenience, the algorithms corresponding to equations (12-17) are denoted as KNN1~KNN6, respectively.

3. The Enhanced KNN Algorithm: KNN-LSPC

In this section, the new algorithm based on KNN (i.e., KNN-LSPC) is proposed. To facilitate our discussion, we introduce some additional notations below.

Let \( q = [q_\lambda; q_+] \) be a compact vector combining the past and forecasted state vectors, where \( q_\lambda = \text{vol}(t_c - \lambda L + 1: t_c) \) is the current state vector, and \( q_+ = \text{vol}(t_c + \lambda L + 1: t_c) \) is the forecasting vector.
This is a draft version. Please cite it as: Zheng, Z., Su, D., 2014. Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm. Transportation Research Part C, accepted.

1: \( t_c + L \) is the unknown state vector that needs to be forecasted. Evidently, the length of \( q \) is \((\lambda + 1)L\). \( \text{Dist}^{(j)}(i) \) is the distance between the state vector \( \text{vol}^{(j)}(i - \lambda L + 1:i) \), \( i \in [t_0 + \lambda L, t_c] \) and the current state vector, which can be measured either by the Euclidean distance or the correlation coefficient distance, as mathematically defined below. Note that the notation \(||v||\) or \(||v||_2\) is the Euclidean norm of vector \( v \).

(1) Euclidean distance: \( \text{Dist}^{(j)}(i) = \sum_{n=1}^{\lambda L} (\text{vol}^{(j)}(i - \lambda L + n) - q_\lambda(n))^2 \);

(2) Correlation coefficient distance: \( \text{Dist}^{(j)}(i) = -\frac{\text{vol}^{(j)}(i - \lambda L + 1:i)\text{vol}^{(j)}(i - \lambda L + 1:i)^T q_\lambda}{\|\text{vol}^{(j)}(i - \lambda L + 1:i)\|_2\|q_\lambda\|_2} \).

Using the correlation coefficient distance can avoid selecting un-correlated neighbors. Moreover, the correlation coefficient distance is appealing to the regulation operations that are used in \( \text{KNN-LSPC} \) because it is scale invariant (\( \text{KNN-LSPC} \) is discussed in detail later). Our analysis also shows that the correlation coefficient distance performed better than the Euclidean distance when it is applied in \( \text{KNN-LSPC} \). As shown in Figure 1, the mean absolute percentage error (\( \text{MAPE} \)) of \( \text{KNN-LSPC} \) decreased about 2~3% when the correlation coefficient distance, instead of the Euclidean distance, was used. Therefore, the correlation coefficient distance is used in this study. The study site, data, and \( \text{MAPE} \) are discussed later.

![Figure 1](image-url)

3.1 Neighbor Selection

To ensure that the selected candidates share similar traffic characteristics, a time constraint is posed in the process of neighbor selection, by assuming that the starting time of a neighbor candidate should be reasonably close to the current time as defined below.

\( \text{Dist}^{(j)}(i) = +\infty, \text{if} |\text{time}(i) - \text{time}(t_c)| > TH \),

where \( \text{time}(i) = \text{mod}(i, \text{DAY}), 1 \leq TH < \text{DAY} \), \( \text{DAY} \) is the total time steps in one day, in our experiments, \( \text{DAY}=960 \) at 1.5min sampling rate; \( TH \) is a threshold of the radius of the time constraint window, which needs to be pre-determined based on observations. Figure 2 depicts the relationship between \( TH \) and the prediction accuracy of \( \text{KNN-LSPC} \). It indicates that \([15, 45]\) is a reasonable range of \( TH \). The smaller the radius is, the more restrictive the time constraint becomes, which leads to a smaller prediction error but fewer eligible neighbors. In our study, to balance the prediction error and number of eligible neighbors \( TH \) is set to 30 (45 minutes).

Figure 3 shows the prediction error of \( \text{KNN-LSPC} \) with and without such time constraint. This figure indicates that posing such time constraint can consistently improve the forecasting accuracy at all the loop detectors (i.e., \( \text{MAPE} \) decreased 1~3%).
Moreover, a typical traffic volume time series is significantly auto-correlated when the time lag is small. If \( k \) neighbors are selected directly from the original time series by ranking \( Dist^{(j)}(i) \), candidates that are significantly overlapping with each other may be obtained, which is intuitively not desirable. To avoid this, we choose \( k \) nearest candidates by ranking the local minima of \( Dist^{(j)}(i) \). As a typical example shown in Figure 4 and Figure 5, when the \( k \) nearest neighbors \((k=10\) in this example) were directly selected using \( Dist^{(j)}(i) \), the candidates were essentially grouped into 3 clusters (see Figure 4). The consequence of such overlapping is that the information in these 10 neighbors is approximately equivalent to that provided by three non-overlapping neighbors. That is, \( k \) was essentially decreased from 10 to 3 in this particular example. This dramatic decrease in \( k \) likely has significant negative impact on the algorithm’s performance as discussed later. In contrast, when the \( k \) nearest neighbors were selected using the local minima of \( Dist^{(j)}(i) \), the overlapping phenomenon disappeared (see Figure 5). This example demonstrates the effectiveness of using the local minima to select information-rich neighbors.

To further demonstrate the benefits of using local minima, \( MAPEs \) of \( KNN-LSPC \) with the local minima vs. without local minima are plotted in Figure 6, which clearly shows that the prediction error can be significantly reduced by using local minima. Note that it is possible to do more than one iteration of the local minima operation to achieve a better effect of diminishing the overlapping phenomenon.

### 3.2 Forecast Generation

From the procedure described above, \( k \) state vectors are selected as the nearest neighbors, based upon which the future state vector is forecasted. The \( k \) compact vectors combining the state vector and the forecasted vector are defined as below:

\[
c_i = [c_i \lambda; c_i +], 1 \leq i \leq k
\]
where \( c_{i\lambda} = vol(j_\lambda^T)(t_i^\lambda - \lambda L + 1: t_i^\lambda) \) is the \( i \)th nearest neighbor of \( q_\lambda^\lambda \); \( c_{i+} = vol(j_+^T)(t_i^+ + 1: t_i^+ + L) \) is the corresponding prediction of \( q_+ \).

To forecast \( q_+ \) based on the \( k \) predictions from the \( k \) nearest neighbors, a simple method is to take the average of \( \{c_{i+}\} \) as shown in Equation (18) (Smith et al. 2002).

\[
q_+ = \bar{c}_+ = \frac{1}{k} \sum_{i=1}^{k} c_{i+} \tag{18}
\]

However, an obvious shortcoming of this method is that the prediction of Equation (18) may be dominated by extreme values in \( c_{i+} \). This undesirable consequence is particularly severe when \( k \) is small, e.g., \( k<10 \). To suppress the negative effect of a few highly noisy candidates, a linearly sewing principle component method (LSPC) is proposed in this study to forecast \( q_+ \), as elaborated below.

Forecasting \( q_+ \) based on \( c_{i+} \) can be treated as two minimization problems, as described in equations (19-20), respectively. These two minimization problems are very similar except that two constraints are introduced in Equation (20).

\[
\begin{aligned}
\argmin_{q_+} \sum_{i=1}^{k} \|q_+ - c_{i+}\|_2^2 & \quad \text{subject to } \|q_+\| = 1, \text{ where } \hat{c}_{i+} = \frac{c_{i+}}{\|c_{i+}\|} \text{ is the normalized vector of } c_{i+}. \tag{19}
\end{aligned}
\]

\[
\begin{aligned}
\argmin_{q_+} \sum_{i=1}^{k} \|q_+ - \hat{c}_{i+}\|_2^2 & \quad \text{subject to } \|q_+\| = 1, \text{ where } \hat{c}_{i+} = \frac{c_{i+}}{\|c_{i+}\|} \text{ is the normalized vector of } c_{i+}. \tag{20}
\end{aligned}
\]

It is easy to verify that the straight average of \( c_{i+} \) (i.e., \( \bar{c}_+ \) in Equation (18)) and the principle component of regulated \( c_{i+} \) (denoted as \( p_+ \)) are the solutions of equations (19-20), respectively. However, the advantage of using Equation (20) is that the high energy noise’s impact can be effectively suppressed.

Furthermore, although we cannot expect \( \|q_+\| = 1 \) as indicated in the second constraint of Equation (20), it is reasonable to express \( q_+ \) as a linear transform of the principle component \( p_+ \) (See Equation (21)).

\[
q_+ = hp_+ + d \tag{21}
\]

Where \( h \) and \( d \) are linear sewing coefficients.

We have proved that the optimal linear transform coefficients \( h, d \) can be analytically solved by maximizing the sum of the negative correlation coefficient distances between \( q \) and \( \{c_i, 1 \leq i \leq k\} \) (see Equation (22)). The proof is in Appendix A.

\[
\argmax_{q_+} \sum_{i=1}^{k} q^T c_i / \|q\|_2 \|c_i\|_2 \tag{22}
\]
The LSPC algorithm described above is summarized below:

*Step 1: Select a set of compact candidate vectors \( \{c_i\} \), \( 1 \leq i \leq k \) based on the collection of nearest neighbors (the candidates).*

*Step 2: Compute the principle component vector \( p_+ \) of regulated \( \{c_i+\} \), \( 1 \leq i \leq k \).*

*Step 3: Compute the optimal linear sewing coefficients \( h, d \) according to equations (A.8) and (A.9) respectively.*

*Step 4: Solve \( q_+ \) according to Equation (21).*

In summary, LSPC has two advantages. First, the principle component of \( \{c_i+\} \) can effectively suppress the dominant influence of the extreme values in \( c_i+ \). Meanwhile, the optimal linear transform coefficients can be solved analytically.

### 4. Performance Evaluation

#### 4.1 Study Site

The data that were used for evaluating the KNN algorithm enhanced by LSPC (KNN-LSPC) were collected for every 1.5 min from seven loop detectors on Alexandras Ave. (a signalized arterial with 3 lanes per direction) in the centre of Athens. The loop detector locations are approximately shown in Figure 7, where detectors L101, L103, L106, and L108 locate in the westbound direction while detectors L102, L104, and L107 locate in the eastbound. Although occupancy is also available, only volume is considered in this study. The data were collected in April, 2000, and halved to the training set (i.e., data from the 1st to 15th of April) and the validation set (i.e., data from the 16th to 30th of April).

As demonstrated in Stathopoulos et al. (2003), despite the presence of signalized intersections, the study site is suitable for testing STTVF algorithms because the roadway retained a relative smooth operation in the data collection period. Furthermore, most existing STTVF algorithms target freeway traffic, and it is generally more challenging to predict short-term traffic characteristics in signalized urban arterials. Thus, this study site is used in developing and testing KNN-LSPC algorithm.

Strong variability of traffic flow data that are collected at short time intervals makes forecasting extremely difficult. As a result, researchers often aggregate the raw data (using some filtering methods) before applying a prediction algorithm (Okutani and Stephanedes 1984; Vlahogianni et al. 2004). In addition, isolated missing values, which are not rare in traffic data, can also be naturally interpolated during filtering. Thus, in our study, we have applied a simple smoothing technique (i.e., a median filter with the window size = 15 minutes), without using any sophisticated filtering methods because the focus of our study is STTVF algorithms. Note that filtering the data at the
pre-processing stage does not have any impact on the analyses and conclusions in this study because the same filtered data were used for all the algorithms that are evaluated.

Traffic conditions on weekdays (and non public holidays) are typically different from those on weekends (and public holidays). Ignoring such difference has negative impact on forecasting algorithms’ performances. Thus, we have classified the data into two groups (i.e., normal days, weekends or public holidays) before applying any forecasting methods.

Furthermore, to account for traffic characteristics differences in different time periods of the day, Stathopoulos et al. (2003) partitioned one day into 6 time periods (period 1: midnight - 6:30 am; period 2: 6:30 -10:00; period 3: 10:00 -13:30; period 4:13:30 -17:00; period 5: 17:00 - 20:30; and period 6: 20:30 - midnight). Analyses in Stathopoulos et al. (2003) indicated that such partitioning was statistically acceptable. Thus, our study has adopted the same strategy.

Figure 7

4.2 Performance Comparison

To rigorously evaluate $KNN-LSPC$’s performance, the six $KNN$ algorithms developed in Smith et al. (2002) are treated as benchmark algorithms. Although this study focuses on non-parametric algorithms, more specifically, $KNN$ algorithms, we have also compared $KNN-LSPC$’s performance with those of two Kalman filter algorithms (one is univariate and denoted as $Kalman-S$; one is multivariate and denoted as $Kalman-M$) because algorithms based on the Kalman filter theory are widely used in $STTVF$ and frequently reported as out-performing other algorithms. For a brief introduction to the Kalman filter theory and for detail of the Kalman filter algorithms that are used in our comparison analysis, see Appendix B.

In Kalman filtering, it is reasonable to set the length $n$ of the state vector proportional to the prediction length (Okutani and Stephanedes 1984; Xie et al. 2007). In this paper we set $n = L$. Accordingly, we set $\lambda = 1$ in $KNN-LSPC$ to ensure a fair comparison with the Kalman filter algorithms. $L$ is the prediction horizon and set to 64 (i.e., $64 \times 1.5 \text{ min} = 96 \text{ min}$).

Meanwhile, $KNN$ algorithms can be easily extended from univariate to multivariate by expanding the searching space $vol^{(j)}(t)$ from a single loop detector ($j=j^*$) to multiple loop detectors ($1 \leq j \leq m$). In this paper, we report the multivariate $KNN$ because it uses the training data more sufficiently.

In implementing $KNN$ algorithms, an important issue is to determine an appropriate $k$, the number of selected neighbors. Our analysis shows that when $k$ was less than 10, different $k$ values had significant impact on $KNN-LSPC$’s performance. However, when $k$ was above 10, its impact on the algorithm’s performance dramatically
decreased and became negligible (see Figure 8). In our study, $k=30$.

The prediction error is measured by the relative absolute percentage error ($APE$):

$$APE(t) = \frac{\sum_{i=1}^{L} |f(t,i) - Vol(t+i)|}{\sum_{i=1}^{L} |Vol(t+i)|}$$

$MAPE$ and $MDAPE$ represent the mean and the median of $\{APE(t)\}$, $t \in P$, where $P$ denotes the set of prediction time points randomly sampled from the training time horizon. The size of $P$ is 1440, which is $1/10$ of the total training size. $MAPE$ and $MDAPE$ are mathematically defined below:

$$MAPE = \frac{1}{|P|} \sum_{i=1}^{|P|} s_APE(i)$$

$$MDAPE = s_APE \left( \frac{|P|+1}{2} \right)$$

Where $s_APE$ a sorted array of elements in $\{APE(t), t \in P\}$ in non-decreasing order. Meanwhile, the Friedman and Wilcoxon signed-rank tests were implemented to statistically compare these algorithms’ performances, as in Smith et al. (2002). Result of the comparison analysis at L101 is summarized in Table 1, which lists the performances of nine different algorithms at that location. A grey line in the table indicates that the algorithm’s performance was significantly better than those of the algorithms in the following rows at the 0.05 significance level either by the Friedman test or by the Wilcoxon signed-rank test. Note that the mean rank of each algorithm does not necessarily coincide with $MAPE$ and $MDAPE$. Comparison analyses at other locations are provided in Appendix C.

Table 1

Several observations are obtained from Table 1 and tables (C.1 – C.6):

1) $KNN-LSPC$ consistently outperformed most of the competing algorithms, and was always in the best-performance group at all the detector locations of the study site. More specifically, among nine algorithms that were evaluated in this paper, $KNN-LSPC$’s performance was the best at L101, L103, L106; $KNN-LSPC$ and $Kalman-S$ performed equally well at L102 and L104; $KNN-LSPC$, $Kalman-S$, and $Kalman-M$ performed equally well at L107.
2) For KNN-LSPC, its MAPE was in a range of [11.3, 17.1] and MDAPE in a range of [7.4, 13.6]. The range of MAPE is consistent with what was reported in Stathopoulos et al. (2003) and reflects the challenges of accurately predicting short-term volume for signalized urban arterials.

3) All the tested algorithms, including KNN-LSPC, performed differently at different locations, which highlights the importance of data quality.

5. Conclusions and Discussion

This paper proposed a two-step approach to enhance KNN’s performance in forecasting short-term traffic volume. First, at the neighbor selection stage, the time constraint is introduced, and local minima is utilized to diminish overlappings among the selected candidates. Then, at the forecast generation stage, a novel algorithm (i.e., KNN-LSPC) based on the principle component has been developed to effectively suppress the undesirable and dominant impact of extreme values. Moreover, we proved that the parameters in KNN-LSPC can be analytically optimized, which provides a theoretical justification for the good performance of this enhanced KNN method.

KNN-LSPC was evaluated using the field data and compared with the benchmark models. In most cases, KNN-LSPC outperformed the eight competing algorithms (i.e., six KNN algorithms and two Kalman filter algorithms).

Compared with the Kalman filter algorithms’ performances, the improvement of KNN-LSPC was not surprisingly big. Nevertheless, this result demonstrates the effectiveness of the strategies that we have adopted to enhance KNN and underscores KNN-LSPC’s good performance, especially when we keep in mind that 1) the Kalman filter algorithms are parametric and often reported to outperform KNN algorithms; 2) as being non-parametric, KNN methods (including KNN-LSPC) generally have (at least) two major advantages over the Kalman filter algorithms: totally data-driven and thus free of assumptions on data distribution; high flexibility and easy extendibility (Clark 2003). In contrast, the Kalman filter algorithms generally assume the state noise and measurement noise (see Equation (B.4)) to be Gaussian. Moreover, the Kalman model's complexity increases with the increase of the number of loop detectors or of the prediction length, which makes estimating parameters more difficult, and also make it more challenging to reach a bias-variance trade-off (Hastie et al. 2008).

Based on our analyses and the literature (Smith, B. L. et al. 2002), when the query sub-time series and the prediction sub-time series are single points, the traditional KNN (e.g., KNN1-KNN6 in this paper) can perform well; when they are multiple points, KNN-LSPC should be preferred since it have advantages of avoiding overlapping candidates and being insensitive to extreme values; and when the state
noise and measurement noise are approximately Gaussian and the complexity is reasonable, the Kalman method can perform well.

Theoretically, KNN-LSPC cannot guarantee the existence of \( k \) local minima if \( k \) is large enough. However, it should not be an issue in practice because KNN-LSPC is not sensitive to \( k \) when \( k \) is larger than 10. As shown in Figure 8, when \( k \) is increased from 10 to 40, MAPE only decreased about 1\%. The recommended \( k \) is any value between 10 and 40, which can be easily satisfied in practice because rich archived data are typically available. For example, if one neighbor is selected from each day (the most conservative case), to get \( k=30 \) neighbors we only need 30 days of historical data.

To further improve KNN-LSPC’s prediction accuracy, the following issues can be investigated: incorporating the distances as weights as in Smith et al. (2002); improving the state vector’s information quality by considering historical averages or by incorporating speed and/or occupancy; making \( TH \) time-dependent (or traffic state dependent). Finally, KNN-LSPC has been developed and tested in the context of an urban street. Robustness of KNN-LSPC needs to be further tested using more field data from different road types. For example, KNN-LSPC can be extended to freeways by integrating bottleneck analysis (Zheng et al. 2011 a; Zheng et al. 2011b; Zheng and Washington 201). This work is ongoing.

Acknowledgement: The authors were grateful for three anonymous reviewers’ detailed, constructive, and insightful comments, which have significantly improved this paper’s quality. This research was partially funded by the Early Career Academic Development (ECARD) program at Queensland University of Technology.

References


vehicular traffic flow in Dublin. Transportation Research Board Annual Meeting, Washington D. C.


flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of Transportation Engineering*, 129(6), 664 - 672.


Appendix A: Solving the Optimal Linear Transform Coefficients of KNN-LSPC

Without loss of generality, we regulate \( c_i \) such that \( \|c_i\| = 1 \) and equivalently, we maximize the square of Equation (22). Thus, the optimization objective is reformulated as in Equation (A.1).

\[
\arg\max_{q_+} \frac{(q^T \sum_{i=1}^k c_i)^2}{q^T q} = \arg\max_{q_+} \frac{(q_+^T \sum_{i=1}^k c_i + q_+^T \sum_{i=1}^k c_i^+)^2}{q_+^T q_+ + q_+^T q_+^+} = \arg\max_{q_+} \frac{(A+q_+^T b)^2}{C+q_+^T q_+^+}
\]

(A.1)

Where \( A = q_+^T \sum_{i=1}^k c_i \), \( b = \sum_{i=1}^k c_{i+} \), \( C = q_+^T q_+ \) are constant.

We also reformulate the linear transform in Equation (21) separately as:

\[
q_+ = \alpha \hat{p}_+ ; \quad \hat{p}_+ = p_+ + \beta
\]

(A.2)

Now the remaining question is how to calculate the optimal value of the scaling factor \( \alpha \) and the translation factor \( \beta \) in (A.2).

By substituting \( q_+ = \alpha \hat{p}_+ \) into Equation (A.1) we have

\[
\arg\max_{\alpha} \frac{(A+B\alpha)^2}{C+D\alpha^2} = \arg\max_{\alpha} \frac{(E+F\beta)^2}{G+L\beta^2}
\]

(A.3)

where \( D = \|\hat{p}_+\|^2 \) and \( B = b^T \hat{p}_+ \).

For the convenience of illustration, we tentatively assume that \( B, D \) are known (in practice, the translation factor \( \beta \) is calculated prior to estimating \( \alpha \); how to determine \( \beta \) is described later). Then the optimization problem in Equation (A.3) is equivalent to the one defined in Equation (A.4).

\[
\arg\max_{\alpha} u \quad \text{subject to } (A + B\alpha)^2 - u(C + D\alpha^2) = 0 \ ; \ \alpha \text{ is real.}
\]

Solving Equation (A.4) we have:

\[
u = \frac{B^2}{D} + \frac{A^2}{C} ; \quad \alpha = B/AD \quad \text{(A.5)}
\]

where \( A = q_+^T \sum_{i=1}^k c_i, B = b^T \hat{p}_+, C = q_+^T q_+, D = \|\hat{p}_+\|^2 \)

Now \( \beta \) is still undecided. The optimal \( \beta \) can be obtained by maximizing \( u \) in Equation (A.5), which is equivalent to maximizing \( \frac{B^2}{D} \) with respect to \( \beta \) since the second term \( \frac{A^2}{C} \) is constant (see Equation (A.6)).

\[
\arg\max_{\beta} \frac{B^2}{D} = \arg\max_{\beta} \frac{(p_+^T \sum_{i=1}^k c_{i+} + \beta \sum_{i=1}^k c_{i+}^T c_{i+})^2}{p_+^T p_+ + 2\beta p_+^T p_+^+ + L\beta^2} = \arg\max_{\beta} \frac{(E+F\beta)^2}{1+\beta+L\beta^2}
\]

(A.6)

Where \( E = p_+^T \sum_{i=1}^k c_{i+} ; \ F = \sum_{i=1}^k c_{i+}^T c_{i+} ; \ G = 2 \times 1^T p_+ ; \ L \) is constant, and denotes the dimensions of \( p_+ \) (also the forecast steps).
Using the same reasoning as solving $\alpha$, we have

$$
\tau = \frac{B^2}{D} = \frac{4(-EFG + F^2 + LE^2)}{4L - G^2} \quad \beta = \frac{-2EF + \tau G}{2F^2 - 2\tau L}
$$

(A.7)

From Equation (21) and Equation (A.2), it is easy to show that

$$
\begin{align*}
\eta &= \alpha \\
\delta &= \alpha \beta
\end{align*}
$$

(A.8) \quad (A.9)

**Appendix B: The Kalman Filter Algorithms**

The Kalman filter approach is commonly used in STTVF and widely regarded as promising. Okutani and Stephanedes (1984) used the Kalman filtering to predict traffic volume and developed an extended Kalman filter to predict traffic diversion at freeway on/off-ramps. The superiority of multivariate Kalman filtering over an ARIMA formulation was reported in Stathopoulos et al. (2003). Compared with other forecasting techniques, Kalman filtering has two major advantages. First, unlike ARIMA, the Kalman filter can be easily extended from univariate to multivariate (Durbin 2000), which is further discussed later. Furthermore, as a state space model, the Kalman filtering has the Markovian nature, which enables it to efficiently handle complex models without substantially increasing computational cost (Stathopoulos et al. 2003).

An introduction of Kalman filter theory is briefly described here. For a complete discussion of the theory, refer to Haykin (2001) and Welch and Bishop (2004).

The Kalman system is expressed as:

$$
\begin{align*}
\dot{x}(t + 1) &= F_t x(t) + G_t u(t) + v(t + 1) \\
y(t) &= H_t x(t) + w(t)
\end{align*}
$$

(B.1)

The first equation in (A.1) is called the state equation, and the second is called the measurement equation. Where $v(t) \sim WN(0, Q_t)$ and $w(t) \sim WN(0, R_t)$ are independent white noise that is drawn from normal distributions; $x(t)$ is the state vector, which is assumed to be invisible (unknown); $y(t)$ is the measured vector (known); $u(t)$ is the input vector (known); and $F_t$, $G_t$, and $H_t$ are system parameter matrices (known). Let $x(t|t)$ and $x(t + 1|t)$ represent the prior and posterior state vector estimates at time $t$, respectively. Their covariance matrices are defined as below:

$$
\begin{align*}
p(t) &= E \left[ (x(t) - x(t|t))(x(t) - x(t|t))^T \right] \\
p(t + 1|t) &= E \left[ (x(t) - x(t + 1|t))(x(t) - x(t + 1|t))^T \right]
\end{align*}
$$
The Kalman filter uses a predictor-corrector algorithm to estimate $x(t)$ recursively as elaborated in Procedure (1).

**Procedure (1): estimating $x(t)$**

**Step 1:** Initializing: $\tau = 1$, $x(\tau | \tau)$, $p(\tau)$;

**Step 2:** Prediction: $x(\tau + 1 | \tau) = F_{\tau} x(\tau | \tau) + G_{\tau} u(\tau)$;

$P(\tau + 1 | \tau) = F_{\tau} P(\tau) F_{\tau}^T + Q_{\tau}$;

**Step 3:** Correction: $S = H_{\tau} P(\tau + 1 | \tau) H_{\tau}^T + R_{\tau}$;

$W = P(\tau + 1 | \tau) H_{\tau} S^{-1}$;

$\Delta x = W (y(\tau + 1) - H_{\tau} x(\tau + 1 | \tau))$;

**Step 4:** Updating: $x(\tau + 1 | \tau + 1) = x(\tau + 1 | \tau) + \Delta x$;

$P(\tau + 1) = P(\tau + 1 | \tau) - W^T S W$;

Traffic volume time series are often described through an auto-regression model as stated in Equation (B.2) (Okutani and Stephanedes 1984; Lan 2001).

$$vol^{(j)}(t) = \sum_{i=1}^{n} \theta^{(j)}(t, i) vol^{(j)}(t - i) \quad (B.2)$$

Where $vol^{(j)}(t)$ denotes the volume recorded at time $t$ on detector $j$. $vol^{(j)}(t), 1 \leq t \leq t_c$ are known to us, and $vol^{(j)}(t), t > t_c$ needs to be forecasted; $\theta^{(j)}(t) = [\theta^{(j)}(t, 1), \ldots, \theta^{(j)}(t, n)]^T$ is a $n$-dimensional coefficient vector of auto-correlation model, and used as the state vector in the Kalman system; and $n$ is the degree of the model.

It is intuitive to consider multiple detectors in the auto-correlation model because volumes at different loop detectors along the same road are usually highly correlated with each other due to the flow conservation (Stathopoulos et al. 2003). Equation (B.2), which describes the auto-correlation model with respect to a single detector, can be easily extended to multiple detectors, as defined in Equation (B.3).

$$vol^{(j)}(t) = \sum_{j=1}^{m} \sum_{i=1}^{n} \theta^{(j)}(t, i) vol^{(j)}(t - i) \quad (B.3)$$

Where $j = 1, 2, \ldots, m$ are the detectors that are considered to be relevant to the forecasted traffic volume.

After establishing the auto-correlation model, the next step is to formulate equations (B.2 – B.3) into a Kalman system,

In STTVF, the Kalman system is often simplified as in Equation (B.4) by assuming $u(t) = 0$ and $F_\tau = I$.

$$\begin{align*}
\{ x(t + 1) &= x(t) + v(t + 1) \\
y(t) &= H_\tau x(t) + w(t)
\end{align*} \quad (B.4)$$
In case of a single detector, \( x(t) = \theta^{(j)}(t) \) denotes the \( n \)-dimensional state vector; \( v(t) \sim WN(0, Q_t) \) is \( n \)-dimensional white noise; \( y(t) = vol^{(j^*)}(t) \) is a scalar denoting the measured vector (volume at time \( t \) on detector \( j^* \)); \( H_t = \tilde{v}ol_t^{(j^*)} = [\tilde{v}ol_t^{(j^*)}(t - l, \ldots, \tilde{v}ol_t^{(j^*)}(t-n)] \) is the known \( 1 \times n \) matrix constructed by the most recent volume sub-time series; \( w(t) \sim WN(0, R_t) \) is 1-dimensional white noise.

In case of multiple detectors, \( x(t) = [\theta^{(1)}(t), \theta^{(2)}(t), \ldots, \theta^{(m)}(t)] \); \( v(t) \sim WN(0, Q_t) \) are \( mn \)-dimensional vectors; \( H_t = [\tilde{v}ol_t^{(1)}, \tilde{v}ol_t^{(2)}, \ldots, \tilde{v}ol_t^{(m)}] \) is a \( 1 \times mn \) matrix that concatenates the most recent volume time sub-time series of length \( n \) from all \( m \) loops; \( y(t) \) and \( w(t) \) are the same as in case of a single loop.

The Kalman filter algorithms that were implemented in this study are not directly replicated from any single study because the information in the literature is not sufficient for enabling us to exactly repeat any existing Kalman filter algorithm. Instead, typical features of the Kalman algorithms in the literature are integrated and executed in this paper, as elaborated below.

As in Stathopoulos et al. (2003), we partitioned one day into 6 time periods (period 1: midnight - 6:30 am; period 2: 6:30 -10:00; period 3: 10:00 -13:30; period 4:13:30 - 17:00;  period 5: 17:00 - 20:30; and period 6: 20:30 - midnight) and assumed \( Q_t, R_t \) to be constant in each period. So there were 12 different parameters to be estimated, and they are denoted as \( \{Q_{[i]}, R_{[i]}, i = 1, 2, \cdots, 6\} \). The parameters were initialized by setting \( Q_{[i]} = R_{[i]} = I \), and estimated via the EM algorithm as in Xie et al. (2007).

After the parameters in (B.4) were defined, we initialized \( x(1|1) = \left[ \frac{1}{n}, \ldots, \frac{1}{n} \right] \) and \( p(1) = I \), where \( n \) is the dimension of \( x(t) \), \( I \) is the \( n \times n \) identity matrix. Then \( x(t|t), t = 2, \cdots, t_c \) were calculated via Procedure (1). Finally, the prediction after \( t_c \) was obtained by following Procedure (2). Note that since the measurement \( y(t), t > t_c \) is unavailable, we applied the Kalman filter process without correction (i.e., \( v(t) = w(t) = 0, t > t_c \)) as in Okutani and Stephanedes (1984), Xie et al. (2007).

**Procedure (2): forecasting \( y(t + i), 1 \leq i \leq L \)**

**Step 1:** Estimating the state vector without correction (assuming \( v(t) = 0 \)):
\[ x(t + 1|t + 1) = x(t|t); \]

**Step 2:** Forecasting one step ahead: \( y(t + 1) = H_t x(t + 1|t + 1); \)

**Step 3:** Updating: \( t = t + 1, H_t = [y(t), y(t - 1), \cdots, y(t - n + 1)]; \)

**Step 4:** Returning to Step 1 until the whole prediction horizon is covered;

To ensure that the forecasted \( y(t) \) is time stationary at Step 2 of Procedure (2), it is necessary and sufficient to restrict \( E[y(t)] = 0 \) for an arbitrary \( t \). To achieve this, we
assumed the traffic volume signal is weekly stationary and formulated $y(t)$ as in Equation (B.5) (Okutani and Stephanedes1984). The forecasted traffic volume is then recovered by using Equation (B.6):

$$y(t) = \text{vol}^{(j)}(t) - \text{vol}^{-{(j)}}(t) \quad \text{(B.5)}$$

$$\text{vol}^{(j)}(t) = y(t) + \text{vol}^{-{(j)}}(t) \quad \text{(B.6)}$$

where $\text{vol}^{-(j)}(t)$ denotes the traffic volume one week before time $t$ at detector $j$.

The process of the Kalman filter algorithm is summarized below:

Step 1: Choosing the Kalman model, univariate model (B.2) or multivariate model (B.3);

Step 2: Computing the measured state vector $y(t)$ using (B.5);

Step 3: Estimating the parameters \{\text{\$Q_{i|i-1}$, \text{\$R_{i|i-1}$, $i = 1, 2, \ldots, 6$ in the Kalman filter system (B.4);}}\

Step 4: Estimating the state vector $x(t|t), 1 \leq t \leq t_c$ using Procedure (1);

Step 5: Predicting $y(t), t_c < t \leq t_c + L$ using Procedure (2);

Step 6: Recovering the forecasted traffic volume via Equation (B.6).

Appendix C: The Comparison Results

Table C.1

Table C.2

Table C.3

Table C.4

Table C.5

Table C.6
This is a draft version. Please cite it as: Zheng, Z., Su, D., 2014. Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm. Transportation Research Part C, accepted.

Figure 1 MAPEs of KNN-LSPC using the correlation coefficient distance vs. using the Euclidean distance

Figure 2 MAPEs of KNN-LSPC using different radius of the time constraint window
This is a draft version. Please cite it as: Zheng, Z., Su, D., 2014. Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm. Transportation Research Part C, accepted.

Figure 3 MAPEs of KNN-LSPC with and without the time constraint; numbers 1~4 denote L101~L104, and numbers 5~7 denote L106~L108.

Figure 4 Candidates selected without using local minima
This is a draft version. Please cite it as: Zheng, Z., Su, D., 2014. Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm. Transportation Research Part C, accepted.

Figure 5 Candidates selected using local minima

Figure 6 MAPEs of KNN-LSPC with and without local minima; numbers 1~4 denote L101~L104, and numbers 5~7 denote L106~L108.
This is a draft version. Please cite it as: Zheng, Z., Su, D., 2014. Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm. Transportation Research Part C, accepted.

Figure 7 The loop detector locations on the Alexandras Ave., Athens (Karlaftis, 2012)

Figure 8 $k$'s impact on MAPEs of KNN-LSPC
Table 1 Comparison results at L101

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean rank</th>
<th>MAPE(%)</th>
<th>MDAPE(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN-LSPC</td>
<td>3.9057</td>
<td>11.3162</td>
<td>7.4236</td>
</tr>
<tr>
<td>KNN1</td>
<td>4.4363</td>
<td>20.5167</td>
<td>8.9526</td>
</tr>
<tr>
<td>KNN2</td>
<td>4.6214</td>
<td>20.5726</td>
<td>8.9524</td>
</tr>
<tr>
<td>KNN5</td>
<td>4.8733</td>
<td>20.7652</td>
<td>9.2958</td>
</tr>
<tr>
<td>Kalman-S</td>
<td>4.9261</td>
<td>13.2792</td>
<td>9.5608</td>
</tr>
<tr>
<td>KNN6</td>
<td>5.0063</td>
<td>20.8247</td>
<td>9.3715</td>
</tr>
<tr>
<td>KNN3</td>
<td>5.5074</td>
<td>21.3646</td>
<td>9.8245</td>
</tr>
<tr>
<td>KNN4</td>
<td>5.6777</td>
<td>21.3456</td>
<td>9.7099</td>
</tr>
<tr>
<td>Kalman-M</td>
<td>6.0457</td>
<td>22.2105</td>
<td>13.2754</td>
</tr>
</tbody>
</table>

Note: * univariate; # multivariate

Table C.1 The comparison result at L102

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean rank</th>
<th>MAPE(%)</th>
<th>MDAPE(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kalman-S</td>
<td>4.3659</td>
<td>11.1394</td>
<td>7.7609</td>
</tr>
<tr>
<td>KNN-LSPC</td>
<td>4.4426</td>
<td>11.3231</td>
<td>7.8605</td>
</tr>
<tr>
<td>KNN1</td>
<td>4.6756</td>
<td>17.8214</td>
<td>8.5810</td>
</tr>
<tr>
<td>KNN2</td>
<td>4.9648</td>
<td>17.9436</td>
<td>8.6161</td>
</tr>
<tr>
<td>KNN5</td>
<td>4.9909</td>
<td>18.0065</td>
<td>8.8123</td>
</tr>
<tr>
<td>Kalman-M</td>
<td>5.0471</td>
<td>13.0212</td>
<td>8.9113</td>
</tr>
<tr>
<td>KNN6</td>
<td>5.2027</td>
<td>18.1123</td>
<td>8.8355</td>
</tr>
<tr>
<td>KNN4</td>
<td>5.4849</td>
<td>18.5860</td>
<td>8.6705</td>
</tr>
<tr>
<td>KNN3</td>
<td>5.8255</td>
<td>18.3040</td>
<td>9.4714</td>
</tr>
</tbody>
</table>

Table C.2 The comparison result at L103

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean rank</th>
<th>MAPE(%)</th>
<th>MDAPE(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN-LSPC</td>
<td>4.2801</td>
<td>17.0956</td>
<td>13.5716</td>
</tr>
<tr>
<td>KNN1</td>
<td>4.4821</td>
<td>25.3883</td>
<td>14.0573</td>
</tr>
<tr>
<td>KNN2</td>
<td>4.6341</td>
<td>25.5136</td>
<td>13.9946</td>
</tr>
<tr>
<td>Kalman-S</td>
<td>4.8557</td>
<td>18.5480</td>
<td>13.7603</td>
</tr>
<tr>
<td>KNN5</td>
<td>4.9796</td>
<td>25.9415</td>
<td>14.5382</td>
</tr>
<tr>
<td>KNN6</td>
<td>5.1203</td>
<td>26.0559</td>
<td>14.5237</td>
</tr>
<tr>
<td>KNN4</td>
<td>5.3399</td>
<td>26.4313</td>
<td>14.3169</td>
</tr>
<tr>
<td>Kalman-M</td>
<td>5.4110</td>
<td>24.0090</td>
<td>15.9508</td>
</tr>
<tr>
<td>KNN3</td>
<td>5.8973</td>
<td>26.7243</td>
<td>15.8170</td>
</tr>
</tbody>
</table>
This is a draft version. Please cite it as: Zheng, Z., Su, D., 2014. Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm. Transportation Research Part C, accepted.

Table C.3 The comparison result at L104

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean rank</th>
<th>MAPE(%)</th>
<th>MDAPE(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN_LSPC</td>
<td>4.4525</td>
<td>15.0701</td>
<td>11.8055</td>
</tr>
<tr>
<td>Kalman-S</td>
<td>4.5236</td>
<td>16.0364</td>
<td>12.1954</td>
</tr>
<tr>
<td>KNN1</td>
<td>4.6756</td>
<td>20.6770</td>
<td>13.4383</td>
</tr>
<tr>
<td>KNN2</td>
<td>4.8543</td>
<td>20.7366</td>
<td>13.5205</td>
</tr>
<tr>
<td>KNN5</td>
<td>4.9726</td>
<td>20.8377</td>
<td>13.9144</td>
</tr>
<tr>
<td>KNN6</td>
<td>5.1112</td>
<td>20.8821</td>
<td>13.9633</td>
</tr>
<tr>
<td>Kalman-M</td>
<td>5.2562</td>
<td>20.1129</td>
<td>15.4484</td>
</tr>
<tr>
<td>KNN4</td>
<td>5.4145</td>
<td>21.3925</td>
<td>13.6255</td>
</tr>
<tr>
<td>KNN3</td>
<td>5.7396</td>
<td>21.2907</td>
<td>14.6306</td>
</tr>
</tbody>
</table>

Table C.4 The comparison result at L106

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean rank</th>
<th>MAPE(%)</th>
<th>MDAPE(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN_LSPC</td>
<td>3.8438</td>
<td>12.2859</td>
<td>8.5078</td>
</tr>
<tr>
<td>Kalman-S</td>
<td>4.5559</td>
<td>13.3002</td>
<td>10.1032</td>
</tr>
<tr>
<td>KNN1</td>
<td>4.6439</td>
<td>17.2558</td>
<td>10.6471</td>
</tr>
<tr>
<td>KNN2</td>
<td>4.8234</td>
<td>17.3157</td>
<td>10.5607</td>
</tr>
<tr>
<td>KNN5</td>
<td>5.0859</td>
<td>17.6947</td>
<td>10.9161</td>
</tr>
<tr>
<td>Kalman-M</td>
<td>5.2752</td>
<td>17.0305</td>
<td>11.8765</td>
</tr>
<tr>
<td>KNN6</td>
<td>5.3019</td>
<td>17.7705</td>
<td>11.0371</td>
</tr>
<tr>
<td>KNN4</td>
<td>5.7171</td>
<td>18.5283</td>
<td>11.3103</td>
</tr>
<tr>
<td>KNN3</td>
<td>5.7530</td>
<td>17.9142</td>
<td>11.6307</td>
</tr>
</tbody>
</table>

Table C.5 The comparison result at L107

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean rank</th>
<th>MAPE(%)</th>
<th>MDAPE(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kalman-M</td>
<td>4.4405</td>
<td>11.8601</td>
<td>8.9680</td>
</tr>
<tr>
<td>KNN_LSPC</td>
<td>4.4567</td>
<td>11.8049</td>
<td>8.4861</td>
</tr>
<tr>
<td>Kalman-S</td>
<td>4.6974</td>
<td>11.9045</td>
<td>8.9668</td>
</tr>
<tr>
<td>KNN1</td>
<td>4.7530</td>
<td>19.1807</td>
<td>9.3939</td>
</tr>
<tr>
<td>KNN5</td>
<td>5.0042</td>
<td>19.1719</td>
<td>9.7074</td>
</tr>
<tr>
<td>KNN2</td>
<td>5.0866</td>
<td>19.3229</td>
<td>9.4560</td>
</tr>
<tr>
<td>KNN6</td>
<td>5.2794</td>
<td>19.2954</td>
<td>9.7451</td>
</tr>
<tr>
<td>KNN4</td>
<td>5.3913</td>
<td>19.7233</td>
<td>9.8273</td>
</tr>
<tr>
<td>KNN3</td>
<td>5.8909</td>
<td>19.4271</td>
<td>10.2597</td>
</tr>
</tbody>
</table>
Table C.6 The comparison result at L108

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean rank</th>
<th>MAPE(%)</th>
<th>MDAPE(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN1</td>
<td>4.3892</td>
<td>18.7215</td>
<td>10.1853</td>
</tr>
<tr>
<td>KNN2</td>
<td>4.4821</td>
<td>18.7538</td>
<td>10.1375</td>
</tr>
<tr>
<td>KNN_LSPC</td>
<td>4.6355</td>
<td>13.3833</td>
<td>10.0743</td>
</tr>
<tr>
<td>Kalman-S</td>
<td>4.8403</td>
<td>13.4203</td>
<td>10.2806</td>
</tr>
<tr>
<td>Kalman-M</td>
<td>4.8761</td>
<td>13.4300</td>
<td>10.2697</td>
</tr>
<tr>
<td>KNN5</td>
<td>5.1027</td>
<td>19.5851</td>
<td>10.6059</td>
</tr>
<tr>
<td>KNN6</td>
<td>5.1675</td>
<td>19.6082</td>
<td>10.5876</td>
</tr>
<tr>
<td>KNN4</td>
<td>5.3371</td>
<td>19.4503</td>
<td>10.9107</td>
</tr>
<tr>
<td>KNN3</td>
<td>6.1696</td>
<td>20.9140</td>
<td>11.6048</td>
</tr>
</tbody>
</table>