TRAFFIC BOTTLENECK ANALYSIS AND VISUALIZATION OF THE GREATER TORONTO AREA HIGHWAY NETWORK

ESC499Y Thesis Final Report

April 11, 2018
Supervisor: Professor Baher Abdulhai

Gisele Samaan

1 Abstract

Big Data visualization has the potential to transform transportation and land use planning. The 400-series highway network in the Greater Toronto and Hamilton Area is a vital piece of infrastructure and a better understanding of its use will go a long way in informing traveller information systems and emergency responders. This thesis is divided into two parts. The first part takes speed measurements during the month of July 2017 from loop detectors embedded in the highway bed are analyzed to identify bottlenecks occurrences. The Travel Time Index, a ratio of actual travel time over a threshold travel time is used to determine the existence and severity of congestion conditions. The second part takes these values and visualizes them via an R Shiny-based interactive web application. Tens of thousands of data points are translated into this simple medium where users can see how traffic conditions evolve over the course of a 24-hour period for each day in the month of July 2017. The results are consistent with previous observations on traffic behaviour on this highway network. Work from here lays the foundation for a more comprehensive traffic information system being developed by the Intelligent Transportation System of Systems group at the University of Toronto.

2 Acknowledgements

In the course of developing this thesis, I have received much help from many people. Firstly, I would like to thank my supervisor Professor Baher Abdulhai for giving me this challenge and providing feedback along the way. Ahmed Aqra, who has been supervising the Intelligent Transportation System of Systems team, has been keeping all of us on task and I would like to thank him for guiding us through the turbulent initial stages of setting up this group. I would also like to thank Mohammed Elshenawy who pointed me to data imputation programming codes. Anastassios Dardas helped with the more complex parts of this thesis, the map-generation tools and procedures, debugging, and with suggestions on web application design.

Contents

3 Table of Figures

4 Introduction

The nature of mobility, the means and patterns of transportation, is shifting toward a consumer-based service that will require accurate real-time information on traffic conditions for various modes of travel (e.g. car, cycling, subway, etc.). Information traveler systems today rely on often outdated, incomplete and unreliable sources of information. Google Maps, which comes closest to providing comprehensive real-time travel information for different modes relies on spotty, random samples of GPS devices. Its reliability depends on availability of GPS-enabled devices and may result in outdated information on traffic and road conditions. The best way to get reliable real-time information on traffic conditions is directly from sensors placed in the infrastructure itself, e.g. loop detectors, Bluetooth detectors. These devices measure speed and volume information along a corridor and, with some Big Data processing, can be a part of an Intelligent Transportation System of Systems [1]. This report starts with a literature review to build a picture of the study area and data processing techniques relevant to the thesis topic. It then describes the work for this thesis divided into 2 parts. The first part examines loop detector speed and volume data and performs bottleneck analysis on highway segments in the Greater Toronto Area to determine the Travel Time Index, an indicator of congestion, of each segment over a one-month period, July 2017. The second part describes the development of an interactive web application based on R Shiny Package that visualizes the TTI values on a map of the highway network. Future work is required to further clean up the data and tweak the visuals using advanced JavaScript-based graphics. The work here forms one part of a bigger academic endeavour with the Intelligent Transportation System of Systems group at the University of Toronto Department of Transportation.

5 Literature Review

As mentioned above, travel information comes in the form of travel apps for flights, transit, and driving. Google Maps is perhaps the most widely used source of basic travel information, making use of crowd-sourced GPS information to determine traffic conditions, travel times, and multiple alternatives. The next phase of the evolutions of these systems is to gather raw data from sensors in the very infrastructure used by commuters and translate it into information understandable to users. The information would be tailored for many types of users ranging from commuters trying to get from point A to point B using real-time travel information, to municipal officials, infrastructure operators, and researchers interested in past data to study travel behavior in various conditions (e.g. shockwave effects of bottlenecks on a highway during various weather conditions, or the effects of construction scheduling on congestion patterns, etc.). The following is a literature review of past and current research related to bringing Big Data traveler information into its next phase. They cover the full spectrum of work, from the back-end data collection and analysis, to the front-end visualization and user-friendly interactivity.

5.1 Toward a Seamlessly Integrated Cyber-Physical Intelligent Transportation System of Systems

This conference paper acknowledges work done in communicating travel information and in smart transportation systems such as smart parking applications and digital trail maps for cyclists, but identifies a gap in bringing them together and integrating them all into one place [2]. Currently, they are all operating independently and isolated from each other. The authors outline a framework, referred to as an Intelligent Transportation System of Systems, as a more efficient and informative source of information on transportation infrastructure throughout entire regions. This framework's purpose would go beyond the function as an Advanced Traveller Information System. The authors outline three pillars of this framework.

The first, called Ontological Semantic Knowledge Representation, proposes the necessity of ensuring smooth interoperability between the 'cyber-physical' components such as sensors, their software components. It also proposes that they be able to communicate with other components [1]. The idea is to design a platform that takes in data from various sensors (e.g. cameras, loop detectors, Bluetooth detectors, crowd-source, etc.) and have them interact within one 'System of Systems' and automatically coordinate to create a holistic picture of the transportation infrastructure that users could understand. The authors describe the four components of the ontology for this system of systems in Figure 1.

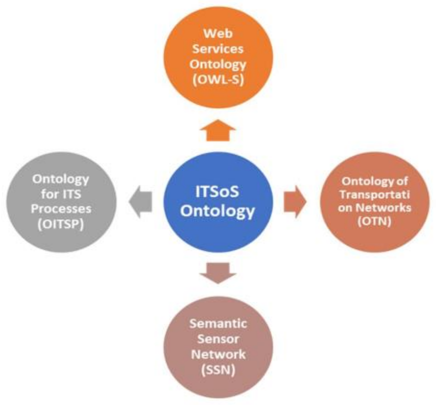

Web Services Ontology (OWL-S)
Ontology for ITS Processes (OITSP)
ITSoS Ontology
Ontology of Transportation Networks (OTN)
Semantic Sensor Network (SSN)

Figure 1: Four components of ITSoS Ontology [2]

The second pillar, called Integrated Service Planning, makes use of the first and calls for combining information from various sources and types of sensors to get a comprehensive set of useful information [1]. The authors suggest a hierarchy involving small, specific services such as pre-trip information and route guidance and navigation. These services are in turn broken down into a data collection component and data analysis mechanism. The platform would draw on these services in a way that is unique to each user's request, e.g. for trip planning.

The third pillar, called Integrated Service Execution, which the authors refer to as the 'management infrastructure', would encompass data processing and user interface execution in a way that meets the communications standard of the platform (real-time information vs past conditions for research purposes) [1]. This part would keep track of and present the results of independently-operating sources of transportation information in a user-friendly context through what the authors refer to as a 'messaging infrastructure', e.g. combining travel times on a roadway and visibility levels [2]. Figure 2 illustrates the theoretical structure of this pillar.

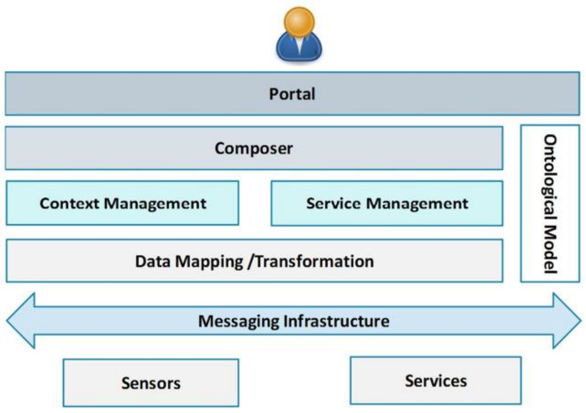


Figure 2: Theoretical structure of the ISE [2]

This paper presents steps taken to implement a prototype that includes a trip-planning component, a software platform that provides real-time road conditions (ONE-ITS), and a platform that provides real-time transit bus information (NextBus). With the ontological format, the prototype can present the user with relevant information from each service insomuch as they rely on to each other and the connections between their data sources on the transportation infrastructure [2]. Together, this makes up an Advance Traveller Information System depicted in Figure 3.

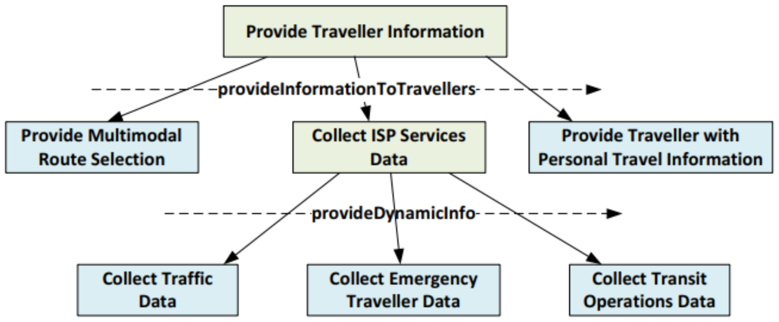

Provide Traveller Information
provideInformationToTravellers
Provide Multimodal Route Selection
Collect ISP Services Data
Provide Traveller with Personal Travel Information
provideDynamicInfo
Collect Traffic Data
Collect Emergency Traveller Data
Collect Transit Operations Data

Figure 3: ATIS task network [2]

The paper concludes by noting future work to be done that includes combining multiple sources of information from various types of sensors, beyond GPS [2]. The design project discussed in this report expands on the work done for this paper by designing a service for highway bottlenecks using data from loop detectors.

5.2 Multi-Sensor Data Fusion for Traffic Speed and Travel Time Estimation

On the back-end of the analysis required for proper visualization of congestion is discussed in this paper by Bachmann where he studies the fusion of data collected from multiple detector technologies [3]. He compares estimation techniques and evaluates them on their ability to accurately measure speed data using Bluetooth-enabled devices, loop detectors, and GPS. There is growing interest in using wireless technologies to gather information and communicate between devices. Bachmann sets out to compare the data collection performance of both devices and examines ways the information they gather could be fused together. He evaluates the effectiveness by comparing to microsimulation results and GPS-enabled traffic information that is widely available and considered to be the main standard [3].

The advantage of data fusion lies in the ability to use one type of sensor to make up for the shortcomings of another type of sensor measuring the same phenomenon, and vice versa. It also increases the resilience of the traffic information system. Having multiple sources measuring the same phenomenon increases the accuracy of the measurement by providing more data points to work with. Also, fewer data points and data processing power is needed, and therefore the speed of operation increases with multiple sources than when relying on merely one type of sensor [3].

The data fusion techniques that Bachmann outlines are: Simple Convex Combination, Bar-Shalom/Campo Combination, Measurement Fusion, Single-Constraint-At-Time (SCAAT) Kalman Filter, Ordered Weighted Averaging (OWA), Fuzzy Integrals, Artificial Neural Networks, Fusion Architectures, and Measures of Effectiveness.

Bachmann included a real-world case study using data collected from Bluetooth detectors and loop detectors on a stretch of Highway 400. He created a virtual version of that stretch in Paramics for microsimulation to cross-validate speed measurements. Bachmann found that loop detectors, although they measure speeds of all passing vehicles, are the least accurate due to technological constraints, but Bluetooth detectors, although reliant on an adequate sample size, is the most accurate. He also found that most of the fusion techniques improve the reliability and accuracy of speed information, especially those that give a larger weight to Bluetooth data. Notably, a fusion architecture that has a largest amount of raw data from each sensor type would give the most accurate measurements as the relationships between various types of sensors would be well understood [3].

This thesis design project designs and builds a service that utilizes all the raw data from all the loop detectors in the Greater Toronto Area so that future work that utilizes multiple sensor types and data fusion techniques outlined here will be able to get a higher quality reading on traffic conditions.

5.3 Integrated Congestion Evaluation and Quantification

Scott Johnson sets out to describe a methodology to quantify congestion [4]. The threshold for congestion used, called the Toronto Congestion Index, is like a Travel Time Index where a value of 2 indicates travel times that are twice as long as normal conditions. The methodology, programmed into the Integrated Congestion Evaluation and Quantification (ICEQ), indicates the TCI for PM 2-hour period and the number of users who experience a certain TCI value. The goal of Johnson's thesis is to combine the use of Intelligent Transportation Systems for data collection with the development of a congestion index that gives information not only about traffic conditions, but also how those conditions affect specific trips. For example, this index would indicate the effect of road incidents on transit riders [4].

Data collected from several highway loop detectors are analyzed and the value of the congestion ratio (actual travel time to reference travel time) distribution determines the value of the Toronto Congestion Index: "The congestion index is the congestion ratio with a set probability of being exceeded based on the cumulative distribution function." Two types of data are used to determine this index: Traffic Management System sources such as highway loop detectors, and GPS data. Johnson includes in his paper an aspect of data visualization useful for research purposes. This visualization that includes graphs of speed readings and fundamental diagrams illustrated the extent of data volatility that required refinement and smoothing techniques. The paper discusses the effectiveness of using various statistical analysis tools to test for viable data such as testing for normality, lognormality, and the use of 3-D space-time-speed diagrams. The data-cleaning involved comparing actual speeds to rolling averages or interval averages [4]. Figure 4 shows the volatile nature of loop detector data and how it can be smoothed by averaging over 3-minute intervals.

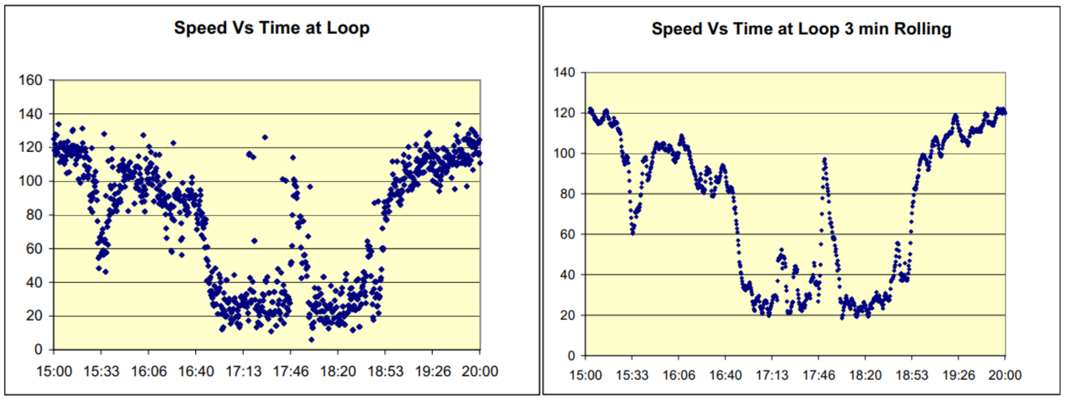


Figure 4: Data smoothing study of loop detector data [4]

Johnson found the average absolute error from both using rolling averages and interval averages to be around 8 km/h for one loop detector location. He chooses to calculate the Congestion Ratio based on the interval averages method since, to make it useful for highway operators and users, it must not be affected by data points in the past [4]. Figure 5 shows the findings of calculating the congestion ratio at one location over the PM peak period. The volatility during the 4pm to 6pm period is related to the issue of higher volatility in the loop detector data at lower speed measurements, a point that is particularly relevant to this design thesis.

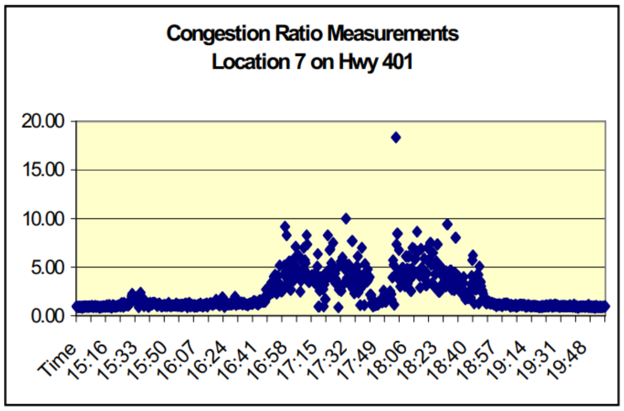


Figure 5: Congestion ratio plot using interval averages [4]

The TMS-based CR gives information on travel performance that is specific to one location. Johnson used GPS data to find the congestion performance of entire trips based on their origin-destination patterns. This part of the study used microsimulation, to determine the congestion ratio based on link performance values from Paramics simulation software. The OD inputs to the simulation are the OD patterns determined from the GPS data. The use of GPS data permits the study of congestion levels for transit users separately from road users.

To find the congestion index of entire regions, in this case of Toronto, Johnson made use of the Travel Time Index aggregated over multiple locations:

$$CR = \frac{60/V}{60/V_{Ref}}$$

Where V is speed in km/h and V_{Ref} is chosen as the speed at capacity, or maximum flow rate as illustrated below:

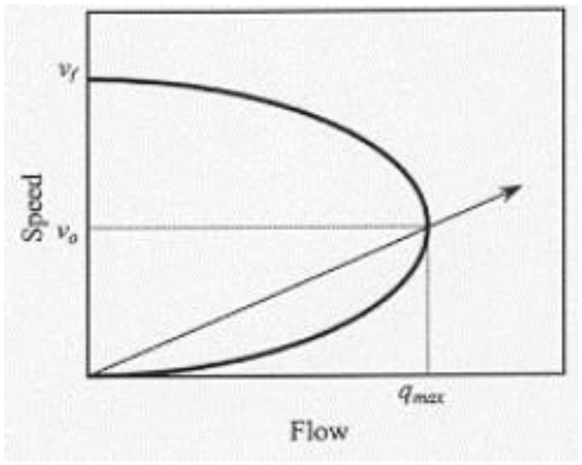


Figure 6: Speed-flow fundamental diagram used to determine speed at capacity used for the reference speed [4]

Finding speed at a capacity requires data on both speeds and volumes, which is provided by loop detectors. The issue here is that this is only available for freeway segments. GPS data does not provide information on volumes and flow rates. Since the goal of this paper is to find the congestion index for Toronto, the reference volume is the free flow speed. A congestion index using the free flow speed will show a value of 1.33 that corresponds with a congestion index using speed at capacity with a value of 1 [4]. For this reason, Johnson concludes that an adjustment of 0.33 will be required to get the actual additional travel times at perceived congestion levels. Using a free flow speed of 110 km/h on freeways based on available speed data was used as the threshold. Figure 7 shows a speed vs CR graph illustrating the relationship between speeds and corresponding congestion ratio:

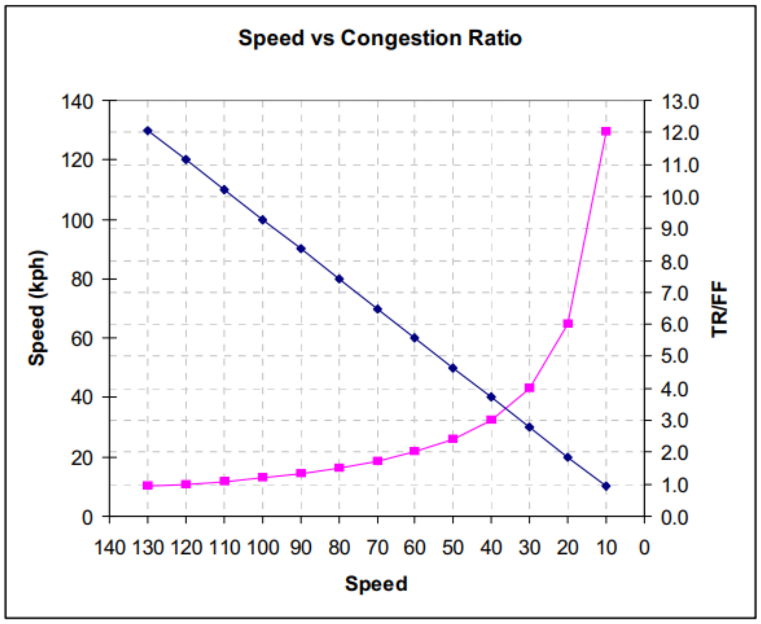


Figure 7: Speed vs Congestion Ratio using free-flow speed as threshold [4]

The congestion ratio is more sensitive at lower speeds, which could be problematic when the speed data is very noisy as is the case with loop detector data (see also Figure 8). It would be very difficult to get an accurate reading of congestion levels precisely at the speed ranges where this measurement would matter to travellers:

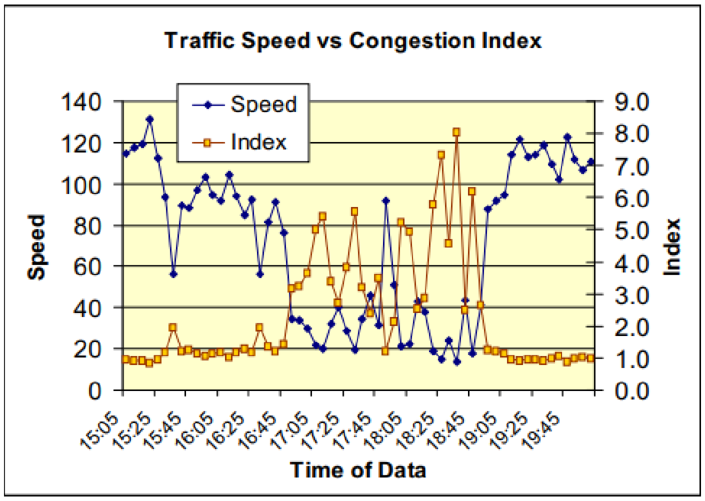


Figure 8: Rush hour speeds give rise to volatile congestion ratios [4]

To help with this, any speed data below 10 km/h has been ignored since it would imply near-infinite travel time, which is not the case indicated by GPS-based origin-destination trips [4]. The resolution of this issue for Johnson is to aggregate the CR’s over the entire city for the TCI, reducing noise associated with these low speeds. The congestion ratios were aggregated for the

4pm to 6pm peak period averaged over several days' worth of data. Specifically, three segments that each represent "limited, moderate, and severe" congestion levels had their congestion ratios averaged over 5 workdays [4]. Below in Figure 9 is the cumulative distribution function for each segment:

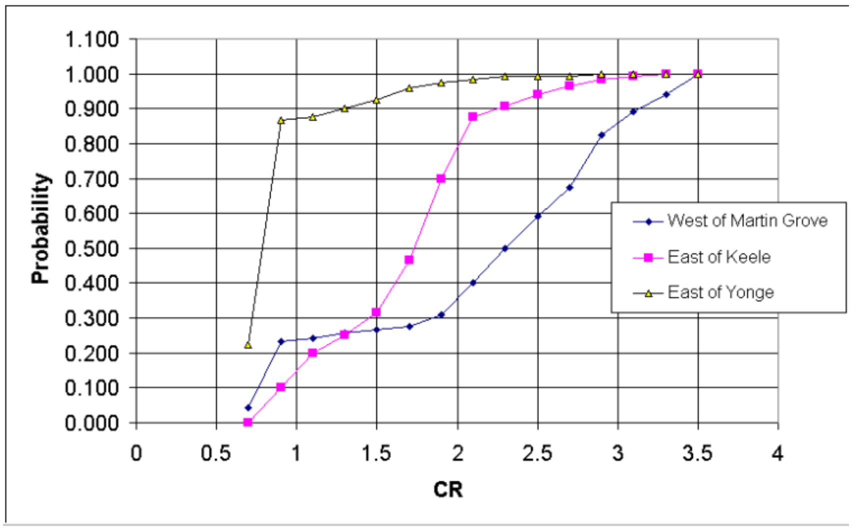


Figure 9: CDF of congestion ratios of 3 freeway segments [4]

The 80th percentile TCI value has been chosen to represent the congestion ratio of those locations. These individual segments' CRs are then applied to OD patterns based on the above Paramics simulation and GPS data. The Toronto Congestion Index is determined thusly [4]:

$$TCI_i = \frac{\sum CR(i, j) \cdot N(i, j)}{\sum N(i, j)}$$

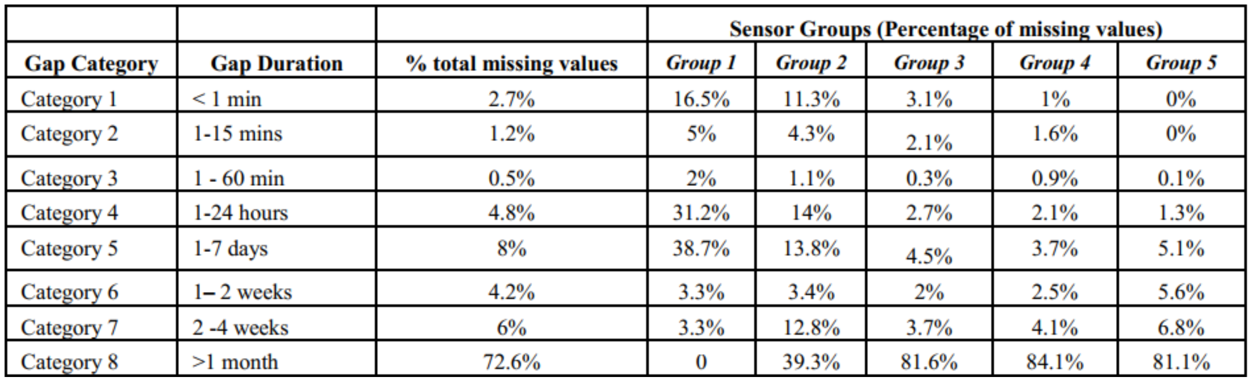
The traffic simulation determines which paths are taken and therefore the value of N (volume) for each segment with a CR value. This weighted average over OD pairs is the Toronto Congestion Index. This paper also describes the software programs developed and designed to evaluate the TCI, called the Intelligent Congestion Evaluation and Quantification based on Visual Basic for Applications Excel [4].

Johnson also extends the analysis by using the TCI to determine Economic Cost of Congestion. For this the Economic Congestion Ratio of the cost of congestion to the cost of improvement is used to determine the potential economic improvements that would result from an improvement in travel time [4].

The methods used by Scott Johnson to collect, process, and analyze speed data to find a Travel Time Index is identical to the one used for this thesis. The challenges with noise are the same since both use the same data source: loop detectors. What differs is that this thesis relies on local TTI values with no spatial aggregation to smooth the volatility in the data.

5.4 Automatic Imputation of Missing Highway Traffic Volume Data

This paper deals directly with the issue of incomplete data, such as speeds and volumes measurement from loop detectors [5]. The authors developed a method to fill in the missing data points that result from system/equipment failure. The sensors studied, which are loop detectors placed along the 400-series highway in the GTA, were divided into 5 groups based on the percentage of missing data points collected over all of 2016. On average, 31.5% of data points were missing. The length of data gaps ranged from a couple of minutes to months. The sensors were divided into 8 groups based on the gap duration. The sensors with the shorter-duration gaps tend to be the same as the sensors with fewer missing data. The paper focused on imputing values for data gaps less than a week long. The classification is shown in Figure 10:



			Sensor Groups (Percentage of missing values)				
Gap Category	**Gap Duration**	**% total missing values**	***Group 1***	***Group 2***	***Group 3***	***Group 4***	***Group 5***
Category 1	< 1 min	2.7%	16.5%	11.3%	3.1%	1%	0%
Category 2	1-15 mins	1.2%	5%	4.3%	2.1%	1.6%	0%
Category 3	1 - 60 min	0.5%	2%	1.1%	0.3%	0.9%	0.1%
Category 4	1-24 hours	4.8%	31.2%	14%	2.7%	2.1%	1.3%
Category 5	1-7 days	8%	38.7%	13.8%	4.5%	3.7%	5.1%
Category 6	1– 2 weeks	4.2%	3.3%	3.4%	2%	2.5%	5.6%
Category 7	2 -4 weeks	6%	3.3%	12.8%	3.7%	4.1%	6.8%
Category 8	>1 month	72.6%	0	39.3%	81.6%	84.1%	81.1%

Figure 10: Classification of sensors based on % missing data and length of data gaps [5]

To fill these gaps, the existing data was aggregated into 15-minute intervals to minimize the noise as shown below in Figure 11:

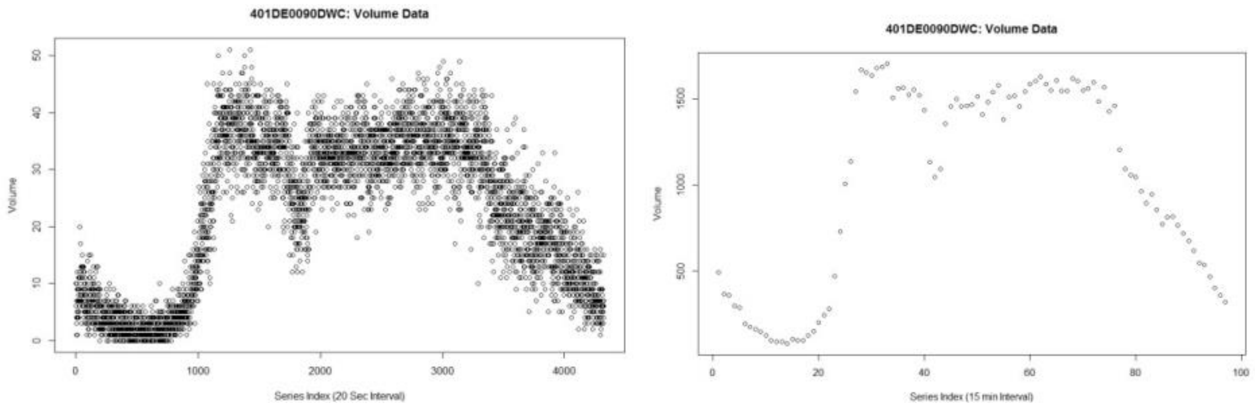


Figure 11: Aggregation of volatile loop detection data into 15 min intervals [5]

Univariate time series methods are used to calculate the missing values. The relevant formula is

$$X_t = S_t + T_t + R_t ,$$

where X is the time series, S represents the variation in the data over one day, T is the data trend, and R represents other irregularities. The method recommends that only regular weekdays be used. Two stationary series (where mean, variance, and others are constant over time), one calculating differences in data values 24 hours apart from each other and the second a week apart, use the ARIMA technique to smooth the variation (using moving averages), and find the imputed values for the missing data points [5]. The data from the real world, along with the imputed values, was compared to data from a simulation to test for accuracy. Three comparison methods, Mean Absolute Error, Mean Absolute Percent Error, and Root Mean Square Error, were used and found the imputed values, especially those during high-volume periods, nearly matched the simulated values [5].

This paper presents a possible method to be used to smooth the great variations found in the loop detector data for this thesis. Indeed, it presents a technique directly applicable to short gap durations that do exist in the data used for this thesis.

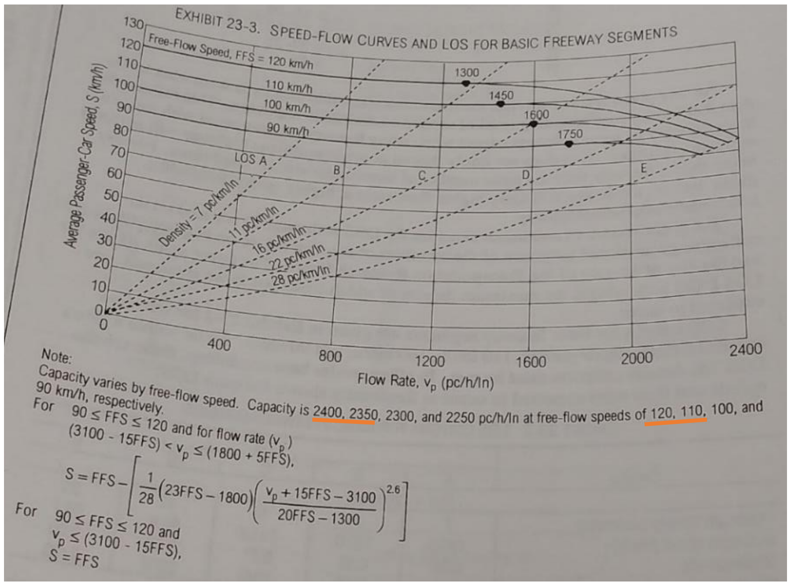
6 Bottleneck Analysis and Web Application Design

There are two parts to this project each with one set of R code. The first part deals with performing bottleneck analysis using loop detector data. It has an R script that takes raw loop detector data as an input, processes it, and outputs daily TTI values, and records the GPS coordinate of the loop detectors analyzed. The TTI values files with the coordinates of the loop detector location, in the form of a .csv file, is used as input for the second R script. The second R code is the web application that utilizes the R Shiny package, app.R, which reads from the list of TTI files, extracts the TTI values and the GPS coordinates of the associated loop detector locations and displays the information visually on a map with a time slider.

6.1 Bottleneck analysis

The R code for this analysis can be found in the Appendix. The speed and volume data for the month of July 2017 used here were recorded by loop detectors that are embedded in the road on the 400-series highway network. The loop detectors are just below the surface of the asphalt and can sense vehicles passing over them. They record speed and volumes.

The first part of doing a bottleneck analysis is to determine a threshold speed below which traffic conditions are those of congestion and above which traffic conditions are comfortable. The threshold selected is also referred to as the Maximum Throughput Speed (MTS), which is the average speed of vehicles at capacity, or at the maximum flow rate of vehicles per hour per lane. The reasoning follows that of Scott Johnson's in that it is closer to the threshold at which drivers perceive congestion. According to the empirically-derived formula from the Highway Capacity Manual, this value depends on the relationship between free flow speed, the highway characteristics (lane widths, presence of medians, etc..) and the corresponding value of flow rate at capacity, both of which could be determined theoretically using the empirical model shown in Figure 12: Highway capacity manual guide to determine speed at capacity; relevant values underlined:



Note:
Capacity varies by free-flow speed. Capacity is 2400, 2350, 2300, and 2250 pc/h/ln at free-flow speeds of 120, 110, 100, and 90 km/h, respectively.
For $90 \le FFS \le 120$ and for flow rate (v_p)
$(3100 - 15FFS) < v_p \le (1800 + 5FFS)$,

$$S = FFS - \left[\frac{1}{28}(23FFS - 1800)\left(\frac{v_p + 15FFS - 3100}{20FFS - 1300}\right)^{2.6}\right]$$

For $90 \le FFS \le 120$ and
$v_p \le (3100 - 15FFS)$,
$S = FFS$

Figure 12: Highway capacity manual guide to determine speed at capacity; relevant values underlined [6]

$$S = FFS - [\frac{1}{26}(23 * FFS - 1800)\left(\frac{v_p + 15 * FFS - 3100}{20 * FFS - 1300}\right)^{2.6}]$$

Using the formula above for MTS (S) [6], a free-flow speed (FFS) of 115 km/h (the average of recorded speeds during overnight periods), and a capacity (v_p) of 2375 pc/h/ln, MTS calculations for a handful of segments for Highway 404 and Highway 401 yielded values hovering around 85 km/h. This is the value used as the threshold for all data analyzed. To confirm this threshold value the speed and volume data from 5 loop detectors, randomly selected, (410DW1040DSS, 401DW0030DES, 400DN0010DSE, 401DE0350DWE, and 404DN0060DNS) were used to generate the Speed-Flow fundamental diagram shown in Figure 13 where the 85th percentile speed is 120 km/h and the 85th percentile flow rate is 1780 veh/h/ln. The resulting MTS is around the 85 km/h mark as was found using the empirically-derived formula above.

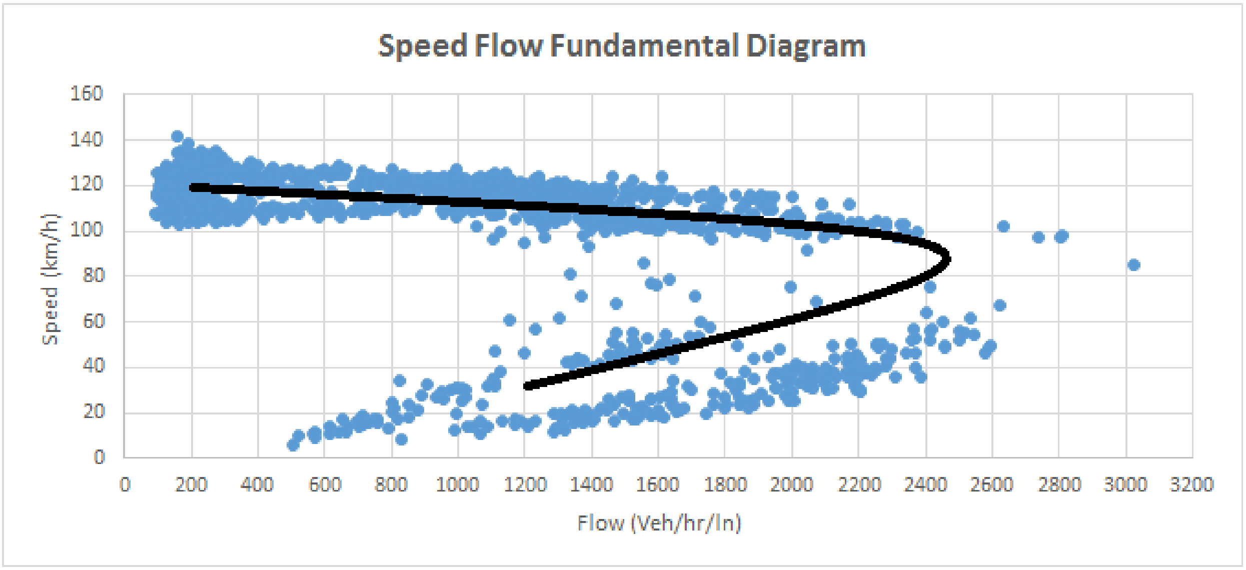


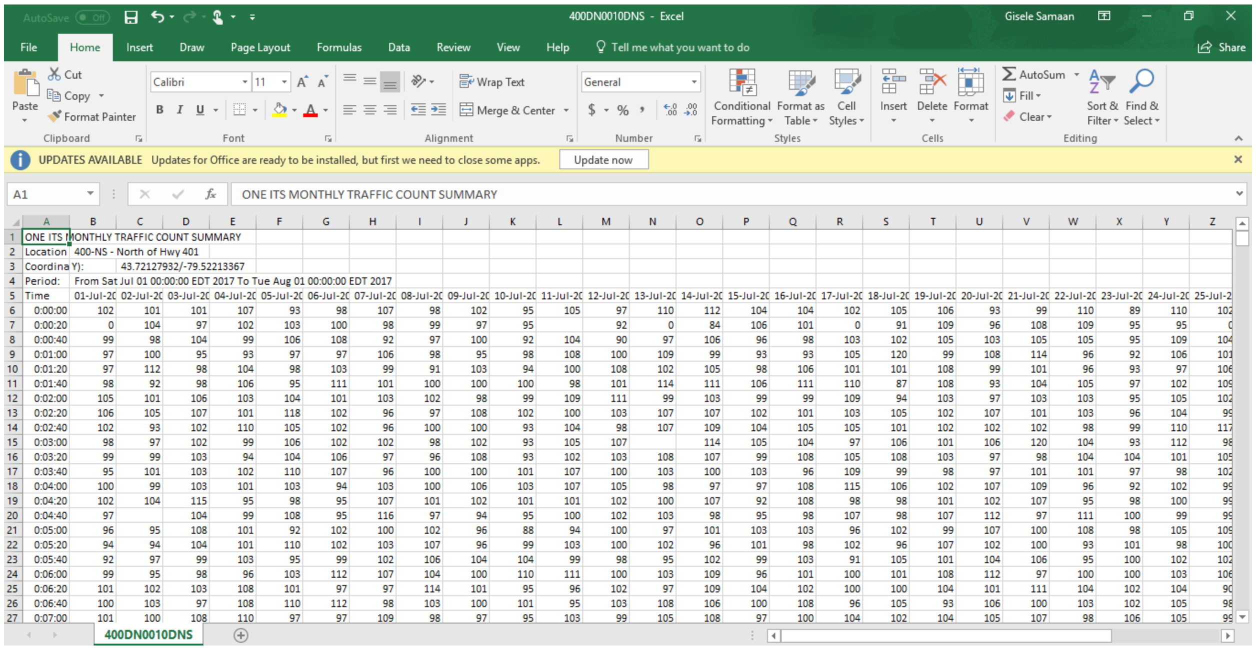
Figure 13: Speed Flow diagram from loop detector data at 5 locations

Loop detector speed data from Highway 404, Highway 401, Highway 410, Highway 409, QEW/Highway 403, with around 300 highway segments have been evaluated so far for entire month of July. One highway segment is represented by one loop detector location. The indicator for congestion is the Travel Time Index, determined for each segment independently as:

$$TTI = \frac{\frac{1}{Actual\ Speed\ in\ km/h}}{\frac{1}{85\ km/h}}$$

The denominator in this formula is the inverse of the MTS. The numerator is found in the speed profile. Alternatively, it is the ratio between the actual travel time and travel time at the Maximum Throughput Speed. The higher the index, the worse the congestion, i.e. an index of 2 means that the drive is taking you twice as long as if the traffic were moving at 85 km/h.

The calculation procedure starts with reading the .csv file of raw loop detector data, a sample of which is pictured in Figure 14. The data is read, cleaned, and analyzed for each day separately. As mentioned in the literature review, loop detectors are unreliable devices in the way that they sometimes record faulty data or turn off for anywhere from a minute to several days. Any day-long loop detector data that is missing over half the values is discarded from analysis. For all others, a moving-average data imputation technique, in the form of the function na.ma() from the ImputeTS R package, is used to interpolate the missing values.


400DN0010DNS - Excel
Gisele Samaan
File Home Insert Draw Page Layout Formulas Data Review View Help Tell me what you want to do Share
UPDATES AVAILABLE Updates for Office are ready to be installed, but first we need to close some apps. Update now
A1
ONE ITS MONTHLY TRAFFIC COUNT SUMMARY
400DN0010DNS

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	ONE ITS MONTHLY TRAFFIC COUNT SUMMARY																									
2	Location	400-NS - North of Hwy 401																								
3	Coordina	Y):	43.72127932/-79.52213367																							
4	Period:	From Sat Jul 01 00:00:00 EDT 2017 To Tue Aug 01 00:00:00 EDT 2017																								
5	Time	01-Jul-20	02-Jul-20	03-Jul-20	04-Jul-20	05-Jul-20	06-Jul-20	07-Jul-20	08-Jul-20	09-Jul-20	10-Jul-20	11-Jul-20	12-Jul-20	13-Jul-20	14-Jul-20	15-Jul-20	16-Jul-20	17-Jul-20	18-Jul-20	19-Jul-20	20-Jul-20	21-Jul-20	22-Jul-20	23-Jul-20	24-Jul-20	25-Jul-2
6	0:00:00	102	101	101	107	93	98	107	98	102	95	105	97	110	112	104	104	102	105	106	93	99	110	89	110	102
7	0:00:20	0	104	97	102	103	100	98	99	97	95		92	0	84	106	101	0	91	109	96	108	109	95	95	0
8	0:00:40	99	98	104	99	106	108	92	97	100	92	104	90	97	106	96	98	103	102	105	103	105	105	95	109	104
9	0:01:00	97	100	95	93	97	97	106	98	95	98	108	100	109	99	93	93	105	120	99	108	114	96	92	106	101
10	0:01:20	97	112	98	104	98	103	99	91	103	94	100	108	102	105	98	106	101	101	108	99	101	96	93	97	106
11	0:01:40	98	92	98	106	95	111	101	100	100	100	98	101	114	111	106	111	110	87	108	93	104	105	97	102	109
12	0:02:00	105	101	106	103	104	101	103	102	98	99	109	111	99	103	99	99	109	94	103	97	103	103	95	105	102
13	0:02:20	106	105	107	101	118	102	96	97	108	102	100	103	107	107	102	101	103	105	102	107	101	103	96	104	99
14	0:02:40	102	93	102	110	105	102	96	100	100	93	104	98	107	109	104	105	105	101	102	102	102	98	99	110	117
15	0:03:00	98	97	102	99	106	102	102	98	102	93	105	107		114	105	104	97	106	101	106	120	104	93	112	98
16	0:03:20	99	99	103	94	104	106	97	96	108	93	102	103	108	107	99	108	105	108	103	97	98	104	104	101	105
17	0:03:40	95	101	103	102	110	107	96	100	100	101	107	100	103	100	103	96	109	99	98	97	101	101	97	98	102
18	0:04:00	100	99	103	101	103	94	103	100	106	103	107	105	98	97	97	108	115	106	102	107	109	96	92	102	99
19	0:04:20	102	104	115	95	98	95	107	101	102	101	101	102	100	107	92	108	98	98	101	102	107	95	98	100	99
20	0:04:40	97		104	99	108	95	116	97	94	95	100	102	103	98	95	98	107	98	107	112	97	111	100	99	99
21	0:05:00	96	95	108	101	92	102	100	102	96	88	94	100	97	101	103	103	96	102	99	107	100	108	98	105	109
22	0:05:20	94	94	104	101	110	102	103	107	96	99	103	100	102	96	101	98	102	96	107	102	100	93	101	98	100
23	0:05:40	92	97	99	103	95	99	102	106	104	104	99	98	95	102	99	103	91	105	101	104	106	95	100	102	102
24	0:06:00	99	95	98	96	103	112	107	104	100	110	111	100	103	109	96	101	100	101	108	112	97	100	100	103	106
25	0:06:20	101	102	103	108	101	97	97	114	101	95	96	102	97	109	104	102	100	100	104	101	111	104	102	104	90
26	0:06:40	100	103	97	108	110	112	98	103	100	101	95	103	108	106	100	108	96	105	93	106	100	103	102	105	98
27	0:07:00	101	100	108	110	97	97	109	98	97	95	103	99	105	108	97	100	104	102	104	105	107	98	106	105	99

Figure 14: Snippet of speed data from a loop detector; coordinates in top left; speed data recorded every 20 seconds; each column is one 24-hour period

The cleaned data is then averaged over 2-minute intervals. The 2-minute interval length is a good balance between reducing the noisy data and the need for short intervals to animate the evolution of congestion bottlenecks for the visualization part of this thesis. The speed value for each 2-min interval is used to calculate the Travel Time Index of that 2-minute interval, i.e. used in the numerator of the TTI formula above. An example of the steps of this analysis is illustrated in Figure 15 with one segment's speed profile and Travel Time Index. The left side shows the location of that detector, northbound side of Highway 404, north of Sheppard Ave. The graph on the top right is the speed profile for July 5 for that location where the 2-min speed averages are plotted. The red curve is the result of Loess curve fitting function in R and the blue line is the 85 km/h threshold speed. The graph directly below is the TTI profile of the same location on the same day. As can be seen, the periods of time where the speed measurements dip below the threshold correspond with the morning and evening rush hours where the TTI values are above 1.

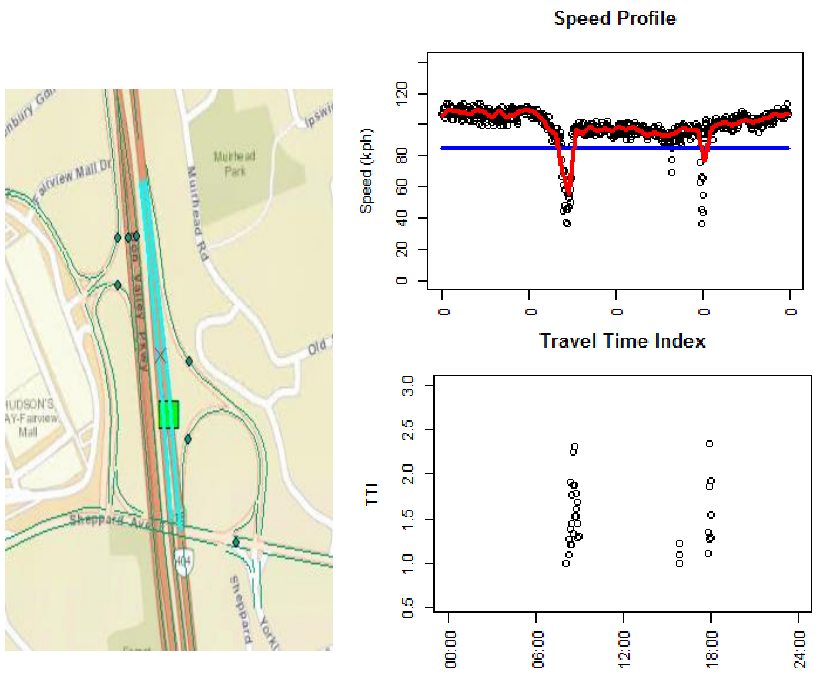
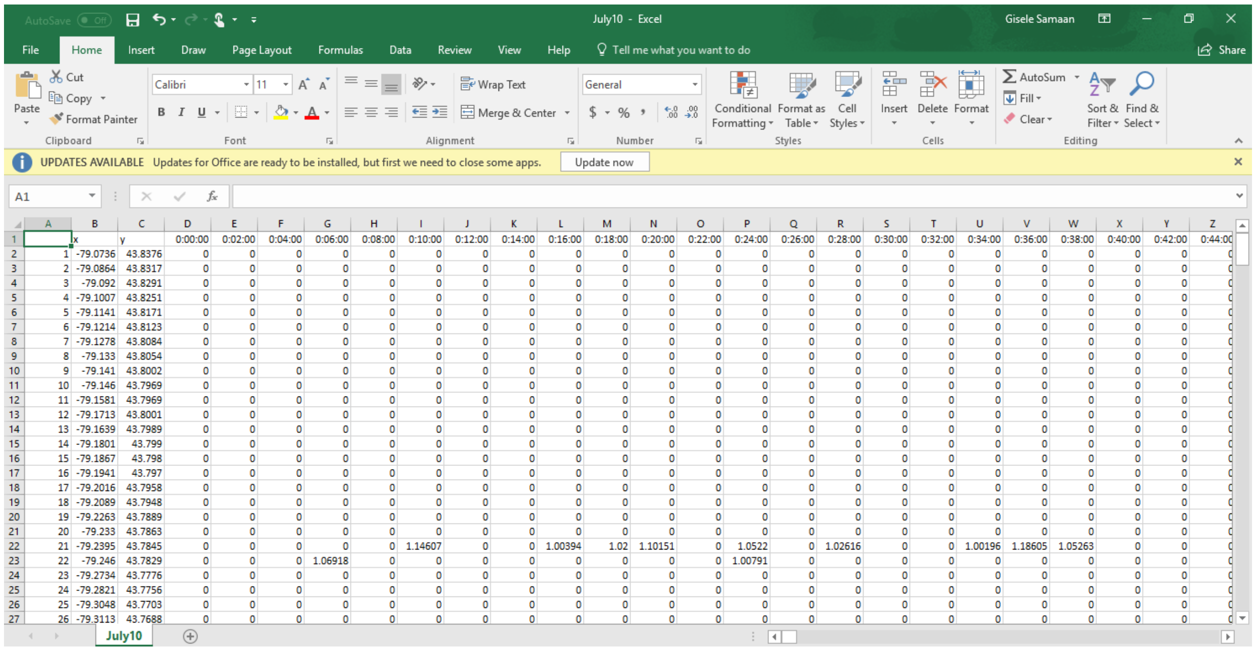


Figure 15: NB of Hwy 404 near Fairview Mall; green square is loop detector location; speed profile top right; travel time index bottom right

The TTI is inversely proportional to speed values, meaning that traffic speed below the threshold have a TTI value of more than 1. The slowest speeds correspond to the highest TTI values as the formula implies. There is a total of 720 data points for the one-day period. This result, along with the coordinates for the corresponding loop detector written into a .csv file, one file per day containing all the loop detector locations with their TTI values. A snippet of the output for July 10 is shown in Figure 16. TTI values above one during the early hours of the morning are a result of either an incident at that time or a malfunctioning loop detector that records slower speeds. Further statistical analysis is required with at least a year’s worth of data to confidently eliminate these points as outliers from the analysis. This output becomes the input to the second part of this thesis.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1		x	y	0:00:00	0:02:00	0:04:00	0:06:00	0:08:00	0:10:00	0:12:00	0:14:00	0:16:00	0:18:00	0:20:00	0:22:00	0:24:00	0:26:00	0:28:00	0:30:00	0:32:00	0:34:00	0:36:00	0:38:00	0:40:00	0:42:00	0:44:0
2	1	-79.0736	43.8376	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	2	-79.0864	43.8317	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	3	-79.092	43.8291	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	4	-79.1007	43.8251	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	5	-79.1141	43.8171	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	6	-79.1214	43.8123	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	7	-79.1278	43.8084	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	8	-79.133	43.8054	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	9	-79.141	43.8002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	10	-79.146	43.7969	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	11	-79.1581	43.7969	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	12	-79.1713	43.8001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	13	-79.1639	43.7989	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	-79.1801	43.799	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	15	-79.1867	43.798	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	16	-79.1941	43.797	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	17	-79.2016	43.7958	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	18	-79.2089	43.7948	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	19	-79.2263	43.7889	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	20	-79.233	43.7863	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	21	-79.2395	43.7845	0	0	0	0	0	1.14607	0	0	1.00394	1.02	1.10151	0	1.0522	0	1.02616	0	0	1.00196	1.18605	1.05263	0	0	0
23	22	-79.246	43.7829	0	0	0	1.06918	0	0	0	0	0	0	0	0	1.00791	0	0	0	0	0	0	0	0	0	0
24	23	-79.2734	43.7776	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	24	-79.2821	43.7756	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	25	-79.3048	43.7703	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	26	-79.3113	43.7688	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16: Snippet of output for July 10; longitudes of loop detectors in column 'x'; latitudes in column 'y'; one 2-min interval per column

6.2 Interactive Web Application Development

The R code for this application can be found in the Appendix (app.R). It uses the R Shiny package that contains the code for both the User Interface and Server functions (front-end and back-end). It is accompanied by a directory, "TTI", containing each loop detector's TTI file of 2-minute averages and ArcGIS-generated shapefiles of hourly TTI averages for each day of July. The ArcGIS-generated shapefiles were created using the following procedure:

1. The 2-minute TTI averages were grouped into hourly TTI averages for each detector
2. Loop detectors are divided by highway and direction, e.g. Highway 404 northbound, QEW EB, etc.
3. Each group's hourly TTI averages csv file is imported into ArcGIS as "Events." Each Event layer contains one day's worth of TTI values (hourly average) for one highway section
4. Each group's loop detector dots are manually connected by a Polyline that follows the highway direction
5. A Buffer is generated of that line that is 40 metres wide, on the right side

6. The TTI values for each hourly interval undergoes spatial interpolation using Bayesian Kriging Interpolation tool, the only option in ArcGIS that generates shapefiles that are easily read by the R shiny package
7. The resulting variogram is "Clipped" using the Clip tool and the Buffer as the "Clipper" so that the spatial interpolation is restricted to the geographical space of the highway
8. This procedure is repeated for every hour for all highway sections for the day, over the 31 days of July
9. All the resulting "Clipped" shapefiles are put together to end up with one shapefile containing the hourly average TTI for the entire network. 24 of these are generated for each day
10. The shapefiles are placed in the "TTI" file to be read by the R Shiny script and displayed visually

The Python script that automates this process using the arcpy library, created by McMaster University PhD student Anastassios Dardas is found in the Appendix along with the R script that transforms the .csv input file of TTI values (so far hourly averages) into point shapefiles (to replicate Step 2 in the automation process). This point shapefile is the input to the Python script.

The interactive web application is currently available to the public via the following web address: http://itsos.ca/Traffic_Bottleneck.php on the Intelligent Transportation System of Systems group's website. The application has two panels. The top panel has a map that displays the Travel Time Index at each loop detector location during the month of July 2017 for a 2-minute interval. The map is accompanied by a legend to explain the colour-coding pattern: green for travel speeds below threshold, yellow for mild congestion (TTI from 1 to 1.5) of up to 50% longer travel times, orange for moderate congestion (TTI from 1.5 to 2) of 50%-100% longer travel time, and red for severe congestion (TTI > 2). Below the map is a 24-hour time slider where each tick represents a 2-minute interval. The user can select the 2-minute interval TTI values to be displayed by clicking anywhere on the slider. The user can also press the "Play" button on the right to activate the animation. This animation shows the evolution of traffic conditions on the network and can be combined with other information such as, for example, evolving weather conditions to visualize the relationship between traffic congestion and weather. This information could prove useful to emergency services to help determine the best routes to take by predicting whether traffic conditions may quickly deteriorate in the route ahead as a storm approaches. This would be the subject of future work. Below the time slider is a drop-down menu the user can use to select the date to display. A screenshot can be found in Figure 17 for the latest version:

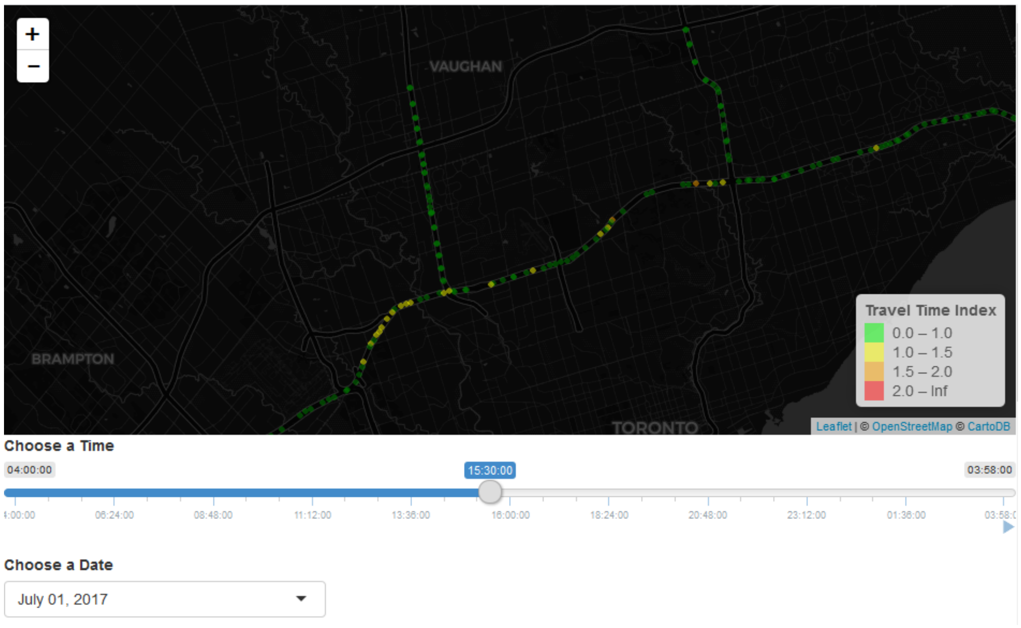


Figure 17: Screenshot of the top panel of the web application showing 2-minute TTI averages

This panel displays the base map CartoDB.DarkMatter layer available using Leaflet maps. The user can zoom in and out as they please. The second panel, below the first, displays the ArcGIS-generated spatially interpolated TTI hourly averages values along the highway network. Again, the map has similar functions as the one for the first panel (24-h time slider with animation capability, but with 24 ticks; drop-down menu to select the date; colour-coding pattern). A screenshot is shown in Figure 18 below. The base map displayed is Esri.WorldGrayCanvas layer available for Leaflet maps.

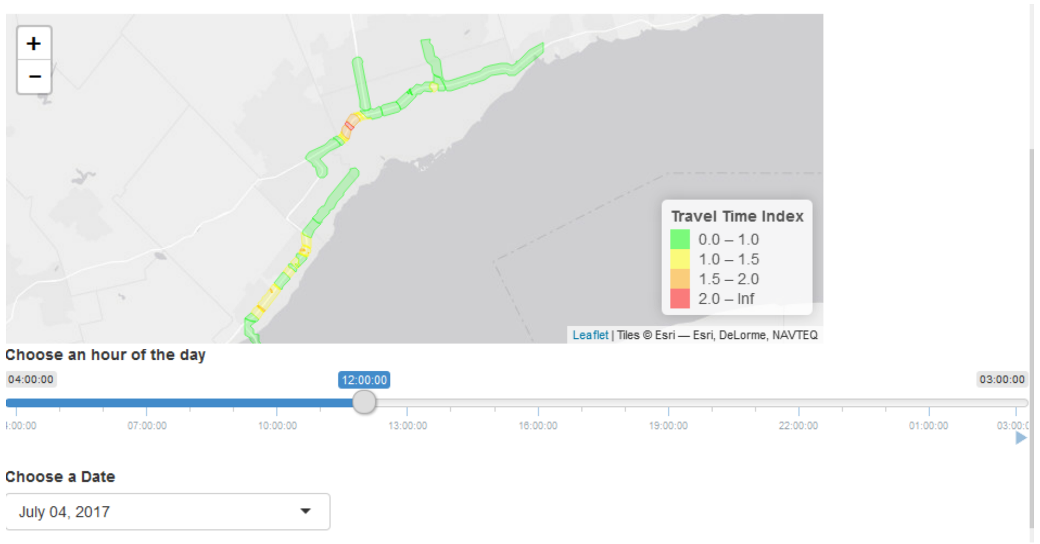


Figure 18: Screenshot of second panel showing ArcGIS-generated visualization of spatially interpolated hourly TTI averages

6.2.1 Web Application Documentation Outline

To be useful to the user, this application will be accompanied by documentation that will be easily accessible through a button that will be added soon to the application. The documentation will have two parts. The first part will mirror the section above that explains each part of the app. This part will also have elements of the Bottleneck Analysis section to explain the origin of the information being displayed. The second part will be akin to a manual, a how-to series of steps with pictures or video to show the user how to interpret the visualized data. An outline is shown below:

- Purpose of the application: Big Data visualization, information directly from infrastructure
- Data source and data analysis process:
    - Loop detectors and their limitations; July 2017 displayed
    - Data cleaning techniques applied
    - Bottleneck analysis process with the R script for reference
- Description of the application
- How-to series of steps
    - Panning to see other parts of the map
    - Zooming in and out
    - Reading and selecting time periods

- Selecting days of July to display
- Animation process
- Interpreting Panel 1: best for studying bottleneck evolution
- Interpreting Panel 2: best for studying recurrent and non-recurrent bottlenecks

Most of the information for the documentation already exists in this thesis and will be rearranged in the above order.

7 Results and Validation

Besides the validation of the empirically-derived formula for speed at capacity from the Highway Capacity Manual, the highway segments of recurrent bottlenecks, that are most visible with the animation of the second panel, confirm observations done elsewhere. The CAA released a report done by CPCS Transcom Limited [7] that analyzed traffic data of highways all over Canada to determine which had the highest delays. The top two locations according to this study were: Highway 401 between Highway 427 and Yonge St, and Highway 404/DVP between Don Mills and Finch Ave. For this thesis, and as visualized in the web application, those two locations are also the ones where the highest TTI's (in the red) are displayed the most.

Also, higher (yellow, red) TTI values appear during AM and PM rush hour periods throughout the network, but particularly in dense urban areas such as the 401 in Toronto and the QEW in Mississauga, Oakville, and Burlington especially on weekdays. Intuitively, this follows the typical travel congestion patterns. Weekends tend to record more "green" travel conditions (no delay compared to travelling at 85 km/h) with mild congestion spots coalescing around major highway interchanges, particularly 401 and 400, 401 and 404, and 401 and Yonge St. Interestingly, the north end of Highway 400 tends to have more severe congestion than the south end in off-peak hours and evening peak period alike for at least 12 of the weekdays in July. Further study is required to determine whether the 407 traffic plays a role or the loop detectors at the north end are malfunctioning by recording lower speeds than the actual speeds.

8 Future Work

- Data cleaning. This part will be a major next step as the moving average imputation only works on data sets that have at least 60% viable data, or maximum of 40% of all data points are missing. Here, the imputation techniques from the paper discussed in the Literature Review section will be used as a reference for methodology. A greater amount of data would be needed to impute the missing chunks of data from days that have been discarded in this thesis. About 30%-40% of all the loop detector data for this thesis has been discarded for missing too many values. One method could be:
    - First, do moving average imputation for as many valid data as possible, as done for this thesis
    - Then, for the missing days, perform a Principal Component Analysis imputation for each loop detector location for:
        - All non-holiday weekdays
        - Holidays
        - Saturdays
        - Sundays

    Where each 20s the period is analyzed individually across each type listed above. This method should follow the one used by Ke et. al. when using loop detector data in Hong Kong to predict real-time likelihood of accidents [8]. For example, for non-holiday weekdays, for each 20s period, each weekday's value at that period is analyzed for its influence in the variation of Principal Component 1. Then, each measurement's PC1 value is obtained by multiplying the influence factor by its recorded speed value. A probabilistic relationship between this PC1 value and the original speed measurement should be determined. Then the imputed speed value should be the one corresponding with the expected PC1-to speed relationship. Data from at least two weeks prior to and two weeks after July would be needed to get accurate values, especially for Saturdays and Sundays.

- Expansion: getting loop detector information from the City of Toronto for the Gardener/DVP would complete the network and provide more depth in understanding whether and how congestion levels evolve in areas with different land uses.

- Replacing the 2-minute TTI averages “dot” visuals with the ArcGIS-equivalent, like the second panel. So far attempts to generate these maps have resulted in computation complexities that indicate 6-8 weeks of continuous processing. A more powerful computer is being sought.
- Processing data beyond the month of July 2017.
- Real-time data processing and visualization. This would require new back-end architecture currently in the works by the ITSoS group.
- Cosmetic changes to fit the theme of the website that will include other services. This web service is part of a broader ITSoS framework to provide useful visualization for researchers. A JavaScript-based aesthetic is being developed by the ITSoS group.
- Determining the extra travel time based on TTI. This will require manually dividing the highway network into 1000+ segments, each with a loop detector in the middle to obtain a corresponding distance value used to then calculate time delay.

9 Conclusion

Big Data visualization can be a powerful tool to inform public policy direction and help planners see the system they deal with differently. This thesis is the beginning of a process that will add depth in terms of the type of analysis performed and breadth in terms of amount of information to be analyzed. Thus far, one can infer congestion patterns, recurrent and non-recurrent, based solely on visual observation through the application designed here. This work is but a sample of the powerful simplification of the understanding of a Big Data set that comes with visualization. The work done here lays the foundation for a more sophisticated web-service-based traffic congestion visualization application.

10 References

[1] M. Elshenawy, "A Three-Pillar Methodology and Framework for Seamlessly Integrated Cyber-Physical Intelligent Transportation System of Systems," 2016.

[2] M. Elshenawy, B. Abdulhai and M. El-Darieby, "Towards a Seamlessly Integrated Cyber-Physical Intelligent Transportation System of Systems," in *ITS World Congress*, Montreal, 2017.

[3] C. Bachmann, "Multi-Sensor Data Fusion for Traffic Speed and Travel Time Estimation," 2011.

[4] S. Johnson, "Integrated Congestion Evaluation and Quantification," Toronto, 2005.

[5] M. Elshenawy, M. El-Darieby and B. Abdulhai, "Automatic Imputation of Missing Highway Traffic Volume Data".

[6] Transportation Research Board, Highway Capacity Manual, Washington, D.C.: National Research Council, 2000.

[7] CPCS Transcom Limited, "Grinding to a Halt: Evaluating Canada's Worst Bottlenecks," CAA, January 2017. [Online]. Available: https://www.caa.ca/wp-content/uploads/pdfs/en/16170_Canadian_National_Bottlenecks_Study_EN_1_4_17.pdf. [Accessed 17 November 2017].

[8] J. Ke, S. Zhang, H. Yang and X. (. Chen, "PCA-Based Missing Information Imputation for Real-Time Crash Likelihood Prediction Under Imbalanced Data," 11 February 2018. [Online]. Available: https://arxiv.org/pdf/1802.03699.pdf. [Accessed 7 April 2018].

11 Appendix

The lines beginning with ‘#’ are comments that describe the code and do not participate in its functioning. The Bottleneck Analysis code is flexible enough to process two or more loop detectors that are in one location, to take their averages to produce the speed profile.

11.1 R Code for Bottleneck Analysis

```
# Bottleneck analysis, input raw loop detector data for July 2017,
# output TTI 2-minute averages

library(imputeTS)
library(dplyr)



for (July in 1:31){

# Set up the time format

travel <- data.frame("time" = c(paste("7/",July,"/17 00:00",sep=""),
                                paste("7/",July,"/17 23:58",sep="")),
                     stringsAsFactors = F) %>%
  mutate(
    time = as.POSIXct(time, format = "%m/%d/%y %H:%M"))

# Initiate objects in which to store final output with long and lat of loop detectors

output <- data.frame(row.names=seq(as.POSIXct(paste("2017/7/",July," 00:00:00",sep="")),
                         as.POSIXct(paste("2017/7/",July," 23:58:00",sep="")),
                         "2 mins"))
latitude <- vector(mode="character",length=0)
longitude <- vector(mode="character",length=0)

# Initialize the speed profile plot
              par(mfrow = c(2,2))

# Read the data files of detectors - User Input

#####
files <- list.files(path="D:\\Thesis\\404\\404NB", full.names=T, recursive=FALSE)
#####

# Read each segment in sequential order, mirroring the geographic sequence

for (folder in files){
  par(mfrow = c(2,3))
  SegmentSP <- vector(mode="numeric",length=720)
  minute <- vector(mode="character",length=720)
  Baseline <- rep(85,720)
  file=1
  f <-list.files(path=folder, pattern="*.csv", full.names=T, recursive=FALSE)
  Day <- matrix(nrow=length(f),ncol=720)

# Read loop detector csv data file

for (i in f) {
  print(i)
   detector = read.table(i, header=FALSE, sep=",", fill=TRUE, stringsAsFactor=FALSE)
               time <- detector[,1]
               if (grepl("/",detector[3,3]) == FALSE){ # Some faulty data files missing coordinates
                 next
               }
               coordinates <- unlist(strsplit(detector[3,3],"/"))
               detector <- detector[-c(1,2,3,4,5),-c(1)]
               day <- as.numeric(detector[,July])
               day[day == 0] <- NA
               detector[,July] <- day

  # Data Moving Average Imputation
               detector[1,July]=100 # Speed at midnight should be free-flowing
               if (class(try(na.ma(as.numeric(detector[,July])),silent=T)) == "try-error" )
                 next
               detector[,July] <- na.ma(as.numeric(detector[,July]))

              # Average 24-hr data over 2 min for each file and store in a matrix
              a=1
              for(n in seq(from=1, to=length(detector[,July]), by=6)) {
                          Day[file,a] = mean(as.numeric(detector[n:(n+5),July]),na.rm=FALSE)
                          minute[a] <- substr(time[n+5],1,5)
                           a=a+1
              }

              # Check code by plotting speed profile
              plot(1:length(Day[file,]),Day[file,], xaxt = 'n',xlab="",ylab="Speed (kph)",ylim=c(0,140))
              c = minute[seq(1, length(minute),100)]
              axis(1, at=seq(1,length(minute),100), labels = c, las=2)
              lines(1:length(Day[file,]),Baseline,col="blue",lwd=2)

               file=file+1
}


# Compute the average over the segment
Values = vector(mode="numeric",length=length(folder))
for (num in 1:length(SegmentSP)) {
              Values = Day[,num]
              SegmentSP[num] = mean(Values)
}

# Plot speed profile

x = 1:length(minute)
y = SegmentSP

plot(x,y, xaxt = 'n', xlab = '', ylab="Speed (kph)",ylim=c(0,140),main="Speed Profile")

c = c(minute[seq(1, length(minute),180)],"24:00")
axis(1, at=seq(1,(length(minute)+180),180), labels = c, las=2)
lines(x,Baseline,col="blue",lwd=3)
SP <- loess.smooth(x,y,span=0.01)
lines(SP,col="red",lwd=3)

# Moving average imputation

if(is.na(mean(SegmentSP))) # Empty day, so skip
  next


# Travel time index

TTI = vector(mode="numeric",length=720)

# Calculate TTI using MTS of 85 km/h
index = 0
for (index in 1:length(SegmentSP)) {
  if (SegmentSP[index] <= Baseline[index]){
                          TTI[index] = (1/SegmentSP[index])/(1/85)
                          if (!is.finite(TTI[index])){
                            TTI[index] = 0
                          }
  }
  else{
    TTI[index] = 0
  }
}

# Check calculations by plotting TTI over 24h period

plot(1:length(TTI), TTI,xaxt = 'n', xlab="", ylim=c(0.55,3), main="Travel Time Index")
c = c(minute[seq(1, length(minute),180)],"24:00")
axis(1, at=seq(1,(length(minute)+180),180), labels = c, las=2)

# Add to final output

output = cbind(output,TTI)
latitude = c(latitude,coordinates[1])
longitude = c(longitude,coordinates[2])
print("done")
```

```
}

# Write csv of 24h TTI with coordinates - User Input

#####
toMap <- data.frame(x=longitude,y=latitude,z=t(output))
write.csv(toMap,file=paste("D:\\Thesis\\App\\ProcessedData\\JulyE\\TTI_400SB\\400_coordinatesSB",".csv",sep="
"))
#####

}
```

11.2 R Code for Web Application Using R Shiny Package

```
library(shiny)
library(dplyr)
library(leaflet)
library(dbplyr)
library(sp)
library(gtools)
library(rgdal)
library(rsconnect)



ui <- bootstrapPage({

  # Set the time range over which data is displayed
  travel <- data.frame("time" = c("00:00", "23:58"),
                       stringsAsFactors = F)
  Time = seq(as.POSIXct("00:00", format = "%H:%M"),
             as.POSIXct("23:58", format = "%H:%M"),"2 mins")

  # Read and store TTI's, long's, and lat's in an accessible dataframe, lists

  setwd("TTI")
  files <- list.files(path=getwd(), pattern="*.csv",
                      full.names=T, recursive=FALSE)
  files = mixedsort(sort(files))

  longitude=list()
  latitude = list()
  TTI.names <- format(seq(as.Date("2017-07-01"),
                          as.Date("2017-07-31"),
                          by="days"), "%B %d, %Y")
  TTI <- vector("list", length(TTI.names))
  names(TTI) <- TTI.names
  names(TTI) = format(seq(as.Date("2017-07-01"),
                          as.Date("2017-07-31"),
                          by="days"), "%B %d, %Y")
  i=1
  for (file in files) {
    TravelIndex = read.csv(file)
    TTI[[i]] = TravelIndex[,4:723]
    colnames(TTI[[i]]) = substr(Time,12,16)
    latitude[[i]] = as.numeric(TravelIndex[,3])
    longitude[[i]] = as.numeric(TravelIndex[,2])
    i=i+1
  }

  },

  tags$body(
    # Panel 1, 2-minute averages
                  leafletOutput('map', width = "100%", height = "400px"),
                  sliderInput(inputId="num", label="Choose a Time",
                              value=as.POSIXct("11:30", format = "%H:%M"),
                              min=as.POSIXct(travel$time[1],format = "%H:%M"),
                              max=as.POSIXct(travel$time[2],format = "%H:%M"),
                              step = 120, round=FALSE, ticks=TRUE,
                              animate = animationOptions(interval=3000), timeFormat = "%T",
                              timezone = "-0000", width = "100%"),
              selectInput("date", "Choose a Date", choices = format(seq(as.Date("2017-07-01"),
                                                      as.Date("2017-07-31"),
                                                      by="days"), "%B %d, %Y")),
                  # Panel 2, hourly averages
                  leafletOutput('hours',width = "80%", height = "300px"),
                  sliderInput(inputId="hour", label="Choose an hour of the day",
                              value=as.POSIXct("08:00", format = "%H:%M"),
                              min=as.POSIXct("00:00", format = "%H:%M"),
                              max=as.POSIXct("23:00", format = "%H:%M"),
                              step = 3600, round=FALSE, ticks=TRUE,
                              animate = animationOptions(interval=3000), timeFormat = "%T",
                              timezone = "-0000", width = "100%"),
                  selectInput("date2", "Choose a Date", choices = format(seq(as.Date("2017-07-01"),
                                                          as.Date("2017-07-31"),
                                                          by="days"), "%B %d, %Y"))
))

server <- function(input,output) {

  output$map <- renderLeaflet({

                  # Plot map
                  leaflet() %>%
                  addProviderTiles(providers$CartoDB.DarkMatter) %>%
          setView(lng = -79.48, lat = 43.75, zoom = 11)
  })

  observe({
    ## For 2 min intervals
    # Create Bubbles
    day = match(input$date,names(TTI))

    data=data.frame(long=longitude[[day]] , lat=latitude[[day]],
                  val=TTI[[day]][,match(substr((input$num),12,16),colnames(TTI[[day]]))])


    # Add circle reactives
    pal = colorBin(c("green","yellow","orange","red"), data$val, bins = c(0,1,1.5,2,Inf))

    leafletProxy('map') %>%
          clearMarkers() %>% clearControls() %>%
           addCircleMarkers(data$long, data$lat, radius = 0,
                    color = ifelse(data$val < 1, "green",
                              ifelse(data$val < 1.5, "yellow",
                                 ifelse(data$val < 2, "orange","red"))),
                    fillOpacity=0.5, popup=as.character(round(data$val,digits=1))) %>%
          addLegend("bottomright", pal = pal, values = data$val, title = "Travel Time Index")
  })


  # Hourly averages map
  output$hours <- renderLeaflet({

    shp <-readOGR(".", ifelse(as.numeric(substr(input$date2,6,7))>=10,
                        paste("July",substr(input$date2,6,7),"_X",substr((input$hour),
                                                      12,13),
                           "_",substr((input$hour),15,16),"_clip",sep=""),
                        paste("July",substr(input$date2,7,7),"_X",substr((input$hour),
                                                      12,13),
                           "_",substr((input$hour),15,16),"_clip",sep="")))
    pal2 = colorBin(c("green","yellow","orange","red"), shp$Value_Max, bins = c(0,1,1.5,2,Inf))

    # Plot second map
    leaflet(shp) %>%
      addProviderTiles(providers$Esri.WorldGrayCanvas) %>%
      setView(lng = -79.4, lat = 43.6, zoom = 9) %>%
      addPolygons(weight = 1, color = ~pal2(Value_Max)) %>%
      addLegend("bottomright", pal = pal2, values = shp$Value_Max, title = "Travel Time Index")
  })

}


shinyApp(ui, server)
```

11.3 R Code for CSV -> Point Shapefiles

```
library("readr")
library("rgdal")

###############################################################################################
####### Author: Anastassios Dardas
# Function: set_wd_to_cur_dir
#
# Purpose: Sets working directory of script to file, returns the path
#
# Caution: if you have a network mapped drive, this may have issues
###############################################################################################
#######
set_wd_to_cur_dir <- function(){
  this_directory <- dirname(sys.frame(1)$ofile)
  setwd(this_directory)
  return(this_directory)
}

###############################################################################################
#######
# Function: gen_leading_zeros
#
#  Purpose: Generate n number of leading zeros so that the output is a fixed size
#  Parameters:  i    -> number
#               size -> size of output string
#
#  Examples:
#       Params: i = 1; size = 5;
#       Output: "00001"
#       Params: i = 23; size = 5;
#       Output: "00023"
#       Params: i = 1023; size = 10;
#       Output: "0000001023"
###############################################################################################
#######
gen_leading_zeros <- function(i, size) paste(paste(rep("0", size-nchar(i)), collapse=""), i, sep="")

###############################################################################################
#######
# Function: rm_file_if_exist
#
#  Purpose: remove a file if it already exists at the specified path
#  Parameters:  fn -> file path
###############################################################################################
#######
rm_file_if_exist <- function(fn)(if(file.exists(fn)){file.remove(fn)})

###############################################################################################
#######
# Function: csv_to_ogr
#
#  Purpose: Read a csv and convert it to GIS readable data
#  Parameters:  ifile_csv -> Input file csv to read
```

```
#		ifile_dir -> Input directory
#		ofile_dir -> Output directory
##########################################################################################
#######*
csv_to_ogr <- function(ifile_csv, ifile_dir, ofile_dir, overwrite_ofile = TRUE){
  ofile_name <- gsub(".csv", "", ifile_csv)
  ifile_path <- paste(ifile_dir, ifile_csv, sep = "")

  ifile_df <- read_csv(ifile_path, col_types = cols(.default = "c"), col_names = FALSE, skip=1)[,c(2:27)]

  num_col_names <- sapply(0:23, function(i) paste(gen_leading_zeros(i,2), "_00", sep=""))

  colnames(ifile_df) <- c("lon","lat", num_col_names)

  ifile_df <- data.frame(lapply(ifile_df, function(x) as.numeric(as.character(x))))

  ifile_df$lon <- as.numeric(ifile_df$lon)
  ifile_df$lat <- as.numeric(ifile_df$lat)

  coordinates(ifile_df) <- ~lon+lat
  proj4string(ifile_df) <- CRS("+init=epsg:4326")

  # ofiles_ogr <- sapply(c(".shp", ".shx", ".dbf"), function(ext) paste(ofile_dir, ofile_name, ext, sep=""))
  # rm_status  <- sapply(ofiles_ogr, function(fn) rm_file_if_exist(fn))

  odsn <- paste(ofile_dir, ofile_name, ".shp", sep="")

  writeOGR(ifile_df, dsn = odsn, layer = ofile_name, driver = "ESRI Shapefile", overwrite_layer=TRUE)
}

this_dir <- set_wd_to_cur_dir()

ifile_dir <- "Gisele/csv_july/"
ofile_dir <- "Gisele/time_shapes/"

ifiles_csv <- list.files(ifile_dir, "*.csv")
create_status <- lapply(ifiles_csv, function(x) csv_to_ogr(x, ifile_dir, ofile_dir))
```

11.4 Python Code for ArcGIS-Generated Spatially-Interpolated Maps

```
##########################################################################################
#################
# Date: 4/2/2018
# Author: Anastassios (Tasos) Dardas
# Notes: This interpolates traffic flow of the highways in GTA / Hamilton region
        # Requirements: Must have ArcPy via valid ArcGIS license
        # May need to change the directory towards a relative
##########################################################################################
#################
import arcpy, glob
arcpy.env.workspace = "C:\\Users\\ZeusBayes\\Desktop\\Processed_Shapefiles" # Change the workspace directory

fields = ["X00_00", "X01_00", "X02_00", "X03_00", "X04_00", "X05_00", "X06_00", "X07_00", "X08_00",
"X09_00", "X10_00", "X11_00", "X12_00", “X13_00", "X14_00", "X15_00", "X16_00", "X17_00", "X18_00",
"X19_00", "X20_00", "X21_00", "X22_00", "X23_00"] # These fields may need to be changed

road_shp        = "C:\\Users\\ZeusBayes\\Desktop\\Gisele\\essential_shapes\\dissolve_network.shp" # Change where
the network shapefile is located
list_point_shps = glob.glob("C:\\Users\\ZeusBayes\\Desktop\\Gisele\\time_shapes\\*.shp") # Change where the list
of point shapes from the csv files are created

i = 0
for point_shp in list_point_shps:
        point_shp = str(point_shp.replace("\\", "/"))
        refine_shp = point_shp[46:len(point_shp)-4]
        print(point_shp)
        for field in fields:
                temp_layer = str(refine_shp + "_" + field)
                arcpy.EmpiricalBayesianKriging_ga(in_features = point_shp, z_field = field, out_ga_layer =
                temp_layer, out_raster = "", cell_size = "", transformation_type = "NONE", max_local_points =
                "", overlap_factor = "", number_semivariograms = "", search_neighborhood = "", output_type =
                "PREDICTION", quantile_value = "", threshold_type = "", probability_threshold = "",
                semivariogram_model_type = "LINEAR")
                arcpy.GALayerToContour_ga(str(refine_shp + "_" + field), contour_type = SAME_AS_LAYER",
                out_feature_class = str(refine_shp +  "_" + field + ".shp"), contour_quality = "DRAFT")
                arcpy.env.workspace = "C:\\Users\\ZeusBayes\\Desktop\\Processed_Clips_July"
                arcpy.Clip_analysis(str(refine_shp + "_" + field + ".shp"), clip_features = road_shp,
                out_feature_class = str(refine_shp + "_" + field + "_clip.shp"))
                print(i)
                i += 1
```