# A Model for Automatic Extraction of Slowdowns From Traffic Sensor Data

Silviu Paun, Udo Kruschwitz, and Massimo Poesio
School of Computer Science and Electronic Engineering
University of Essex
Colchester, UK
{spaun, udo, poesio}@essex.ac.uk

***Abstract* – The ability to identify slowdowns from a stream of traffic sensor readings in an automatic fashion is a core building block for any application which incorporates traffic behaviour into its analysis process. The methods proposed in this paper treat slowdowns as valley-shaped data sequences that are found below a normal distribution interval. This paper proposes a model for slowdown identification and partitioning across multiple periods of time and it aims to serve as a first layer of knowledge about the traffic environment. The model can be used to extract the regularities from a set of events of interest with recurring behaviour and to assert the consistency of the extracted patterns. The proposed methods are evaluated using real data collected from highway traffic sensors.**

***Keywords—slowdown detection; rush hour detection; traffic sensors; activity monitoring; time series***

## I. INTRODUCTION

Intelligent Transportation Systems (ITS) make use of sensor data embedded into the environment and into the objects we interact with in order to enhance the transport sector and its direct interaction with people, in a Smart Cities context. Sensor data is more than often available in a real time streaming context and under a continuous-type of model which imposes challenges but in the same time allows the usage of a wide range of knowledge extraction techniques which can benefit from both the historical and the real time aspects of the data. Sensor streams are also one of the sources that fuel the large dimensions (3Vs – volume, velocity, veracity) of Big Data.

There is a large and heterogeneous collection of sensors encompassed into the Internet of Things dataset. From that collection, this work focuses on sensors that provide raw data or information which has utility in the traffic analytics. The sensor readings offer the speed values recorded at road segment level - this type of data is known as aggregate data [1] due to the fact that a sensor reading does not characterize an individual object but rather a collection of objects; in this case the sensor records the average speed of all passing vehicles in a given time window which defines its sampling rate.

The main contribution of this work is the extraction of slowdowns from the sensor streams associated with a collection of road segments. The extracted slowdowns are further clustered to identify their regularity in the traffic behaviour. The slowdowns are treated by the proposed system as valleys in the distribution of the data and the clustering algorithm looks to group together the events which share a similar duration and shape across the sensor streams. The data model introduced in this paper encompasses as well a methodology to slice the sensor stream into appropriate temporal windows to favour pattern related operations such as detection, verification or prediction and a system of metrics which allows to closely control the way the data is grouped to satisfy a variety of data environments. A pattern is detected at temporal window level, it is verified by assessing the historical data and then can be further used for predictions of future events if its consistency score calculated at the verification stage is satisfactory. An example of pattern with a high probability of detection due to its regularity is represented by rush hours, but the same methods could be applied to detect whether there are any recurring accidents, road blocks or other types of congestions which reduce the speed values.

The model proposed in this paper can be utilized in a variety of contexts from multiple fields which require monitoring the behaviour of a variable of interest over time. If the variable of interest is vehicle speed, one can use this approach to extract not valleys but peaks from a traffic sensor stream by looking at the data that is situated above the normal distribution interval. The same approach can be used when analysing social media feeds, where the variable of interest would be the number of posts – this would allow to identify the topics of discussion that are popular or to run query expansion algorithms based on timespan co-occurrence techniques [2] or burst-based pseudo-relevance methods [3].

The remainder of this paper is organized as follows. Section II details the proposed model, presenting the relevant mathematical notations together with the algorithms used to detect the slowdowns and group them as recurring events with similar characteristics. In Section III the experimental results are presented and discussed, while in Section IV the related work is reviewed. The conclusions are offered in Section V.

## II. PROPOSED METHODOLOGY

The data provided by the sensor readings is in most cases accompanied by timestamps or by time windows. The important observation here is that sensor data has a temporal component – this observation introduces an area known as time series analysis. The utility of time series analysis can be

This work has been funded by an EPSRC CASE award studentship with BT plc, whose financial and technical support we gratefully acknowledge.

ISBN: 978-1-902560-27-4 

observed when trying to understand the behaviour of certain variables of interest over time. The usual analysis involves univariate time series – that means having under observation only one variable. In the context of the work presented in this paper the variable of interest is speed and the goal is to identify slowdowns.

### A. Data Representation

The representation of the data has a big impact on the overall quality of the proposed model and on its applicability in other areas. Although the representation is meant to be generic in order to be able to serve multiple activity-monitoring purposes, the discussion in this section lowers the boundaries of abstraction down to our specific problem of slowdown detection from traffic sensor streams. The description of the data representation is going to be made in a bottom-up manner, from the most fundamental element – a sensor reading – to the actual input that goes into the data collection point of the framework.

A sensor reading is represented in the model as a pair of timestamp (t) and speed (s) – a speed point (SP):

$$SP = (t, s) \text{ where } SP.1 = t \text{ and } SP.2 = s \quad (1)$$

A sensor embedded into a road link sends multiple speed points to a data collection service. These speed points are then placed into appropriate collections based on temporal requirements (e.g.: temporal windows of 24 hours):

$$RLD\ (T_s, T_e) = \{SP_1, SP_2, \ldots, SP_n\} \quad (2)$$

where $SP_1.1 = T_s$, $SP_n.1 = T_e$, $SP_i.1 < SP_{i+1}.1$

Equation (2) shows a subset of the entire road link data (RLD) mined during a time window ($T_s$, $T_e$) where the encompassed speed points are ordered by time. In our case, both the start time ($T_s$) and the end time ($T_e$) elements correspond to the first (0:00) and the last hour (24:00) of a daily temporal window. The entire road link data is represented as a collection of such subsets:

$$RLD = \{RLD(T_1, T_2), RLD(T_2, T_3), \ldots, RLD(T_{n-1}, T_n)\} \quad (3)$$

where $T_2 - T_1 = T_3 - T_2 = \ldots = T_n - T_{n-1} = T$

Equation (3) illustrates how the data related to a road link is represented as a collection of speed points mined during equally sized time windows. The size of the time window (T) is selected based on the temporal requirements of the application (e.g.: days, weeks, months, etc.) – in this particular context T is equal to an entire day as we are trying to identify slowdowns which in most cases (e.g.: rush hours) manifest the same behaviour across daily windows (e.g.: it can be certain days of the week, it can be weekdays vs. weekends, etc.).

Now that the data representation layer has been properly introduced it can be said that the framework proposed in this paper takes as input (I) a collection of road link data that comes from multiple road links of interest:

$$I = \{RLD_1, RLD_2, \ldots, RLD_n\} \quad (4)$$

### B. Slowdown Identification

Slowdowns are areas situated below a normal distribution interval in a sensor stream. One can look at slowdowns as valleys in a time series data – based on that observation, slowdowns and valleys will be used in this document interchangeably. To simplify the explanations, this section will restrict all the discussions related to the extraction of valleys to a single road link and from a single RLD(Ts, Te). Based on (3) and (4) introduced in the previous section one can easily see how to extend the logic from a subset of the road link data to the entire data and from a single road link to multiple road links.

The fundamental observation that dictates the way valleys are extracted from sensor data comes from the behaviour of the traffic itself: most of the time sensors emit speed points that fall into a normal distribution interval, but when a slowdown takes place lower speeds are recorded only until the event that caused the slowdown ends – after that the speed points go back up to the regular behaviour. Based on that observation the selection of the normal distribution interval becomes straightforward:

$$[\bar{x} - \sigma, \bar{x} + \sigma] \quad (5)$$

where $\bar{x}$ is the arithmetic mean and $\sigma$ is the standard deviation calculated given the finite set of speed points and the just mentioned mean.

Fig. 1 shows two plotted graphs, both illustrating the same collection of speed points captured over a period of 24 hours from a randomly selected road link. The important thing to notice here is the area between the green lines from the bottom graph representing the normal distribution interval – one can clearly see how the data points fit inside the interval and how a candidate valley looks like.

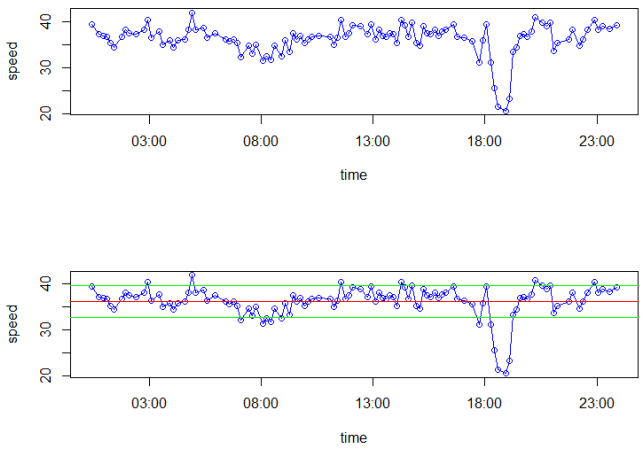


**Figure 1: The recorded speed values for a given road link during a 24-hours period.**

Expressed in a formal manner, a valley is modelled as follows:

$$V = \{SP_1, SP_2, \ldots, SP_n\} \quad (6)$$

where $SP_i.2 < \bar{x} - \sigma$, $SP_n.1 - SP_1.1 > t$

Equation (6) says that a valley (V) is a sequence of speed points with speeds not bigger than a dynamic threshold and a total duration not smaller than a fixed time value. Based on this formal definition, Fig. 2 illustrates an elegant solution to extract valleys of any duration from a stream of speed points with linear complexity. The idea is to extract those sequences of points that are situated below the threshold line. The algorithm also adds the appropriate projection or intersection points at the left and right sides of a candidate valley to make sure the entire shape below the threshold line is captured. The way to achieve all that is to start adding subsequent under-the-threshold-line points into a valley collection until you find a point that breaks the sequence (is situated above the line) – when that happens you simply add the discovered valley/sequence into the a discovered-valleys collection and create a new empty valley. Repeat the logic until the entire time series has been processed.

```
V = empty valley
Vs = empty collection of valleys
V_threshold = x̄ − σ
for(SP_i in RLD(T_s, T_e)) {
  if(SP_i.2 < V_threshold) {
    if(V empty) {
      if(SP_i-1 exists)  //add intersection point
        V U {new SP((V_threshold-SP_i-1.2)*
              (SP_i-1.1-SP_i.1)/
              (SP_i-1.2-SP_i.2) + SP_i-1.1,
              V_threshold)}
      else //add projection point
           V U {new SP(SP_i.1, V_threshold)}
    }
    V U {SP_i}
    if(SP_i.1 == T_e) {
      V U {new SP(SP_i.1, V_threshold)} //adding projection point
      Vs U {V}
    }
  }
  else {
    if(V not empty)
      if(SP_i-1∈V) { //add intersection point
        V U {new SP((V_threshold-SP_i-1.2)*
              (SP_i-1.1-SP_i.1)/
              (SP_i-1.2-SP_i.2) + SP_i-1.1,
              V_threshold)}
        Vs U {V}
        V = new empty valley
      }
  }
}
return Vs
```

**Figure 2: The slowdown extraction algorithm.**

### C. Clustering Similar Valleys

In the previous section it was shown how to extract valleys from the data of a given road link. In this section we are interested in building a model that groups together the extracted valleys which share similar characteristics in terms of location, duration and shape. The grouping required for this job is clustering with automatic k (the number of clusters) inference.

In order to be able to cluster the valleys a similarity metric needs to be introduced. Valleys are stored in the system as polygon-shapes and their similarity should be assessed by the degree of overlapping across temporal windows. Based on that observation, a mathematically strict similarity metric for two valleys is formally presented as follows:

$$S_{temp}(V_i, V_j) = w*S_D(V_i, V_j) + (1-w)*S_S(V_i,V_j) \quad (7)$$

Equation (7) says that the strict similarity between two valleys ($S_{temp}$) is the sum between the similarity in terms of duration ($S_D$) and the similarity in terms of shape ($S_S$). There is also a weighting factor (w) involved which allows the model to control how much the algorithms should rely on duration and shape when calculating the similarity score. The shape-based metric takes into consideration both the (X, Y) axis while the duration-based metric treats the slowdowns as one-dimensional lines. Shape similarity is important as it will prevent, if desired, valleys with small heights to be clustered together with valleys with large heights. In order to assess the degree of overlapping the valleys need to be translated beforehand into the same ($T_s$, $T_e$) interval. $S_D$ is calculated by dividing the intersection of the lines created by the start and end points of the valleys to their union while $S_S$ is obtained by dividing the area of the polygon resulted by intersecting the valley shapes to the area of the polygon resulted by the union of those same shapes. In set theory, the operation of dividing the intersection of two sets to their union is known as the Jaccard index. The result of our similarity metric introduced above is a number between 0 and 1 with 0 meaning there is no overlap between the shapes and 1 meaning there is a perfect overlap. More than often the valleys under analysis will have poor overlapping score – this motivates the need to introduce a degree of relaxation into the metric:

$$S(v_i, v_j) = \begin{cases} S_w + S_{temp}(v_i, v_j), & 0 < S_w + S_{temp}(v_i, v_j) \le 1 \\ 1, & S_w + S_{temp}(v_i, v_j) > 1 \\ 0, & S_{temp}(v_i, v_j) = 0 \end{cases} \quad (8)$$

where $S_w \in [0, 1]$

Equation (8) adds a similarity weight ($S_w$) to the overlapping score in order to allow valleys with strong and less strong similarities to be clustered together. The new similarity metric (S) has a big impact on the way the clusters are built and it is of great importance for the flexibility of this model under generic contexts. To clearly explain the properties of the equation, a use case scenario example is introduced. Let's say we have two pairs of valleys with qualities $Q_1$ and $Q_2$ calculated using $S_{temp}$, where $Q_2 > Q_1$. If $Q_1$, $Q_2 > 1-S_w$, then $Q_1=Q_2=1$ and the clustering algorithm will treat the situation as if the valleys had a perfect fit and they will end up in the same cluster. Another very important aspect is represented by the fact that if $Q_1$, $Q_2 \le 1-S_w$ then the rate between $Q_2$ and $Q_1$ will no longer be $Q_2/Q_1$ but $(Q_2+S_w)/(Q_1+S_w)$ which produces a significant impact on the output of a quality aggregator. To illustrate that let's say that $Q_1=0.1$, $Q_2=0.4$ and $S_w=0.2$. The $Q_2/Q_1$ rate is 4 while the $(Q_2+S_w)/(Q_1+S_w)$ rate is 2 – this

clearly shows that the two qualities are much closer to each other after $S_w$ is added which means the valleys they represent are more likely to be clustered now, after the weight addition, than before.

Having defined a similarity metric for valleys allows us to introduce a metric to assess the quality of a cluster:

$$Qc_k = \begin{cases} \frac{\sum S(v_i,v_j)}{\binom{2}{|C_k|}} * \frac{|C_k|}{n_v}, |c_k| > 1 \\ 0, \ |c_k| \leq 1 \end{cases} \quad (9)$$

where $n_v$ is the total number of valleys from all clusters

Equation (9) shows that the quality of a cluster ($Qc_k$) is the average of the similarities of all the pairs of valleys from that cluster weighted in such a way to favour the clusters with more elements. If the weighting component would not exist, in a use case scenario, given five valleys and two clusters $C_1$, $C_2$ with $Qc_1=1$, $|C_1|=2$ and $Qc_2=1$, $|C_2|=3$, the system will treat both clusters with equal importance. By taking the weighting component into consideration, $C_1$ would have $Qc_1=2/5$ and $C_2$ would have $Qc_2=3/5$ and the system will recognize $C_2$ as a better cluster. This is important because the weight addition aids the creation of clusters with multiple but similar slowdowns and avoids outputs such as "best-fit" clusters of two elements only.

Knowing how to calculate the quality of a single cluster allows us to move even further and define a quality for the entire set of clusters:

$$Q_G = \frac{1}{G_w + n_c} * \sum Qc_j \quad (10)$$

where $n_c$ is the number of clusters in a given iteration and $G_w$ is a weight for the aggregation rate.

Equation (10) says that the quality for all clusters ($Q_G$) should not be aggregated under a traditional average rate ($1/n_c$) but under a $1/(G_w+n_c)$ rate. The average rate is not suitable because it has a drastic impact on the way it increases as the number of clusters decreases; therefore, it favours the system to produce fewer clusters than desired and to treat the cluster qualities between iterations with less importance. For example, having three clusters would produce an average aggregation rate of 1/3; having two clusters the average aggregation rate will be 1/2 which will result in a global score at least 1.5 times bigger regardless of the values of the clusters' qualities. Adding a large $G_w$ into the equation will make sure the aggregation manifests a smooth increasing rate as the number of clusters decreases.

The reason why both the local and the global cluster qualities were introduced is that they will be used by the clustering algorithm to select a proper value for k. Fig. 3 shows in a pseudocode style how the valleys are clustered together based on similarity. The clustering algorithm proposed is similar to a hierarchical/bottom-up clustering in the sense that each valley starts as part of a separate cluster and with each iteration a merger occurs. The algorithm decides each iteration to merge those two clusters that produce the maximum $Q_G$. If there is no possible merger during an iteration that produces a bigger $Q_G$ value then the algorithm stops and outputs the set of clusters obtained until that step.

The set of clusters returned by the algorithm provides many useful insights about the slowdowns. The quality measures can be used to assess the consistency of a slowdown in the historical data. On high quality clusters one can use a frequency-based technique to extract the temporal regularities of the slowdowns and create rules such as the hours in the day and the days in the week when rush hours occur for each road link.

```
Cs = empty collection of clusters
Vs = collection of valleys
for(Vi:Vs)
  Cs U {new C(Vi)}

clusterFlag = true
Qmax = 0
betterCs = collection of empty clusters
while(clusterFlag) {
   clusterFlag = false
   for(i=0; i<|Cs|-1;i++) {
    for(j=1;j<|Cs|;j++) {
      tempCs = {new C(Ci, Cj)} U (Cs\{Ci,Cj})
      QG = calculateGlobalQuality(tempCs)
      if(QG > Qmax) {
        Qmax = QG
        betterCs = tempCs
        clusterFlag = true
      }
    }
   }
   if(clusterFlag)
     Cs = betterCs
}
return Cs
```

**Figure 3: The slowdown clustering algorithm.**

## III. EVALUATION

The evaluation of the proposed methods consists in an illustration of their behaviour in a real data environment. The data used in the experiments was collected in real time from traffic sensors installed on various UK highways over a period of 10 days. The evaluation regards two aspects of our work which are evaluated in separate phases: the slowdown detection method and the clustering algorithm.

### *A. Evaluation of the Slowdown Detection Method*

It was described in a previous section that slowdowns are valley-shaped sequences of speed points which are situated below a normal distribution interval $[\bar{x} - \sigma, \bar{x} + \sigma]$. The role of this evaluation section is to illustrate why that specific interval was chosen.

For this experiment the data from 117 road link sensors was considered across a period of time of 10 days – more than 1,300,000 speed points were collected in that period using a polling technique fired every minute. After a filtering layer the data was reduced to a total of approximately 30,000 speed

points ready for analysis. The filtering stage had a drastic impact due to the fact that all the points inside the sampling rate interval (quite constant, in the 10 minutes range) were discarded and that each daily road link data had to contain at least 100 recordings in order to be qualified as valid. Fig. 4 illustrates the mean and the standard deviation for the daily road link data as reported by the 117 sensors. One can clearly see from the graphic by looking at the standard deviation distribution that the assumption there is little variation in the daily data was correct – the mean of the standard deviations recorded is 2.87 while the standard deviation is 1.4; the small variation in the standard deviation distribution is kept even if the means of the recorded speed values vary from day to day and from road link to road link. This observation confirms that the $[\bar{x} - \sigma, \bar{x} + \sigma]$ range is a good choice to represent the normal distribution interval. Please note that the little variation formulation is appropriate here in the traffic data environment based on the value of the standard deviation but it might not be appropriate in other more sensitive contexts. More than that, a smaller variation of the standard deviation distribution could be obtained if one would not consider the cases in which the road link sensors record no speed at all due to the lack of traffic.

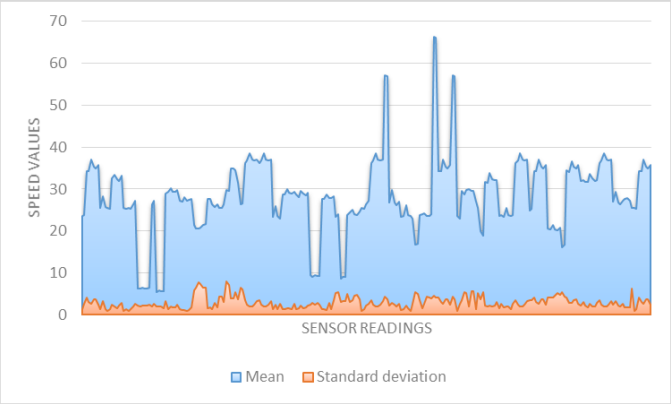


**Figure 4: The mean and the standard deviation of a collection of daily recorded speed values from multiple road links over a period of 10 days.**

### *B. Evaluation of the Clustering Algorithm*

In this section the output of the clustering algorithm is examined in order to illustrate its ability to group the slowdowns based on characteristics such as shape, duration and location.

For this experiment the data from a single road link recorded during a 10 days period was considered. More than 14,000 sensor recordings were filtered and reduced to approximately 760 speed points based on the filtering process described in the previous section. From that set of data points a total of 11 valleys were extracted by the slowdown identification process and taken as the input of the clustering algorithm. The goal for this experiment was to identify the regularities (e.g.: rush hours) in the slowdowns of a given road link.

Fig. 5 illustrates in the top graph the valleys detected during the 10 days period while in the bottom graph it shows how the extracted valleys were clustered. Each rectangle represents the average characteristics of the valleys from a cluster: position, area and duration. The proposed system automatically builds an R template with all the results for easy visualization. Since the goal was to identify the consistency in the daily slowdowns, the shape of the valleys was not considered (w=1). The other weighting parameters used in the experiment were $S_w$=0.2 and $G_w$=3*$n_v$. The parameter selection is performed based on the desired outcome. One can easily assess how to control the grouping and similarity by running the algorithm with different parameter settings. Fig. 5 illustrates only a 12 hour interval due to the fact that in the daily streams used in this experiment no valleys were detected outside that range.

The clustering algorithm produced a total of 5 clusters: one with 5 valleys, two with 2 valleys, and two with 1 valley. The results show a possible pattern based on the number of valleys recorded in the first mentioned cluster, but the small number of assessed days/valleys does not allow to draw such conclusions. Another useful piece of knowledge is that in 4 out of the 10 days of monitoring no valleys were identified.

This type of information about the characteristics of the slowdowns such as location, duration, shape and regularity in the daily streams or even about the lack of slowdowns can be easily exploited in a route-recommendation system as a priori knowledge for multiple purposes: to speed up the algorithms by properly skipping the congestion-links or to plan ahead the route based on congestion regularities.

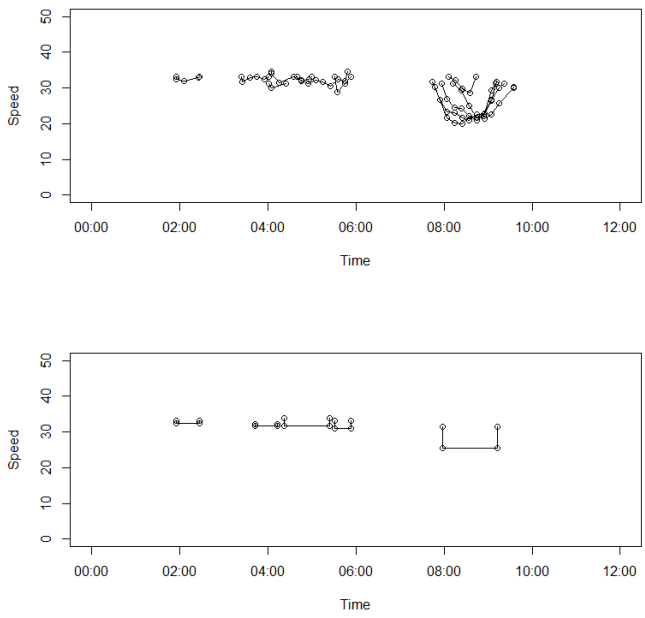


**Figure 5: The input (top) and output (bottom) of the slowdown clustering algorithm on a collection of 10 days of data mined from a road link sensor.**

## IV. RELATED WORK

In the field of traffic sensor analysis there is a lot of work done in correlating different types of sensor information for pattern extraction purposes [1] [4] [5] [6] [7]. There is also an increased focus on trajectory analysis [8] [9] which is a core topic for research efforts trying to enhance the existing Intelligent Transportation Systems.

The problem of extracting slowdowns from a sensor stream required a unique approach which does not fit the traditional methods for valley detection. Both MatLab and R have useful packages to identify the points in time series data which are situated lower on the y-axis in comparison to their neighbour regions – so the problem of valley detection in these cases is reduced to the problem of identifying the local minimums. This solution does not serve the purpose of this work from multiple reasons. A traffic slowdown is an area which is situated below the normal distribution interval of the data and which can have more than one local minimum point due to the small variations in the traffic speeds. More than that this work targeted to capture the entire shape that represents a valley; so a single point detection method is not suitable.

There is work in the literature related to the problem of clustering the data in a hierarchical manner until a stopping condition is met. In traditional hierarchical clustering you start with each data point in a separate cluster and you try to climb up the hierarchy by merging pairs of clusters that satisfy best a similarity metric. This kind of approach is mostly found in bottom up methods used to segment time series data based on Piece-wise Linear Representation techniques [10] [11] [12] [13] [14]. The similarity of this work with the bottom-up clustering algorithms is only at conceptual level: this work uses the bottom-up methods as a natural way of comparing the data, but the actual novel grouping is performed by the proposed quality metrics.

There is also work done in the area of detecting similar shapes in time series data [10] [15], but the metric proposed in this paper is more concerned with assessing the degree of shape overlapping rather than capturing sensitive shape variations.

There is one research effort from the literature which follows the idea of identifying frequent activities from a historical dataset of sensed events [16], but their methods do not deal with shapes as they treat the events as one-dimensional lines and there is no need to identify the events in question beforehand as they have them as a priori information.

Generic links can also be made between this work and statistical outlier/anomaly detection methods (e.g.: confidence intervals in Gaussian distributions). The proposed methodology is modular making it connected with fields where a deviation from a statistical normality is aimed to be discovered (e.g.: network traffic analysis).

## V. CONCLUSIONS

This paper presented in detail a model for the automatic extraction and clustering of slowdowns from traffic sensor streams. The importance and novelty of this work comes from the two core aspects of the proposed framework: a data representation which favours the slowdown extraction as valley-shaped sequences of points that are found below a normal distribution interval and a bottom-up clustering algorithm with metrics which allow to closely control the way the data is grouped in order to satisfy a variety of contexts and data environments.

## REFERENCES


[1] Tanvi Jindal, Prasanna Giridhar, Lu-An Tang, Jun Li, and Jiawei Han. Spatiotemporal periodical pattern mining in traffic data. In Proceedings of the 2Nd ACM SIGKDD International Workshop on Urban Computing, UrbComp ’13, pages 11:1–11:8, New York, NY, USA, 2013. ACM.

[2] Donald Metzler, Congxing Cai, and Eduard Hovy. Structured event retrieval over microblog archives. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 646–655. Association for Computational Linguistics, 2012.

[3] Chen Lin, Chun Lin, Jingxuan Li, Dingding Wang, Yang Chen, and Tao Li. Generating event storylines from microblogs. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM ’12, pages 175–184, New York, NY, USA, 2012. ACM.

[4] Farnoush Banaei-Kashani, Cyrus Shahabi, and Bei Pan. Discovering patterns in traffic sensor data. In Proceedings of the 2Nd ACM SIGSPATIAL International Workshop on GeoStreaming, IWGS ’11, pages 10–16, New York, NY, USA, 2011. ACM.

[5] Eric Bouillet, Bei Chen, Chris Cooper, Dominik Dahlem, and Olivier Verscheure. Fusing traffic sensor data for real-time road conditions. In Proceedings of First International Workshop on Sensing and Big Data Mining, SENSEMINE’13, pages 8:1–8:6, New York, NY, USA, 2013. ACM.

[6] Anthony Harrington and Vinny Cahill. Route profiling: Putting context to work. In Proceedings of the 2004 ACM Symposium on Applied Computing, SAC ’04, pages 1567–1573, New York, NY, USA, 2004. ACM.

[7] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. Discovering spatio-temporal causal interactions in traffic data streams. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’11, pages 1010–1018, New York, NY, USA, 2011. ACM.

[8] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: Driving directions based on taxi trajectories. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS ’10, pages 99–108, New York, NY, USA, 2010. ACM.

[9] Kai Zheng, Yu Zheng, Xing Xie, and Xiaofang Zhou. Reducing uncertainty of low-sampling-rate trajectories. In Data Engineering (ICDE), 2012 IEEE 28th International Conference on, pages 1144–1155, April 2012.

[10] E. Keogh. Fast similarity search in the presence of longitudinal scaling in time series databases. In Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on, pages 578–584, Nov 1997.

[11] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on, pages 289–296, 2001.

[12] Eamonn J Keogh and Michael J Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In KDD, volume 98, pages 239 – 243, 1998.

[13] Eamonn J Keogh and Padhraic Smyth. A probabilistic approach to fast pattern matching in time series databases. In KDD, volume 1997, pages 24–30, 1997.

[14] Stephan Spiegel, Julia Gaebler, Andreas Lommatzsch, Ernesto De Luca, and Sahin Albayrak. Pattern recognition and classification for multivariate time series. In Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data, SensorKDD ’11, pages 34–42, New York, NY, USA, 2011. ACM.

[15] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In KDD workshop, volume 10, pages 359–370. Seattle, WA, 1994.

[16] Dimitrios Lymberopoulos, Athanasios Bamis, and Andreas Savvides. A methodology for extracting temporal properties from sensor network data streams. In Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services, MobiSys ’09, pages 193–206, New York, NY, USA, 2009. ACM.