Annual Report
Transportation Data Research Laboratory
2004

Prepared by
Taek M. Kwon

Northland Advanced Transportation Research Laboratories

Summary

In FY 2003/2004, the Transportation Data Research Laboratory (TDRL) has finally achieved on-line automation for the two large sets of Mn/DOT's transportation data, i.e., the Transportation Management Center (TMC) traffic data, and the statewide Road Weather Information System (RWIS) data. This success was due to the development of a new data archiving technology referred to as the Unified Transportation Sensor Data Format (UTSDF) that allows efficient archiving of many types of transportation sensor data regardless of their data types. With this new technology, the users now need to learn only one type of data format to understand the entire TDRL's data archives. Presently, TDRL Data Center automatically acquires the RWIS and traffic data from Mn/DOT, archives them, and posts them on the Web for on-line distribution. This archived data set has not only been used by Mn/DOT researchers and engineers, but also has already been widely downloaded by researchers from other institutions and regions such as Texas Transportation Institute, California, Kentucky, Ohio, etc. Along with the raw data service, TDRL continues its processed data services for the Mn/DOT's Office of Transportation Data and Analysis (Mn/DOT TDA) providing real time data on continuous count (referred to as ATR) and short duration counts over 500 stations. These services are provided through on-line by TDRL in-house developed software. The data imputation techniques developed in FY 2002/2003 were the key to the present processed data service. Recently we also added another level of automation that Mn/DOT analysts can now freely make on-line changes on the number of stations and the content of the detector equivalency table of the stations. Such improvements provided great flexibility that Mn/DOT analysts need to generate data for their stakeholders.

Beyond the basic data archive services, TDRL also developed a method of detecting faulty loop detectors using software based on the archived traffic data. A new algorithm that classifies detectors into four classes, i.e., healthy, marginal, suspected, and highly suspected detectors, was successfully developed in 2003 and implemented as a useable software by operators. This software quickly identifies faulty and suspicious detectors out of about 5,000 loop detectors managed by the Regional Transportation Management Center (RTMC) and relieves the time of maintenance crew. This software is presently tested and evaluated by RTMC in St. Paul, Minnesota. Another important task completed by TDRL in FY 2003/2004 was Mn/DOT's request on developing a raw-signal probing tool for the state's Weigh-In-Motion (WIM) systems. TDRL successfully developed a new probing system and delivered it to Mn/DOT. This system allows analysts to examine various stages of raw signal processing and computational procedures from which it helps determine where the source of the problem is.

Table of Contents

List of Tables

List of Figures

CHAPTER 1: INTRODUCTION

The key feature of today's Intelligent Transportation Systems (ITS) has been the use of variety of traffic and road weather sensors to improve the overall system performance. ITS generated data has been successfully used in managing system operations and/or providing information on system conditions (ASUS, 2000; Magiotta, 1998). Recently, increasing deployments of ITS brought an awareness that ITS-generated data offers a great promise beyond present operational and monitoring uses, e.g., they can be used for planning and research (ASUS, 2000). Unfortunately, much of today's ITS generated data has not been archived, i.e. the important historical record has been continuously lost. Recognizing these problems, the U.S. DOT created a subcommittee called the Archived Data User Service (ADUS) as an official program for studying transportation data archiving and serves as a part of the National ITS Architecture (NCHRP Report 446, ADUS, 2000). ADUS defines archiving problem as "urgent". The Transportation Data Research Laboratory (TDRL) at NATSRL was established to meet this national need and to archive the Minnesota state's ITS generated transportation data.

There are several challenges in archiving large scaled ITS generated data[1]. First, ever increasing deployment of ITS by state DOTs increased the data size to an unprecedented amount, creating a technical barrier that has worked against archiving the large scaled statewide data. Second, data must continuously flow in (collected) to a central location from the whole state without interruption of communication or device failures for archiving. Within the statewide network, there are many failure points that are hard to manage, so reliable collection of statewide data is a continuous challenge. Third, no standard archive formats or tools are available for archiving statewide data. As a result, archiving has been only accomplished in partial with non-uniform formats, which often defeats the purpose of archiving for sharing. In general, archiving and managing statewide data is considered expensive, complex, and challenging. Many reports concur with this opinion (Edwards, 1995; Fairhead, 1995; Fogarty, 1994). Regarding the maintenance of data center, Kodor states that there is no such a thing as a complete data warehouse, either in terms of the environment or the tools (Kador, 1995). In other words, data center or warehousing should be an evolving concept.

The TDRL's objectives and views are different from the general data warehousing view. We believe that the expensiveness and complexity of building data warehouse is largely contributed by relying on the structure of the traditional relational database management systems (RDBMS), allowing all different types of formats which may provide operational simplicity but at the cost of complex data structure and large data size. TDRL believes that such approaches do not provide an efficient, robust system that can archive large-scaled data for long-term (such as next 100 years). RDBMS based traditional approaches have been pursued by the researchers at the University of California Berkeley through a PATH research program (Chen, 2001) and by the University of Virginia (Smith, 2003) using only traffic data. These trials show the above stated drawbacks, i.e., expensive cost ($multi-million) and complexity. At TDRL, we set a goal of archiving the Mn/DOT's ITS generated for the next 100 years (or more) by

[1] Large scale refers to the data size that is equivalent to one or more of statewide ITS data.

strictly following the standard data formats. This rigid approach is proved more economical and efficient for large scaled data.

As TDRL has started working with ITS generated data and accumulated experiences on the characteristics of data, it learned that any type data generated from transportation sensors can be expressed into a uniform form. TDRL studied Common Data Format (CDF) developed by the National Space Science Data Center (NSSDC) (Goucher, 1994)] and Hierarchical Data Format (HDF) developed by the National Center for Supercomputer Applications (NCSA) as well as databases at the initial state to find the solution (Kwon, 2003). During FY 2003/2004, TDRL finally developed a new framework referred to as the Unified Transportation Sensor Data Format (UTSDF) for archiving large-scale transportation data and decided to archive all future data using UTSDF (Kwon, 2004). UTSDF creates an archive that is compact and easy to retrieve and provides a uniform standard archive format. As a result, the users of TDRL archives only need to learn a single format to be able to use the entire data. The properties of UTSDF can be summarizes as follows:

- Unified data format for all types of transportation sensors
- Adaptable for changes in spatial configuration of sensors
- Easy to create and use the archive
- Simple to learn and manage
- Fast retrieval of large amount of data
- Compatible with all types of computers and operating systems
- Compact size for efficient storage and distribution
- Inclusion of meta data (description of data)
- Low cost to build and manage large scaled data

The efficacy of UTSDF can be demonstrated using the archive size and its retrieval performance (Kwon, 2004). When a single day amount of RTMC traffic data was stored into a MS SQL (RDBMS), the size became 370MB. When the same data was archived using UTSDF, its size shrank to 12MB. This means that UTSDF achieves storage efficiency of 3,000% over RDBMS. For the statewide RWIS data, the size of raw data for a single day is about 4.1MB. When the same data was archived using UTSDF, its size usually shrinks to about 415KB, which is 10:1 storage efficiency (Kwon, 2004). Such a huge saving in storage is extremely important when the size of the raw data is very large. We also tested retrieval efficiency of the archive against RDBMS. Our benchmark test shows that UTSDF retrieval time of large amount of data (larger than single day data, i.e. larger than 370MB) is about 80 times faster than that of RDBMS. With the initial study results, TDRL concluded that UTSDF is a logical choice of archiving large scaled data, and it will be the TDRL's future standard archiving technology.

With the completion of the development of UTSDF, the methodology for archiving Minnesota's statewide ITS generated data is now well established. The remaining work is to actually build the archives. A list of immediate works includes UTSDF conversion of the present binary traffic data, documentation of UTSDF and data, and continuous development of UTSDF analysis tools. Also, TDRL needs to implement a server side software at Mn/DOT to avoid missing data from the present Internet communication links. Another important work needed is the conversion of the present

location specification to differential GPS coordinates so that uniform location information can be applicable to locate all types of the transportation sensors. These works will be completed in FY 2005

As TDRL expands more working relations with Mn/DOT TDA and RWIS office on establishing statewide ITS data archives, it was learned that the fundamental problem also lies in the raw data acquisition systems especially in the rural roads. In most states, 80 percent of lane miles are rural roads, and 50 percent of them are not paved. Even though large percent of roads are unpaved dirt/gravel roads, according to Mn/DOT staffs and engineers, there are no reliable technologies for collecting traffic data from unpaved rural roads to date. Pneumatic tube technologies developed for hard flat surfaces cannot be used in rural gravel roads because of frequent punctures. Radar or ultra sonic technologies cannot be used because of their high power consumption. Inductive loops cannot be used because of their high failure rates by ground shifts and wire punctures and breakages (by gravel). In early 2004, Mn/DOT's Office of TDA who manages statewide transportation data recommended TDRL for research engagement in development of data acquisition systems for rural dirt/gravel roads. One difficult challenge is that the cost of device must be low and be operational for many days with only battery power. For FY 2004/2005, TDRL plans to assist Mn/DOT in finding the proper technology and develop an appropriate data acquisition technology as a collaborative effort with Mn/DOT TDA. Developing ITS technologies for rural roads also matches with the original spirit of NATSRAL's research focus on regional transportation issues.

Another important outcome of FY 2003/2004 at TDRL is the development of a Weigh-In-Motion (WIM) probe. One basic problem of today's WIM systems is that they do not allow users to see the raw WIM signal. As a result, when vehicle classification results from the WIM system are questionable, there is no easy way of knowing the problem source and the amount of adjustments needed. One of the important steps of data quality control is knowing where the source of the problem is and knowing how much adjustments are needed in the data to compensate the error. Without knowing the problem source, quality control of WIM data has been a continuous challenge. In FY 2003/2004, Mn/DOT's Office of TDA finally requested TDRL to develop a raw WIM signal probing system that allows see the signal state from the pavement embedded WIM sensors. Upon this request, TDRL worked on the system with the sensors and charge amplifiers provided by Mn/DOT, and completed the system development in Jan 2004. It was designed as a portable interface box that can be carried around the WIM installation sites and test the questionable WIM signals. The Mn/DOT TDA office supplied the notebook computer and charge amplifier required, and the TDRL developed the data probing software and interface electronics including the data acquisition boards. The WIM signal probe developed by TDRL essentially helps diagnose the system problems and provides a means to quality measurement for WIM data.

The rest of report will provide further detailed information on four areas: UTSDF, processed data service, loop detector fault diagnostic software, and WIM probe. Each will constitute a chapter in this report.

CHAPTER 2: UNIFIED TRANSPORTATION DATA FORMAT (UTSDF)

2.1 Introduction

Today's transportation systems have been increasingly utilizing a wide range of sensors to monitor, control, and analyze many parts of transportation system components. The sensor usage has been further accelerated by the US DOT's emphasis on Intelligent Transportation Systems (ITS) technologies in recent years. On the other hand, while the usage of sensors has been increased, archiving (or saving) of the sensor data has not been increased at the same rate (ADUS, 2000). In fact, only a small fraction of sensor data from the today's transportation systems has been archived. For example, each intersection typically includes a number of vehicle detecting sensors to optimize the timing of the traffic controller, but the data is rarely archived.

There are a number of reasons that archiving of transportation sensor data (TSD) has not been eagerly pursued by transportation departments. First and the most influential factor is the cost, i.e., while the cost of sensors is low, the cost of archiving their data is expensive. As a result, archiving has been frequently unwelcome by maintenance engineers and managers. Second, continuous flow of data from transportation sensors adds an additional burden to the management of data and archiving. Sensors continuously generate data once they are activated, and the amount of data can be quickly accumulated to a large amount. Moreover, all parts of the data acquisition system must continuously work without faults. In effect, archiving often reveals the weakness or reliability of the system, which sometimes is not a pleasant thing. For maintenance personnel, archiving is an additional work but it can also lead to a substantial pressure because missing or lost data can be a responsibility. Third, when data is collected from many different types of sensors, which would be the case in Road Weather Information Systems (RWIS), management of data is complicated. To date there exists no uniform and efficient data format that can be used for archiving all types of sensor data. As a result, management of data from various sensors and data acquisition systems developed by many different types of manufacturers is in itself a challenge. For example, it requires a large amount of work to keep up with the data format differences and modifications, incompatible file formats, version changes of software tools and operating systems, etc. Therefore, acquisition, archiving, and maintenance of data from a large number of sensors used in today's transportation systems are often more difficult than what individual sensor shows.

The next question is then "Why do we need archiving?" or "Do we really need archiving?" The answer would depend upon the needs. However, if we assume that system analysis (performance, reliability, etc.) is needed at some point in the future, archiving of data would be a necessary part since analyzing a system without data is difficult and unreliable. Therefore, the more important question on archiving is not in the need of or not, but in what extent, i.e., which selected locations, what sensors, and how long the data should be archived. The Unified Transportation Sensor Data Format (UTSDF) introduced in this documentation is an attempt towards making TSD collection and archiving simple and easy regardless of the number of sensors, sensor types, and variability such as location change or removal.

Our main objective of developing UTSDF was to simplify the archiving task of TSD. An important step in this process is to create a uniform and efficient data format that is simple and independent of operating systems and programming languages. The users of transportation sensors should only need to learn a single type of data format for archiving and for the use of the

archived data. Based on the UTSDF we also intend to build reliable data acquisition models and efficient methodologies for archiving large-scaled TSD such as a statewide system.

At our Transportation Data Research Laboratory (TDRL), the need for the development of UTSDF was born out of the needs in developing statewide archives of TSD that have characteristics of large scaled data and variety of data types including non-numeric data. In developing UTSDF, the following list was set as the objectives.

- A single unified data format for all types of transportation sensors
- Simple to understand and use
- Easy to manage
- Compatible with all types of computers, OS, and programming languages
- Easy to distribute or share large amount of data
- Compact, compressed form
- No or low cost in adopting the technology
- Fast and easy retrieval of a large amount of data from the archived data
- Adaptable for changes in sensor locations or configuration
- Inclusion of description of data (meta data)

This documentation describes the format of UTSDF and archiving methodologies that could be applied for statewide TSD archives.

2.2 Assumption on Transportation Sensor Data (TSD)

We refer all types of sensors (electrical, magnetic, mechanical, optical, chemical, etc) that are used in transportation systems as transportation sensors. Transportation sensors are typically used in monitoring the state or conditions of a transportation system component and often placed under the pavement or near the roadways. The digitized values or decision results of the sensor state comprise the sensor data. We assumed that all sensor readings are obtained from a fixed sampling rate, which would be the case for the most of the real-world TSD. For example, if traffic counting was done at every 30-second interval, we expect 2,880 data points per day. We further assume that the sampling rate is determined based on the sampling theorem, i.e. twice the bandwidth (also called a Nyquist rate) of the original signal (Alan, 1893). If sensor readings are sampled at a Nyquist rate, the sampling theorem guarantees that the complete original signal can be reconstructed from the sampled data (Alan, 1893). Consequently, it is assumed that re-sampling is possible from the reconstructed signal without loss of information.

Some sensors do not produce numerical values but descriptive conditions. For example, pavement sensors produce pavement conditions such as wet, dry, ice, etc. As long as those readings are recorded at a fixed rate, the data can be stored in UTSDF. A single sensor may produce multiple types of values. For example, a single inductive loop detector produces two types of values, volume and occupancy. In order to differentiate between the sensor and values produced, we refer each type of sensor values as parameter, i.e., volume and occupancy are parameters of inductive loop detectors. These parameters are the final data (or variables) that are stored as sensor values in UTSDF.

2.3 Basic UTSDF Archive File

A single UTSDF archive file (or simply called UTSDF file) is a zip-compressed archive file of many small data files called *daylets* (described in the Section 2.4). More specifically, a single UTSDF file is created based on the time unit of a single day, in which it is a collection of *deaylets* from the same day and adheres the following name convention:

yyyymmdd.Class_Name

where the date of the archived data is encoded as the file name with eight digits, i.e., yyyy is the year, mm is the month, and dd is the day. The Class_Name is the name of the sensor class such as RWIS, traffic, or WIM (Weigh-in-Motion). For example, an RWIS archive file on Feb 23, 2003 would have the name *20030223.rwis*. Similarly, the traffic file on the same day would have the name *20030223.traffic*. As a result, when the archived files are viewed as a sorted list, it should be in a chronological order. In general, different classes of the archived files are stored in separate directories, and one year of complete RWIS or traffic archive would consist of 365 UTSDF files. The structure of a single UTSDF file is illustrated as a block diagram in Figure 1. The size of a daylet would depend on the type of parameters it stores and described in Section 2.4.

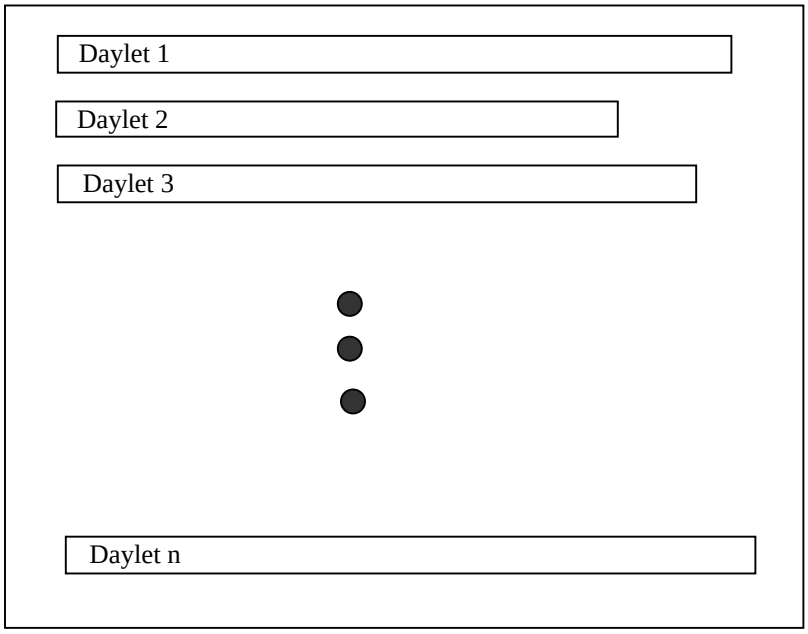


Figure 1: UTSDF archive consisting of n daylets. The number of daylets in an archive file would depend on the different types of parameters and the number of sensor locations.

2.4 Daylets

Daylets are the basic components of UTSDF file and contain the actual sensor data. The name of each daylet is assigned based on the spatial information (i.e., location) and the parameter type of the sensor, while the name of UTSDF file is assigned using the temporal information (date) and the sensor class name. The reason behind the choice of this name convention is to utilize the temporal and spatial properties of TSD, which are discussed in (Kwon, 2004). The basic name format consists of several fields separated by dots and is shown below:

SysID.SiteID.SensorID.ParaName

SysID: System ID. It is a unique number assigned for system characteristics such as a manufacturer type.

SiteID: Site ID. It is a unique numeric number assigned to each site based on the geographical location of sensor.

SensorID: Sensor ID. It is a unique number assigned for multiple sensors of identical types within a site (same location). For example, if three pavement temperature sensors are installed at a site, the sensors are assigned with SensorID, 0, 1, and 2.

ParaName: Parameter Name. It is a shortened parameter name without any space. For example, air temperature is shortened as "atemp" in this field. Please refer to Appendix A and B for RWIS and traffic data.

UTSDF itself does not define each field. The four-field name convention is provided as a recommendation. It is the archive provider's responsibility that each field is defined, documented, and provided along with the archive. An example documentation of theses definitions are provided in Appendix A and B based on the actual UTSDF archives of the Mn/DOT (Minnesota Department of Transportation) RWIS and traffic data used in TDRL.

For illustration, the name convention of daylets for statewide RWIS used in Mn/DOT data is used. Say, we wish to create a daylet for air temperature at System ID=330, Site ID=17, and Sensor ID=0, then the daylet's name would be assigned as "330.17.0.atemp".

If the statewide system consists of only one type of systems and no duplicated sensors exist in a single site, the first three fields can be combined as a single field of site ID number, but this will limit the flexibility and future extensibility. Again, a proper documentation must be provided for the definition of the daylet's name. At a minimum two fields must exist, i.e. the Site ID and the Parameter Name to be qualified as UTSDF.

The content of daylet files is simply a long string of ASCII characters that represent a single day data for the parameter it stores. Use of ASCII string provides excellent portability and allows storage of both numeric and non-numeric data. Each datum within a daylet must have the same length (the same number of characters per datum), so that the string length of a daylet is always computed by the datum length multiplied by the number of data items in the daylet. If a null datum exists, repetition of "N" characters for the allocated datum length is entered. Repetition of the same characters for null data is later efficiently compressed by the compression process. Since each datum has the same length, sampling period is precisely determined by dividing 24 hours by the number of data entries in the daylet, or vice versa. For example, if wind direction data is sampled at every 10 minutes and three digits are allocated for each datum to represent an angle in clockwise degrees from north, then the total string length of the wind-direction daylet would be 432 and it contains 144 data entries. Time stamp is not entered for each datum since we assume that all data is sampled at a fixed sampling rate within that day. We assume that data can be always reconstructed from the sampled data and resampled to produce data for any time of the day based on the sampling theorem (Alan 1983). For the negative

numbers, a single character " – " is used as a prefix, but positive numbers do not use any prefix character.

Example) Suppose that air temperature was sampled from a sensor for a single day with a 10 minute sampling period. The data collected from the sensor are degrees in Celsius and shown below:

```
00:00  27.5
00:10  10.5
00:20  5.8
00:30  N/A        ; missing data
00:40  0.5
   .
   .
   .
23:40  -13.5
23:50  -10.5
23:50  -5.5
```

Suppose that four digits are allocated for each datum representing a unit of one-tenth degrees in Celsius. Then, the string for the above data is packed as an ASCII string by simply concatenating four digit numbers in chronological order, i.e.,

027501050058NNNN0005…-135-100-055

When this string is saved as a file with the predefined daylet name, it becomes the daylet for the air temperature for the given date of the UTSDF file.

In the daylet ASCII string, it is important to notice that no line breaks, commas, or spaces are used to separate the data. Such data separators are not needed, since we are using the same number of ASCII characters for each datum within a daylet. One may concern about the increased size of the data due to the use of ASCII string and fixed length. However, since the daylets are later zip-compressed to an archive file, the size of the initial data should not be of concern. According to our study, zip compression algorithm efficiently compresses the ASCII strings with fixed data length. The size was often smaller than the zip-compressed result of the equivalent size of binary data (Kwon, 2004).

One advantage of using daylets in archiving is that since each daylets are independent each other in terms of storage, they can be easily added or removed without any modification in overall data structure. For example, as more sensors are installed at new locations or removed from old location, daylets can be simply added or removed. This parallelism with hardware makes the overall management of the archive simple.

2.5 Log and missing information

Each UTSDF archived file includes two special files. They are yyyymmdd.missing and yyyymmdd.log files where yyyymmdd is the numerical values of the year, month, and day of the archived date. The yyyymmdd.missing file includes the list of names of missing daylets (null data for the entire day) on that day. The daylet list is separated by a comma.

The yyyymmdd.log file serves as meta data and includes information about the data in the archive file. The format of this file is not defined except that ASCII characters should be used. The archive provider should supply the documentation on the *.log file where the detailed format should be defined. Any information the user of the archive must know, should be stored in this file such as the number of active sites, the sites in out of service, special events, etc.

2.6 Data Compression

A UTSDF archive file is simply a zip-compressed file of daylets. When a single UTSDF file is uncompressed (unzipped), it should reproduce all of the original daylets that were compressed into a single archive file. Since most unzip tools allow unzipping of a single or just few files, retrieval of only needed daylets can be easily done from a UTSDF file.

Zip compression uses a compression algorithm referred to as the Deflate. Deflate combines the LZ77 algorithm (Ziv, 1977) for marking common sub-strings and Huffman coding (Huffman, 1952) to take an advantage of the different frequencies of occurrence of byte sequences in the file. Deflate does have an important advantage in that it is **not patented** (no need to obtain licenses). Thus, it is presently the most widely used file compression method. It is used in the WinZip™ freeware in Windows™ and the gzip program in Unix, and the jar files in Java. The Deflate algorithm is also a standard for the Internet IP payload compression (RFC 2394). Today, the term, zip or unzip, is commonly used replacing the algorithm name Deflate. For programmers, free source codes are available from Internet for zip and unzip. Also, many convenient commercial software tools, such as dynaZip, Sax.net, Xceed, ComponentOne, etc., are available for embedding unzip or zip function into application programs. At TDRL, a freeware WinZip™ and DynaZip™ utilities have been used as the basic tool for compression and decompression.

2.7 Organization of Archive Directories

The recommended organization of UTSDF archives is a hierarchical organization based on a file directory structure. File directory structure (or system) has been successfully used in storing all types of data since the beginning of the computer age and has proven very effective in handling large complicated data. There are a number of benefits in using a file system as the structure of archive organization. First, file system is such a familiar form to any computer users that it is probably one of the easiest structures to understand and manage. Second, it is one of the most stable and reliable parts of any computer operating system. Third, temporal, spatial, and computational hierarchies of the TSD properties nicely fit into the hierarchical nature of the file directory structure (Kwon, 2004; Kwon 2004).

The organization of archives should be based on clarity and efficiency in retrieval of the data. We will consider organization of two types of common transportation sensor data, i.e., RWIS and traffic data. First, consider that we wish to build a statewide archive for traffic data. Since the number of traffic detectors used in a state is such a huge number, it is convenient to divide the data into districts to form a reasonable size of the archive files. Within each district, the sensors can then be given unique ID numbers or can be organized using dot separated fields as shown in Section 2.4. Each district directory is then further divided into year directories where daily UTSDF archives are stored. This directory structure is simply and utilizes the division of data that we are familiar with, which has benefits of clarity. This directory structure is illustrated in Figure 2. In this case, if the district, year, date, and the detector ID number are known, the data can be quickly searched. Notice that the spatial and temporal hierarchies of traffic data properties are alternatively utilize in the directory tree.

Next, we consider a statewide RWIS archive. Since the number of RWIS stations in a state is typically less than 1,000 and the stations are centrally managed, dividing them into districts can result in small fragmented archived files. Too many fragmented archived files are less efficient in sharing data. Therefore, it is more logical to organize the archive directories into simply year directories as shown in Figure 2. In this case, each UTSDF archive file would contain RWIS daylets for the entire state for a single day. TDRL presently uses this organization to archive the Mn/DOT's statewide RWIS data. However, if the number of stations in a state were very large such as exceeding 5,000s, then dividing the directories into Districts would be more sensible. Again, the overall structure utilizes temporal and spatial relations since daylet's names are organized based on spatial relations.

One important part of the UTSDF directory structure is the inclusion of /docs directory at the next to the root level as shown in Figure 2. In the /docs directory, the archive provider should include all documentations necessary to understand the archive. It helps the users of the archive as well as the maintenance. The documentation could include daylet field name definitions, string length allocated for each parameter, basic units, sensor locations, sensor manufacturer information, maintenance history, addition or removal of sensors, etc. Inclusion of the /docs directory follows the sprit of the inclusion of a log file inside the daily UTSDF archive file, i.e., description of the data is provided at multiple levels, directory level and daylet level.

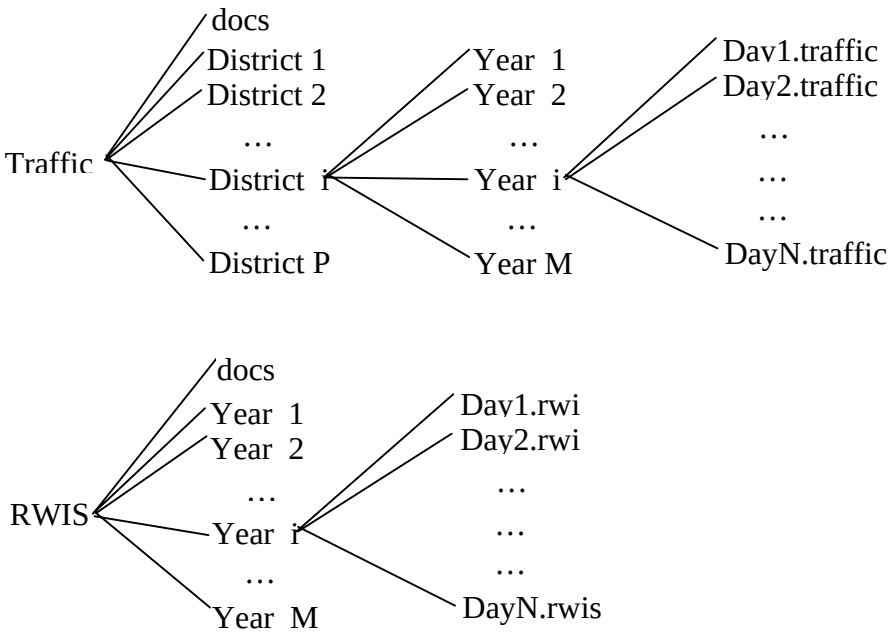


Figure 2: Directory structure of statewide UTSDF archives: RWIS and traffic data example

2.8 More Complex Structure of UTSDF: *Monthlets* and *Yearlets*

Until now, we only discussed archiving of raw sensor data in which daily operation of archiving is assumed, where daylets are utilized. However, we frequently need processed data such as Average Annual Daily Traffic (AADT) or daily average/low/high of pavement temperatures. For those processed data, expressing the data in a larger time scale is necessary such as a year rather than a single day. This type of needs can be met using *monthlets* and *yearlets*, which are similar to daylets except that they contain a whole month of data or a whole year of data.

Unlike daylets, *monthlets* and *yearlets* would require multiple parameters in a single file. For example, a *yearlet* storing daily average/low/high air temperatures for the entire year requires three parameters. In such a case, the yearlet should contain three strings one for average, one for low, and one for high air temperatures. Each string should follow the same principle used in daylets, i.e., fixed length for each datum. Each string should be separated by a pair of carriage return and line feed ASCII characters for distinction. Figure 3 illustrates a *yearlet* with three strings.

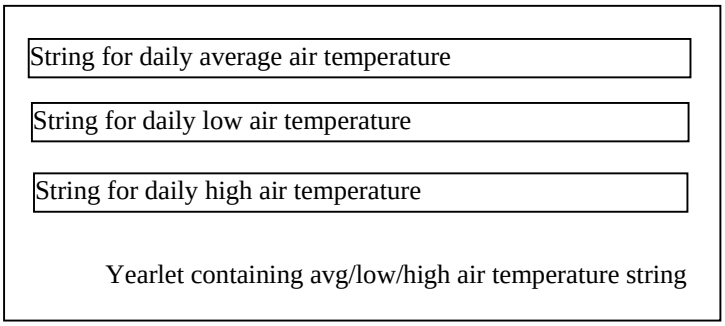


Figure 3: Yearlet example of daily avg/low/high temperature for the entire year.

In reference (Kwon, 2004), we described a computational hierarchy in which processed data can be organized and archived as a hierarchical directory structure. In a very large system such as a statewide network of sensors, archiving processed data for sharing is considered highly beneficial. Examples include AADT or daily average of RWIS parameters. For developing a directory structure for processed data, we recommend to follow the structure of the examples given in Figure 2. More specifically, additional root directory can be created for the computational hierarchies for each class of data. In the above example, we could create one directory for processed RWIS data and another directory for processed traffic data. The children directories of the computational hierarchy would depend on the type of computation and outcome, so it would require a development of subdirectories for the specific needs.

2.9 Binary UTSDF

In a standard UTSDF, all of the sensor data in *daylets*, *monthlets*, and *yearlets* are stored using ASCII characters. However, if the data consists of only numerical values and fixed sizes, binaries could be used instead of ASCII characters. When the data in daylets are stored in binary form, we refer the archive as *binary UTSDF*. In general, we do not recommend binary UTSDF since they are less portable between different operating systems and programming languages. Binary data can create a compatibility problem of byte orders known as Little-Endian and Big-Endian as well as the size definition in integers and floating points. Since the benefits of using binary for a smaller file size is diminished after zip-compression as demonstrated in (Kwon 2004), binary UTSDF is not recommended for archiving TSD.

2.10 Conclusion

We presented UTSDF in this introductory documentation. UTSDF was developed for archiving a very large set of transportation sensor data that include many different types of sensors. It is a simple and easy to use, and can be used in developing well-organized large archives. It is our hope that UTSDF is adopted in other states so that transportation sensor data can be easily archived and shared. At TDRL, we are continuously working on developing data visualization and analysis tools for UTSDF data we are providing. These software tools are presently distributed through http://www.d.umn.edu/~tkwon/TDRLSoftware/Download.html or links from http://tdrl1.d.umn.edu/services.htm.

CHAPTER 3: DETECTOR FAULT IDENTIFICATION USING FREEWAY LOOP DATA

3.1 Introduction

Although many types of vehicle detection technologies have been developed over the past few decades, inductive loop detectors are still the predominant type today (FHWA, 1990; Chen 2003). Most metropolitan cities use a large number of loop detectors in their freeway network to monitor and control traffic. Traffic Management Centers (TMC) are typically responsible for this function and collect real-time and historic loop data. In Minnesota, 30-second data is collected from about 5,000 loop detectors installed on the Twin Cities' freeway network, which is used for real-time ramp control and monitoring. These data are daily archived for various purposes such as congestion analysis, traffic data reporting, regional transportation planning, etc. In California, data is collected from nearly 15,000 loops (Chen, 2001; Chen 2003). It is therefore important that the detectors and data acquisition systems function correctly and produce reliable data, but it is also a challenge to maintenance staff to manage so many loops. Unfortunately, recent budgetary problems did not help by further reducing the maintenance staff. One way of alleviating the shortage of maintenance staff and dealing with so many loop detectors is developing a software tool that can diagnose and screen the problematic loops from the daily archived data. This paper presents an algorithm developed with the collaboration of the Minnesota TMC that identifies and classifies faulty conditions of loop detectors for maintenance.

Over the years, many loop detector maintenance manuals and algorithms have been developed and documented. Early works focus on developing installation acceptance tests and maintenance criteria based on technical data obtained under controlled tests (James, 1976). One of the most comprehensive documentation on loop detectors was published by Federal Highway Administration (FHWA, 1990), which encompasses from the basic principle to design, installation, and maintenance practices. This documentation provides a good overview of faulty conditions on a single detector. A simple and yet very practical approach that exploits redundancy of dual loops in speed traps was proposed by Coifman (Coifman, 1991). This approach assumes that dual loops have similar on-time (occupancy), from which correlation index discovers large deviations caused by detector faults. This method works well for free-flow conditions, but, if traffic is congested or if both loops fail with similar conditions, it can lead to a detection error. Since majority of today's loop detectors are not implemented as a speed trap, there is a need for single loop based detection algorithm.

Several statistical acceptance tests to identify erroneous loop data have been developed (James, 1976; Cleghorn, 1991). Some approaches utilize Fourier Transforms to identify abnormality (Peeta, 2002). Other approaches utilize illogical occupancy/volume relation or entropy (or randomness) of occupancy samples (Daily, 1993; Chen 2003). Yet another method proposed by Wall and Daily uses pairs of single loops under similar traffic flow for detecting erroneous data (Wall, 2003). These approaches and other imputation techniques (Schmoyer, 2001; Smith, 2003) employ relatively simple techniques for detecting erroneous data, but more concentrate their

efforts on developing algorithms for correcting or adjusting the data before the data is used for applications. More specifically, the mentioned and similar approaches focus more on detecting **data** problems rather than identifying the loop-specific hardware or software problems that require maintenance. Many algorithms for loop diagnostics using archived data may have been developed in the past, but they are often either not well documented or proprietary and not available to public.

This paper focuses on developing a decision tree for identifying specific detector problems for loop maintenance. Our objective is to develop a software tool that implements a decision tree to suggest specific types of loop problems that indicates what types of maintenance checks and repairs are needed. For simplicity and convenience, we classify each detector to one of the four classes based on the level of problems: highly suspicious, suspicious, marginal, and healthy detectors. This classification provides an organized summary of the detector status and suggests priority of the service needs. In addition, the final output of the decision tree provides the specific types of problems identified by the algorithm. This type of software (or algorithm) would help reduce manual inspection time for maintaining a large number loops in TMC.

3.2 Classification

Detector health status is classified into four categories according to the severity of abnormalities observed from the loop analysis. The meaning of each class is described below.

1. **Highly Suspicious**: The detectors in this category show a sustained period of missing data implying a severe faulty condition. This condition could be caused by temporary power failure at the detector/controller cabinet, communication failures, complete loop wire breakages, or from not activated/used detectors.
2. **Suspicious**: The detectors in this category do not include a sustained period of missing data. However, the data pattern shows one or more abnormalities such as the pulse mode. A criterion set by the predetermined parameters and a decision tree identifies the abnormalities. This condition could be caused by sensing of adjacent lanes, missing counts, transient connection problems in loop wire, or water damages. The important maintenance operation for this category of detectors is to check the sensitivity settings of the detector card, coupling between two closely spaced lead-in wires, grounding of wires, pavement crack and sealant, and manual verification of vehicle counts. If the detector counts correctly, the abnormality pattern was likely caused by transient signal failures, incidents, or unusual patterns of special events.
3. **Marginal**: The detectors in this category show a pattern close to a normal healthy detector but the data pattern does not indicate that the detectors are completely healthy. They are in a marginal state between healthy and suspicious. Transient faulty conditions and special events can produce the data type in this category. Therefore, for the benefit of doubt, it is recommended that the detector cabinet, lead-in wires, and loops be checked. However, a low priority in the maintenance order should be given to this category of detectors.

4. **Healthy**: When a detector passes all of the test criteria, the detector is claimed healthy. Except for annual or biannual preventive maintenance, the detectors in this category may not require maintenance operation.

This classification simplifies the overall view of the detector status and helps repair planning. For understanding the detailed types of loop problems, please refer to Table 1. It summarizes the types of loop problems that appear in the Mn/DOT loop repair record.

Table 1: Terminology for Loop Repair Records in Mn/DOT TMC

Type	Reason in the report	Description
1.	No hits	Detector is not counting vehicles, sometimes happens for a short period of time but often is permanent until the detector is fixed.
2.	PM	Preventive maintenance. No flaw in the data is found but a field problem that needed correction is noticed.
3.	Occ spikes	Highly fluctuating values of occupancy.
4.	Lock on	The detector remains ON and fails to record two or more vehicles as separate vehicles and counts them as one. This happens when a vehicle is tailgating another during congestion. Detection: 100% occupancy for several minutes (5 – 10 min).
5.	Chattering	Detector is reading extremely high volumes and stay high, or fluctuating wildly between 30 sec samples, mostly due to sensitivity setting errors.
6.	Low counts	Detector is counting fewer vehicles than actual count.
7.	High counts	Detector is counting more vehicles than actual count.
8.	Road damaged	The loop is exposed in roadway or the underground conduit and lead in cable has been damaged due to some construction work.
9.	Flow spikes	Spikes in the flow rate at which vehicles pass the detector.
10.	Splice	Splice defect, problem in the way the loop is joined.
11.	Bad counts	The detector is not counting vehicles properly.
12.	Lead in cable bad	Lead in cable is defective and may need replacement.
13.	Pulse mode	The detector is in pulse mode and needs to be changed to presence mode. In pulse mode, occ = (vol * d)/duration where d is the width of the occupancy pulse.
14.	Swapped	Detector is swapped with another loop. Needs to be joined to detector specified in the previous work report.

15.	Needs replacement	Loop needs to be replaced due to some defect.
16.	Wired new loop	New loop was wired as specified in previous work report.
17	Separate from another loop	Separate a loop from another one and give it a new location.

3.3 Measurement Parameters

Various data measures are used in the algorithm to differentiate different types of faulty conditions. The algorithm is then implemented as a decision tree that takes into account the measured criteria to classify the detector problems. The measurements are obtained from a single day observation of 30-second single loop data and summarized below.

1. *ConsqZeroCnt* represents the number of consecutive 30 sec slots with zero or invalid volume.
2. *LockOnCnt* is the number of consecutive 30 sec slots with 100% occupancy.
3. *CorrlationCoef* (Correlation Coefficient) indicates the degree of linearity between the volume and occupancy. It is computed using the following equation.

$$CorrelationCoef = \frac{\left(n\sum_{i=0}^{2879} Vol(i)\times Occ(i) - \sum_{i=0}^{2879} Vol(i)\sum_{i=0}^{2879} Occ(i) \right)}{\sqrt{n\sum_{i=0}^{2879}\left(Vol(i)\right)^2 - \left(\sum_{i=0}^{2879} Vol(i)\right)^2}\sqrt{n\sum_{i=0}^{2879}\left(Occ(i)\right)^2 - \left(\sum_{i=0}^{2879} Occ(i)\right)^2}} \quad \ldots(1)$$

where *n* is the total slots with valid volume and occupancy data and i is the index for the 30sec time slots.

4. The changes or fluctuations in occupancy between 30 sec slots are measured using *OccSpike*. It is calculated using the following equation.

$$OccSpike = \sum_i 1\left(\sqrt{\frac{\{(Occ(i-1)-Occ(i)\}^2 + \{(Occ(i)-Occ(i+1)\}^2}{2}} \geq \theta \right) \quad \ldots(2)$$

where function $1(x \geq \theta)$ is a threshold function that produces 1 when *x* is greater than equal to θ.

5. The changes or fluctuations in volume between 30 sec slots are measured using *VolSpike*. It is calculated using the following equation.

$$VolSpike = \sum_i 1\left(\sqrt{\frac{\{(Vol(i-1)-Vol(i)\}^2 + \{(Vol(i)-Vol(i+1)\}^2}{2}} \geq \theta \right) \quad \ldots\ldots\ldots(3)$$

6. A volume dataset, *Vol (Occ)* (where *Occ* = 0 to 100) for each occupancy is prepared. In all there will be 101 such datasets. The first data set contains all the values of volume when the occupancy was zero. Similarly, *Vol* (*100*) contains all

the values of volume when the occupancy was between 99 and 100. First, the mean for each volume data set *Vol*(*Occ*) is computed as:

$$Mean(Occ) = \frac{\sum_{j=1}^{k} Vol_j(Occ)}{k} \quad \text{.....(4)}$$

where *k* is the number of data points in the occupancy computing. Next, *Spread(Occ)* for each occupancy is computed using the standard deviation of each volume data set and represents the spread amount of the occupancy.

$$Spread(Occ) = \frac{\sqrt{\sum_{j=1}^{k} (Vol_j(Occ) - Mean(Occ))^2}}{k} \quad \text{..... (5)}$$

where *k* is the number of data items in *Vol (Occ).*

The deviation index is a measure of deviation from the expected (standard) data collected from a healthy detector. It is calculated separately for low values of occupancy (0 to19) and the high values of occupancy (20 to 100). This is because more deviation in the low occupancy region is a stronger indication of problem in the detector health measure than the deviation in the high occupancy region.

$$Low\_DevIndex = \frac{\sum_{Occ=0}^{19} Spread(Occ)}{No\_of\_nonzero\_Vol(Occ)} \quad \text{..... (6)}$$

$$High\_DevIndex = \frac{\sum_{Occ=20}^{100} Spread(Occ)}{No\_of\_nonzero\_Vol(Occ)} \quad \text{..... (7)}$$

The final deviation index is calculated by combining Eqs. (6) and (7) by taking a weighted sum. More weight is given to the low occupancy deviation index and the total deviation index (*DevIndex*) is computed as follows.

$$DevIndex = 0.7 \times Low\_DevIndex + 0.3 \times High\_DevIndex \quad \text{.....(8)}$$

7. The next measure used in the decision tree is the average volume for occupancies from 85 to 100. At very high occupancies, the volume is expected to be low. If this is not the case, it indicates a possible loop or detector problem. *VolAvgOnHighOcc* (Volume Average of High Occupancy) is computed as follows.

$$VolAvgOnHighOcc = \frac{\sum_{Occ=85}^{100} Mean(Occ)}{16} \quad \text{.....(9)}$$

8. Another measure or criterion used in the decision tree is the over-count percentage (*OverCountPercent*). It is not practically viable for more than 20 vehicles to pass over a single loop detector in a period of 30 seconds. If a detector is counting more than 20 vehicles in 30 second slot, it would mean that it is likely over counting. Moreover, if it happens frequently, it is likely a good indicator that the detector is having an over-count problem so *OverCountPercent* is used as one the measurements. *OverCountPercent* is the percentage of total time slots that show over-count and computed as follows.

$$OverCountPercent = \frac{No\_of\_30\sec\_slots\_with\_volume > 20}{Total\_no\_of\_30\sec\_slots} \quad \text{..... (10)}$$

9. *5MinVolMax* is the maximum volume observed over 5 minutes during the entire day.

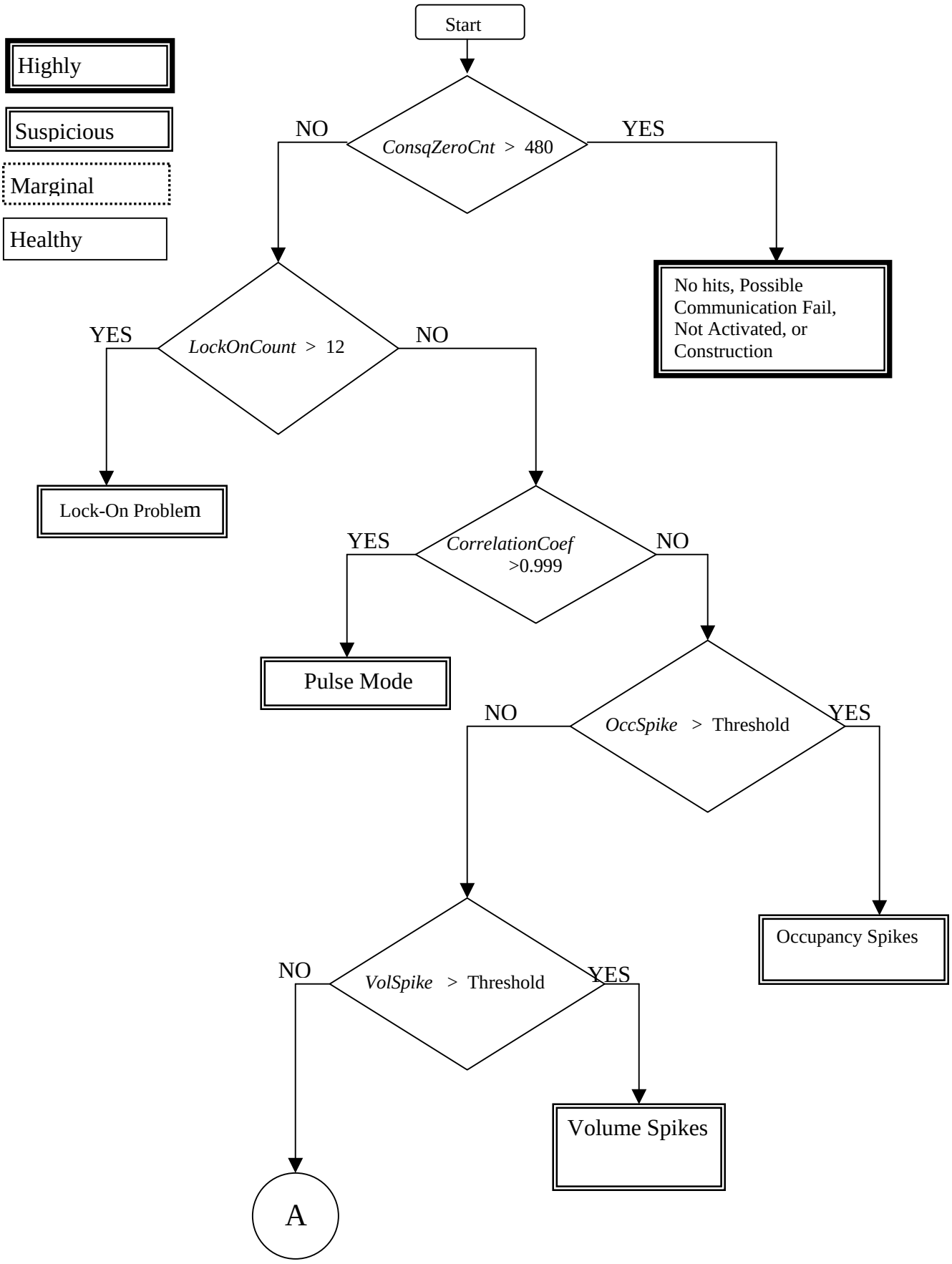
3.4 Algorithm Description

The overall algorithm implements the decision tree provided in Fig 4. Below describes the basic algorithm for easier reading.

1. Check if the detector data shows zero-volume for more than four consecutive hours after 6 A.M. If this is true, report No-Hits and classify the detector as highly suspicious.
2. Check if the detector is getting Locked-On for several minutes. If this is true then report a Locked-On problem and classify the detector as suspicious.
3. Check if the volume and occupancy have a too exact linear relationship, i.e., *CorrelationCoef* in Eq. (1) is close to 1. If this is true, report a Pulse-Mode problem and classify the detector as suspicious.
4. Check if the *OccSpike* in Eq. (2) is greater than a set threshold (default is 30). If true, report an Occupancy Spikes problem and classify the detector as suspicious.
5. Check if *VolSpike* in Eq. (3) is greater than a set threshold (default is 25). If true, report a Flow Spikes problem and classify the detector as suspicious.
6. Check if the average volume of high occupancies (*AvgVolOnHighOcc* in Eq. (9)) is greater than a set threshold (default is 60). If this is true, report Bad Count and classify the detector as highly suspicious. Else, go to 7.
7. Check if the 5min maximum volume is greater than 280. If true go to 8 else go to 10.
8. Check if the over count percentage (*OverCountPercent* in Eq. (10)) is greater than 30%. If it is true, report High Count and classify the detector as suspicious. Else, go to 9.
9. Check if deviation index (*DevIndex*) is greater than a set threshold (default 15). If true classify it as a suspicious detector with abnormal pattern else classify it as suspicious with transient problem.
10. Check if the deviation index (*DevIndex*) is greater than a set threshold (default 15). If true classify it as a suspicious detector with abnormal pattern. Else, go to 11.
11. Check if deviation index is greater than a set second threshold (default 12). If true, classify it as a marginal detector. Else, classify it as a healthy detector.

For software implementation, all of the threshold values used in the algorithm should be designed as a programmable parameter with default settings. Such a software tool was developed according to the algorithm described above and integrated as a part of the existing Detector Data Extractor (DDE) V 3.4 developed by the Transportation Data Research Laboratory (TDRL) at the University of Minnesota Duluth in collaboration with

Mn/DOT TMC. This integration has an advantage that data of each detector can be plotted and studied using the various visualization tools available within the DDE utilities while the erroneous detectors are checked. This software can be downloaded from http://www.d.umn.edu/~tkwon/TDRLSoftware/Download.html.


Highly
Suspicious
Marginal
Healthy
Start
ConsqZeroCnt > 480
NO
YES
No hits, Possible Communication Fail, Not Activated, or Construction
LockOnCount > 12
YES
NO
Lock-On Problem
CorrelationCoef >0.999
YES
NO
Pulse Mode
OccSpike > Threshold
NO
YES
Occupancy Spikes
VolSpike > Threshold
NO
YES
Volume Spikes
A

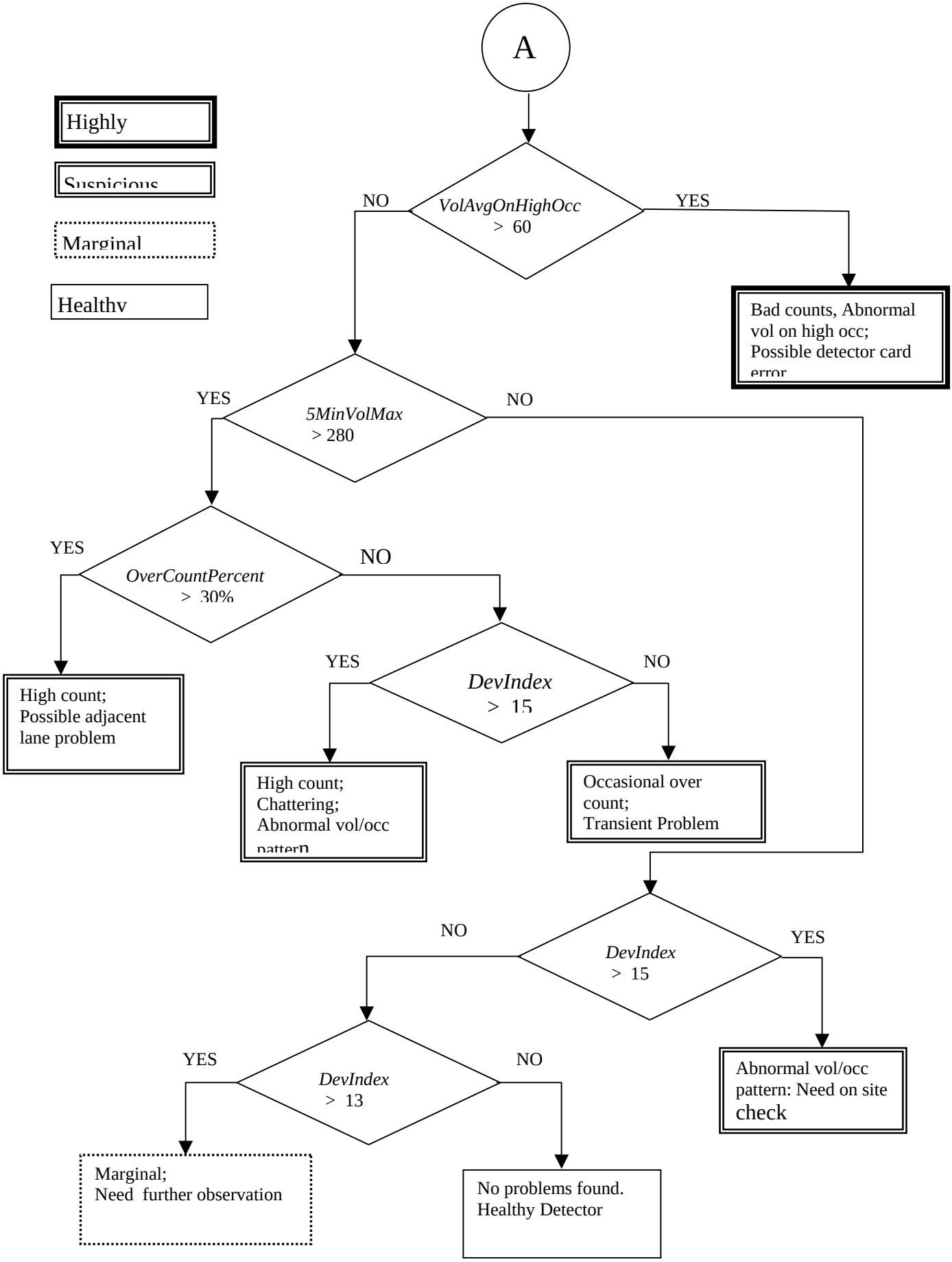

A
Highly
Suspicious
Marginal
Healthy
VolAvgOnHighOcc
> 60
NO
YES
Bad counts, Abnormal vol on high occ; Possible detector card error
5MinVolMax
> 280
YES
NO
OverCountPercent
> 30%
YES
NO
High count; Possible adjacent lane problem
DevIndex
> 15
YES
NO
High count; Chattering; Abnormal vol/occ pattern
Occasional over count; Transient Problem
DevIndex
> 15
NO
YES
Abnormal vol/occ pattern: Need on site check
DevIndex
> 13
YES
NO
Marginal; Need further observation
No problems found. Healthy Detector

Figure 4: Decision Tree for Loop-Detector Diagnostics and Classification

3.5 Test using Mn/DOT Loop Repair Record

Testing of the developed detector classification and diagnostic algorithm has been carried out with the help of staff in Mn/DOT's Regional Transportation Management Center (RTMC) who manage all loops in the Twin Cities' freeway. The RTMC has been maintaining loop repair logs called Loop Repair Record in which the date of reported incidents, types of problems, and repair records have been kept. This report was used as the test bed for the algorithm.

The first 100 cases shown in the repair record from 08/02/2001 to 10/31/2003 were used for testing the algorithm. The basic approach used is to check before the repair date and after repair date to see any difference is detected. The result is summarized in Table 2, which is organized by the number of cases reported, cases detected by the algorithm, cases missed by the algorithm, and the cases shown no visible difference found among the missed by the algorithm (possible misreporting cases). Depending on the types of problems, the performance of the algorithm varied. The algorithm performed well on detecting No hits, High counts, Road damaged, Lead in cable bad, Need replacement, and Wired new case reports, but it did not perform well on detecting PM (Preventive Maintenance), Flow spikes, Splice, Bad counts, Swapped, Faulted, and Separate from another loop cases.

Subtracting possible misreporting cases (i.e., removing the cases with no visible differences), the algorithm detected 56 out of 74 reported cases, which is about 76% detection rate. This detection rate is somewhat misleading. The actual detection rate could have been much higher if the loop repair record was accurate and complete. Several points to consider are discussed. First, the algorithm actually identified many more cases than the problems reported in the repair record, i.e., only a small fraction of the actual detector problems was recorded in the loop repair record. Second, some of the problems such as Preventive Maintenance and New Splice are not logically detectable by software. Third, for some detectors multiple repair visits were made for the same problem implying that before and after repair does not exactly reflect the health status change of the detector. Forth, some problems such as Flow spikes could have been detected if the detection threshold level was lowered. We draw the test results only using the default threshold values. However, the bigger problem was that visual and data inspection of volume changes showed that the spikes reported as a problem were within a normal range suggesting that the repair request should not have been issued. In conclusion, the loop repair record itself was not entirely reliable data for testing the performance of the algorithm. Nevertheless, it provided us some good insights on how the algorithm performed and what are the limitations when we inspected specific cases.

Table 2: Algorithm Detection Results

Type	Reason in the report	Cases reported	Detected by s/w	Missed by s/w	No visible difference
1	No hits	24	20	4	2
2	PM	11	2	9	6
3	Occ spikes	8	4	4	3
4	Lock on	8	4	4	1
5	Chattering	7	3	4	2
6	Low counts	7	4	3	2
7	High counts	6	6	0	0
8	Road damaged	5	5	0	0
9	Flow spikes	4	0	4	3
10	Splice	4	0	4	3
11	Bad counts	3	0	3	2
12	Lead in cable bad	4	4	0	0
13	Pulse mode	2	1	1	0
14	Swapped	2	0	2	1
15	Faulted	1	0	1	0
16	Needs replacement	1	1	0	0
17	Wired new loop	1	1	0	0
18	Separate from another loop	1	0	1	1
19	No reason	1	1	0	0
	Total	100	56	44	26

3.6 Conclusion

This paper presented an algorithm for identifying loop-detector problems that require a repair. Development of this algorithm was intended for quickly identifying and summarizing loop problems from a large pool of loops such as the loops in a freeway network managed by metropolitan TMCs. The algorithm is not perfect yet but demonstrated that it is a viable tool to identify many loop problems. The measurement parameters and the decision tree developed are fairly extensive, but we feel that further refinements can be made by improving the decision tree and incorporating more sophisticated measurement parameters. We are presently working in that direction by working together with the traffic management group and the detector maintenance group at the Mn/DOT RTMC. Another benefit of this algorithm worth mentioning is that, since the overall algorithm goes through a more comprehensive and thorough test, it can provide more accurate identification of erroneous data for traffic data applications.

CHAPTER 4: MN/DOT CONTINUOUS AND SHORT-DURATION COUNT COMPUTATION

4.1 Introduction

4.1.1 Mn/DOT's Traffic Monitoring Program[2]

Traffic monitoring programs have been one of the important functions in state and federal level transportation departments. The Minnesota Department of Transportation (Mn/DOT) has been maintaining an active traffic monitoring, forecasting and analysis program. Mn/DOT has been responsible for collecting, analyzing and publishing traffic count, classification and weight data from the various roadway systems throughout the state. These traffic data have a wide variety of users including five of the six federally mandated management systems in Mn/DOT. Elements of today's Mn/DOT Traffic Monitoring System (TMS) are administered cooperatively through the efforts of three separate Divisions[3] and all the District Offices:

- Program Support Group,
- Program Delivery Group,
- State Aid for Local Transportation Group, and
- Metropolitan Division.

The Traffic Forecasting and Analysis Section (TFAS) of the Program Support Group has been planning and administering the Department's traffic monitoring program. Today's Mn/DOT's Traffic Monitoring System (TMS) is a product of ongoing automation activities designed to improve traffic data quality and timeliness for traffic volume data users. Although the objectives may vary over time, the central premises of these efforts are the following:

- The TMS must be based on statistically valid principles
- The TMS must use data systems that integrate all necessary data types
- Traffic data should be collected, processed and reported in electronic form. Manual aspects of TMS operation should be minimized.
- Lines of communication must be established and maintained between those involved with the TMS and the customers using information coming from it.

[2] This portion was written based on a two-page summary of the Mn/DOT's traffic counting program published in Summer 1996 and additional information provided by the Traffic Forecasting and Analysis Section of Mn/DOT (Mark Flinner).

[3] At the time of this writing, Mn/DOT was going through reorganization, thus names and divisions provided here only represent divisions as they were before the reorganization.

- The TMS must be dynamic and flexible in order to take advantage of new methodologies and technologies that bear on traffic data.

These premises and TFAS' long-term objectives served as the main impetus for creating a project that would extend the degree of automation for the traffic volume data collected by the MN/DOT's Traffic Management Center (TMC) in the Metropolitan Division.

One measure of roadway use is the Annual Average Daily Traffic (AADT) volume. These estimates represent how many vehicles are traveling on the state's roadway segments (in both directions) on an average day of the year. These traffic volume data are derived from two kinds of traffic counting activities. The first involves continuous traffic counting devices or ATRs (automatic traffic recorders), which record hourly volume data 24 hours a day throughout the year. The second involves short-duration counting devices such as road tubes and manual or portable automatic vehicle classification devices. Data collected from these counting activities are screened, factored if necessary, and analyzed to create AADT volumes that are mapped and distributed for use by the Federal Highway Administration (FHWA), MN/DOT, county and local highway departments, and area planning organizations. Additionally, private sector business consultants, engineering firms, and real estate interests, among others frequently request and use the department's traffic volumes in their work.

Mn/DOT's ATRs are located primarily on trunk highways throughout the state. Traffic volumes are retrieved from these ATRs by the TFAS staffs several times a week. The ATR data are nominally screened using a SAS program. They are then output into a format that is suitable for loading into the MN/DOT's Traffic Analysis Expert System (TAES). Analysts then edit the ATR data using the TAES to check for equipment malfunctions, to cull out bad data, and to synthesize data where data are missing. After the ATR data have been edited, they are ready to be used for reports and to create seasonal/day-of-week adjustment factors for the short-duration count data collected at approximately 32,000 locations throughout the state.

The short-duration count data are collected using portable data collection devices such as pneumatic road tubes for a minimum of 48 hours duration. Where there are permanent sensors available (such as are managed by the TMC), short duration samples are manually taken from the loop sensor data files and sent to the TFAS staff. Every short duration count is manually entered into a relational data base management system programmed in R:BASE[4], and is further adjusted by the seasonal/day-of-week factors that are derived from the ATR data. After the short-duration count data are entered into the database, they are evaluated against past AADT estimates, and recounts are ordered when anomalous data values, equipment malfunctions, or tube set failures indicate the need for a recount.

At the end of the counting season, the short-term counts are evaluated for spatial and temporal coherency and placed on draft traffic volume maps. The draft maps are circulated to Mn/DOT district and/or county and municipality engineers for feedback. Final traffic volume maps are then prepared and distributed to Mn/DOT's traffic volume data users. The Department's TFAS has already published the Department's first automated traffic volume map, and anticipates automating its entire traffic volume mapping process in the future using a combination of CADD technology, database

[4] Information on R:BASE can be found in the web site "www.rbase.com."

integration, and correlation to the department's GIS base map. The automation process will also enhance the Department's ability to examine concurrently both spatial and temporal changes in trunk highway use. AADT are automatically loaded into the department's computerized Transportation Information System (TIS) annually. In addition, as the supporting GIS map base is completed, TFAS plans to make its traffic volume data available electronically throughout the state. Traffic volume maps are already available on the Department's web site to facilitate dissemination of the department's AADT traffic volume information.

4.1.2 Project Goals

4.1.2.1 Background

The Metropolitan Division's Traffic Management Center (TMC) monitors and manages traffic on the metro area freeways and arterial highways and is responsible for collecting the traffic data for the roads under their management. The TMC maintains and collects volume and occupancy data from about 4,000 inductive loop detectors at a constant rate of 30 seconds through an extensive network of detectors and computerized data communication. This type of traffic data is often referred to as ITS (Intelligent Transportation Systems) traffic data and has been used for traffic control and monitoring operations at TMC. For most state and local transportation departments, ITS traffic data is a largely untapped resource for traditional traffic counting programs, although it provides a rich set of data that could be used for traffic counting (Schmoyer, 2001). The problem lies in that ITS data is susceptible to outliers, missing values and other types of data anomalies that are not easy to resolve. Moreover, since the data is collected at 30–second intervals, the amount of data is substantially large and difficult to manage using simple desktop PC tools. Nevertheless, since the 1980s, Mn/DOT has tapped into the ITS traffic data and has produced the short-duration and continuous count data through manual compilation for the locations along instrumented metro freeways. As part of the TFAS's on-going efforts to integrate and automate the Department's traffic monitoring program, the present project was developed in collaboration with the Transportation Data Research Lab at the University of Minnesota Duluth. Simply stated, the project aimed to provide well screened and high quality data for the TMC portion of the ATR and short-duration traffic data.

The TFAS has established unique sequence numbers (different from the TMC's station numbers) that identify traffic counting locations throughout the state. The traffic locations from TMC consist of a primary set of detectors that typically represent a segment of the road. Along with the sequence numbers, a new concept was introduced in this project, which allows designation of alternative sets of detectors for each station in order to improve the quality and reliability of the data. Therefore, a station, whether it is ATR or Short-duration Court (SC) station, was allowed to be assigned up to three sets of detectors, which are referred to as primary, secondary and tertiary detector sets. If the volume count collected from the primary of a station is disqualified from the acceptance test, the data from its secondary replaces the data of primary and works similarly for the secondary and tertiary. The central mechanisms of the methodology employed in this

project are prioritized choices of detector sets for spatial inferences along with multi-level Bayesian data imputations utilizing temporal relations.

4.1.2.2 Project Goals

The objective of this project was to develop an automated system for the TMC portion of ITS traffic data contributed to the Mn/DOT's TMS. The system should provide all automated acceptance tests required by TFAS for both continuous-count and short-duration count volume data. The system should provide data formatted according to the Mn/DOT's SAS and R:BASE application input requirements and should automatically transfer data to the TFAS server. The automation system should work with minimum human intervention.

4.2 Overview of the system

As described in Section 4.1.2.2, the goal of this project was to extend the present Mn/DOT's TMS automation efforts for the TMC's ITS traffic data portion. In order to achieve this objective, the overall system was designed around on-line availability of data through Internet connections. The main linkage to establish was an on-line relation between the data production capability of the Data center at the UMD TDRL and the servers at the Mn/DOT TMS.

Although the overall system was designed based on a multi-tiered architecture, at the conceptual level, it may be described as a blackboard concept in a classroom. This relationship is illustrated in Figure 5. Data can be written to or read from the blackboard server by TDRL Data Center or Mn/DOT TMS servers. The arrow lines indicate Internet data connections, and the sequence of data flow works as follows. Files that include the detector lists that specify primary, secondary and tertiary detectors for ATR and SC stations are posted by Mn/DOT on the blackboard server. The TDRL Data Center servers download the detector list files from the blackboard server in order to use them to compute the SC and ATR data. If multiple versions of detector list files were uploaded, Mn/DOT may specify which file to use. The Data Center servers then produce the required data (AADT and ATR data), and post the data on the blackboard server. For clear distinction, the data is always transferred in a file form with a file name that includes year, moth, and day information. The Mn/DOT servers or analysts regularly monitor the data files and download the files into the TMS when the data are available. In all cases, the file names on the blackboard include date information along with data type to prevent any conflict or confusion in the data version or usage of the data.

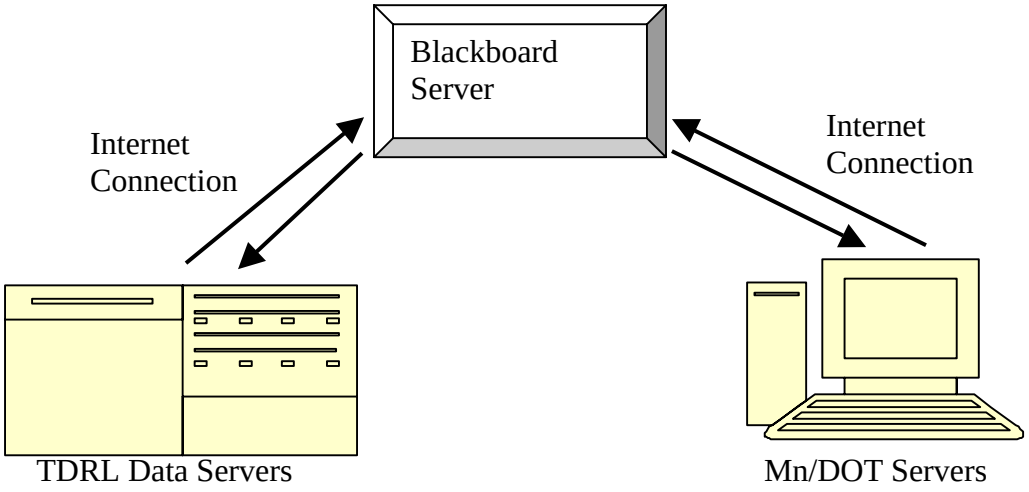


Figure 5: System level concept of data automation

4.3 Treatment of Missing Data

4.3.1 Introduction on Missing Data

As with most real-world data, ITS traffic data contains missing and incorrect data. In fact, since ITS traffic data are collected 24 hours a day throughout the year using computerized data collection systems, presence of data-loss due to hardware malfunction at the site or along the transmission lines is a high probability. More specifically, construction, power outage and temporary maintenance operations are unavoidable aspects, which mostly likely lead to a data loss. Missing data itself could provide us a great deal of information about the loop detectors, reliability, maintenance requirements and the expected quality of data. However, for traffic counting purposes, estimating the missing data is essential.

Attempts have been made with some success, to estimate missing data in a collection of ITS traffic data. Research at the Texas Transportation Institute (TTI) has explored regression analysis in combination with the Expectation-Maximization (EM) algorithm and compared the results with those from simple techniques such as straight-line interpolation and "*factor-up*" on traffic data (Gold, 2001). The results are very encouraging. The EM algorithm, however, is rather computationally intensive and, as the researchers conclude, the marginal improvement in performance did not weigh well against the time and effort that goes into the implementation of the EM algorithm. Moreover, treatment of larger blocks of missing data has not been addressed in their study, a potential problem with EM. Schmoyer et al. (Schmoyer, 2001) proposed a simple filtering approach for detecting missing data and linear regression estimates for the treatment of missing data. Again, this approach does not address large blocks of missing data. A school of time series estimation and filtering approaches exists, which has been known to be effective in recovering missing data or removing noise from band-limited signals (Box , 1994; Chatfield, 1996; Naidu, 1996; Warner, 1998). Since most ITS traffic data are obtained by sampling data at a constant rate such as 30 seconds or 5 minutes, they are indeed a time series and could be applied to the vast array of available time-series algorithms. However, no direct study results on traffic data are presently available to the best knowledge of the author.

Many rigorous research works on imputing missing data have been conducted in the field of statistics for applications in social science survey data, since such data most likely contain non-responses. Little & Rubin (Little, 1987) essentially developed and laid foundations on the analysis of multiple imputation approaches on non-response survey data and suggested a number of statistical models based on historical inferences. These pioneering works are mostly based on likelihood estimates derived from formal statistical models. Schafer extended the analysis to incomplete multivariate datasets with continuous and discrete variables and applied EM algorithms and Monte-Carlo based Markov chain approaches. In a broad sense, the approaches mentioned can be called Bayesian approaches, since they explicitly use probability for quantifying uncertainty in inferences based on statistical data analysis (Gelman, 1995).

This chapter describes classification of missing data patterns and the treatment of them as developed in this project.

4.3.2 Classification of Missing Data Patterns

4.3.2.1 Spatial and Temporal Characteristics of Traffic Data

Before investigating the missing traffic data patterns, it is important to recognize that traffic data inherently holds spatial and temporal relationships if it is comprised of data from multiple detectors in multiple locations. Spatial relation refers to a geographical relation of detectors, and it may be characterized using the size of geographical area from a smaller to a larger scale. For example, detectors could be characterized as detectors in a station[5], a road, a county, or a state. Similarly, temporal relation may be described using an increasing time-scale such as seconds, hours, days, months, and years. These inherent relations could be used as a reference for how to classify the missing data patterns. For example, data may be missing at a different spatial level such as a detector (lane) or a station (directional total) level, or at a different time scale such as minutes or hours. The challenge is how to effectively combine both the spatial and temporal characteristics into one uniform representation.

4.3.2.2 Classification by a Tree Structure of Missing Data Patterns

In order to investigate missing patterns in the TMC traffic data, we observed statistics on stations for year 2001. Figure 2 shows missing data statistics for a typical station based on counting of days for missing percentage per day for the year 2001. Notice that the number of days containing more missing data in a year decreases as the percentage of missing increases. Based on this observation and the characteristics of traffic data discussed in Section 4.3.1.1, we found that the missing patterns fall into a leaf of a tree structure illustrated in Figure 7. This tree structure of missing data patterns was taken into account in designing the overall imputation strategy.

[5] A station is formed at a location of road where loop detectors are installed at each lane to observe the sampled view of the traffic flow in that road.

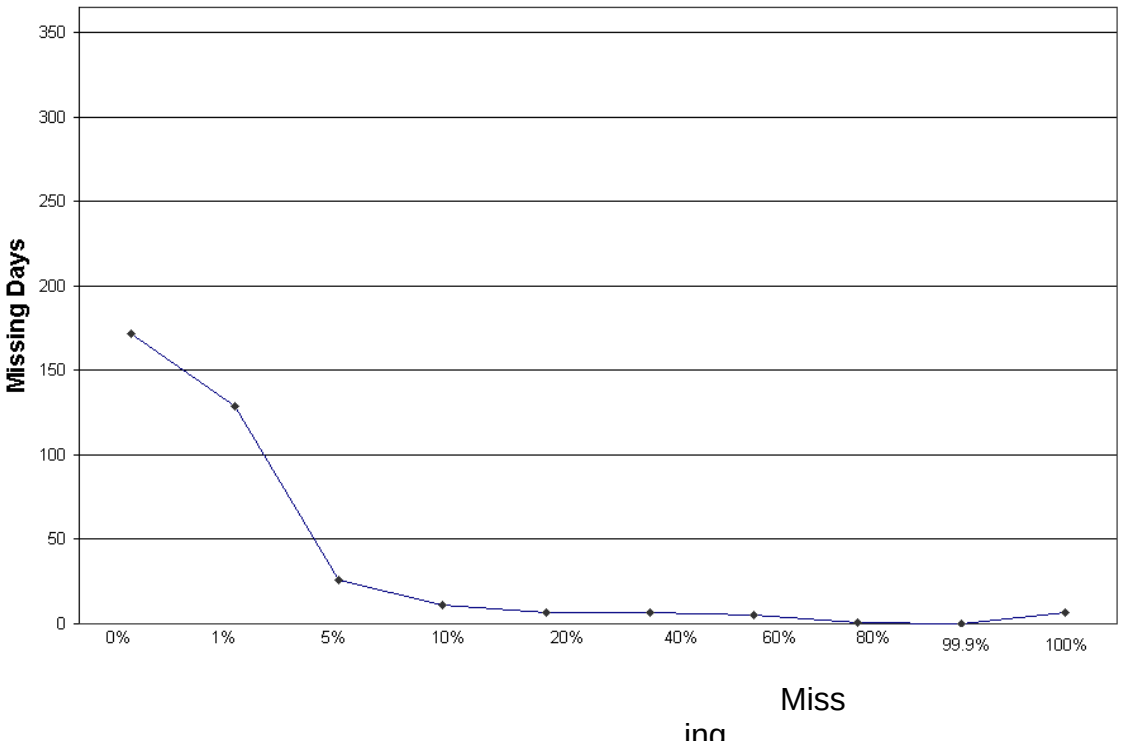


Figure 6: Typical annual missing percentages of a station (station number 1078E)

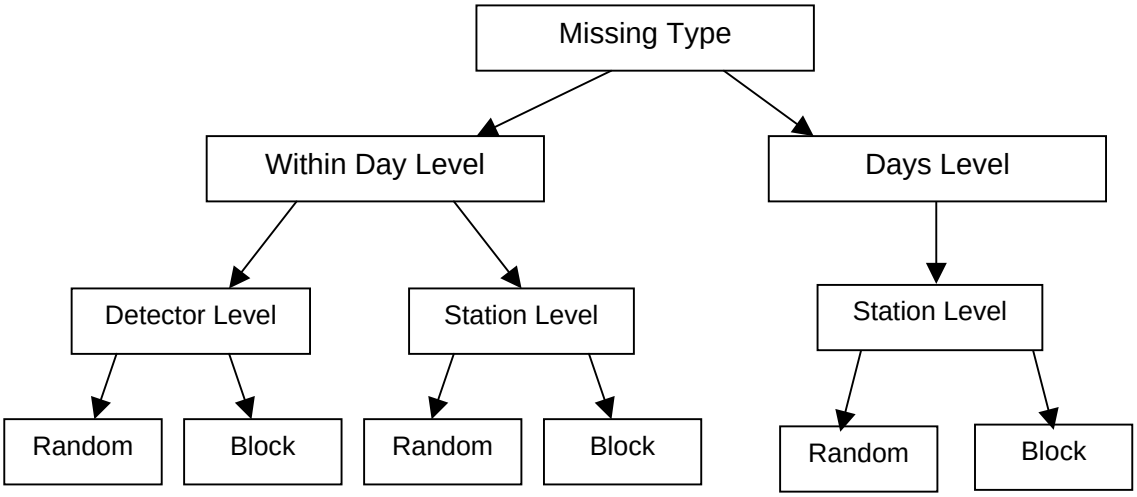


Figure 7: Classification of missing patterns in a tree structure

As shown in Figure 7, missing patterns fall into one of the branches of the tree. At the top level, we classify the missing data types into two types, either the whole days are missing, or a part of a day is missing. If only a part of the day's data is missing, we further divide it into two missing types in spatial relation, i.e., a part of detectors data is missing, or the whole directional station data is missing. The next level down is classified based on the occurrences of random missing or blocks of missing (a block means a group of consecutive data). For the day level, the missing data patterns occur either at random or in blocks of days but are only classified at the station level, since the detector level overlaps. Also, since our objective is computing AADT at the station level, treatment of station level covers the needs of the algorithm development and missing data pattern analysis. For convenience of description, we name each leaf of the tree from *Type A* to *F* from left to right branches. The basic idea of our imputation strategy is the following: when data imputation is started from *Type A* and progressed towards *Type F*, each stage ends up supplying more data for the next level, providing further inference. Below, we further clarify missing data relations in detector/station level and random/block level.

Detector or Station Level Missing

This distinction occurs due to the spatial relationship of detectors. In a station, only one or two detectors could be broken and produce missing or incorrect data. Such cases exist due to a partial construction or maintenance operation of roadways or breakage of loop wires by cracks. In other cases, all of the detectors in a particular station can be broken, which leads to station-level missing data patterns. Station level missing data also happens because the detectors in a station are usually connected to a single controller box that sends data to the central data collection server. Therefore, if a

controller malfunctions (e.g., looses power or communication link), the result becomes station-level missing data pattern.

Random or Block Level Missing

Random or block level missing data is determined using a temporal relationship of missing data patterns. Random missing data refers to missing values that occur completely randomly. This is equivalent to ignorable non-response data in statistics where many multiple imputation techniques have been applied (Little, 1997). In general, random missing data are caused by transient hardware or software problems that are difficult to identify and correct. Therefore, we always have to expect existence of random missing data patterns in traffic data. Block missing data refers to missing values that occur in a consecutive blocks of data in temporal relationship. Although a high density of random missing data theoretically can lead to block missing data, such rarely happens in real data. Most block missing data occurs in a long sequence of data such as half day, few months, or whole year in some cases according to our observations. In the real world, construction of a segment of a road frequently occurs for an extended time period during the summer construction season, which leads to a long sequence of block missing. This type of missing data pattern cannot be imputed using the techniques used in random missing data (Little, 1997; Rubin, 1987). This type of missing data pattern is more difficult to impute or estimate due to limited inferences.

4.3.3 Multiple Imputation

Multiple imputation (MI) is a statistical technique for analyzing incomplete data sets, that is, data sets for which some entries are missing. Each missing datum is replaced by $m>1$ simulated values, producing m simulated versions of the complete data. Each version is analyzed by standard complete-data methods, and the results are combined using simple rules to produce inferential statements that incorporate missing data uncertainty (Rubin, 1987).

Rubin (Little, 1997; Rubin, 1987) presented a method for combining results from a data analysis performed m times, once for each of m imputed data sets, to obtain a single set of results. From each analysis, one must first calculate and save the estimates and standard errors. Let Q be the quantity of interest, such as the mean of population. Suppose that $\hat{Q}_j$ is an estimate of a scalar quantity of interest (e.g. a regression coefficient) obtained from data set j (j=1, 2, ... , m) and U_j is the standard error associated with $\hat{Q}_j$. The overall estimate is the average of the individual estimates,

$$\bar{Q} = \frac{1}{m}\sum_{j=1}^{m}\hat{Q}_j \qquad (1)$$

For the overall standard error, one must first calculate the within-imputation variance,

$$\bar{U} = \frac{1}{m}\sum_{j=1}^{m}U_j \qquad (2)$$

and the between-imputation variance,

$$B = \frac{1}{m-1}\sum_{j=1}^{m}(\hat{Q}_j - \bar{Q})^2. \quad (3)$$

The total variance of $(Q-\bar{Q})$ is

$$T = \bar{U} + \left(1+\frac{1}{m}\right)B. \quad (4)$$

The overall standard error is the square root of *T*. Confidence intervals are obtained by taking the overall estimate plus or minus a number of standard errors, where that number is a quantile of Student's t-distribution with degrees of freedom

$$df = (m-1)\left(1+\frac{m\bar{U}}{(m+1)B}\right)^2. \quad (5)$$

A significance test of the null hypothesis Q=0 is performed by comparing the ratio

$$t = \frac{\bar{Q}}{\sqrt{T}} \quad (6)$$

to the same t-distribution. Additional methods for combining the results from multiply imputed data are reviewed by Schafer (Schafer, 1997).

4.3.4 TDRL Algorithms

Little and Rubin suggested several imputations that are defined statistically proper (Rubin, 1987). In this project, one of them referred to as the nonnormal Bayesian imputation procedure that is proper for the standard inference was adapted. This section describes the detailed algorithms developed for this project.

4.3.4.1 Nonnormal Bayesian Imputation Algorithm

According to Rubin's analysis, many Bayesian models beside the normal approximately yield the standard inference with complete data, and thus many such models can be used to create proper imputations for ignorable nonresponse. He suggested the following algorithm:

Algorithm 1: Nonnormal Bayesian Imputation
Input: Observed Values $(Y_1,...,Y_n)$
Output: *M* Imputed Values

Step1: Draw (*n-1*) uniform random numbers between 0 and 1, and let their ordered values be $(a_1,...,a_{n-1})$; also let $a_0 = 0$ and $a_1 = 1$.

Step2: Draw each of the M missing values by drawing from $(Y_1,...,Y_n)$ with probabilities $(a_1 - a_0),(a_2 - a_1),...,(1 - a_{n-1})$.

4.3.4.2 Imputation of Randomly Missing Data Patterns

Whether data is at the detector or station level, random data missing implies randomness of the occurrences and thus availability of observable data in the neighborhood of missing data patterns. While missing data samples are randomly located and unpredictable, traffic volume counts during the day approximately follow distinctive patterns that repeat over and over again. More specifically, it has a camel back pattern; that is, traffic volume is generally very low from mid night to about 5:00am, and then it is gradually increased as time approaches towards morning rush hour. During the morning rush hour, traffic volume reaches the morning peak and then it is decreased again but not as much as the midnight. In the afternoon it reaches another peak. In order to incorporate such time dependent patterns while maintaining the variability, we devised an algorithm that combines linear regression with a Nonnormal Bayesian imputation (Rubin, 1987) for imputing randomly missing data patterns. We refer to this algorithm as the Nonnormal Bayesian Linear Regression (NBLR) algorithm and it is presented below. The basic idea follows Rubin's suggestion on creating nonignorable imputed values using ignorable imputed models (Rubin, 1987). Let a sequence of volume counts in *n* elements that includes *m* missing values be denoted by

$$V = (V_{x_1}, V_{x_2}, ..., V_{x_k}, V_{x_{k+1}}, ..., V_{x_{k+m}}, ..., V_{x_n}).$$

It is a consecutive portion of volume data taken around the missing values where one or more observed data exist. The observed *(n-k)* values are denoted as $V_{obs} = (V_{x_1}, V_{x_2}, ..., V_{x_n})$, and the missing values are denoted as $V_{mis} = (V_{x_k}, V_{x_{k+1}}, ..., V_{x_{k+m}})$.

Algorithm 2: Nonnormal Bayesian Linear Regression (NBLR) Imputation
Input: *V*
Output: estimate of missing values $\hat{V}_{x_k}, \hat{V}_{x_{k+1}}, ..., \hat{V}_{x_{k+m}}$

Step 1: Find the parameters of a linear regression model given by $\hat{y}_{x_i} = \hat{\beta}_0 + \hat{\beta}_1 x_i$ using V_{obs}.

Step 2: Construct a random variable D_{obs} using the difference between the regression estimate and the observed values, that is,

$$\begin{aligned} D_{obs} &= (V_{x_1} - \hat{y}_{x_1}, V_{x_2} - \hat{y}_{x_2}, ..., V_{x_n} - \hat{y}_{x_n}) \\ &= (d_{x_1}, d_{x_2}, ..., d_{x_n}) \end{aligned} .$$

Step 3: Draw *M* imputed values for each missing values by applying D_{obs} to Algorithm 1 and then compute the estimate of missing values as:

$$\hat{V}_{x_k} = \hat{y}_{x_k} + \tilde{d}_{x_k}$$

where $\tilde{d}_{x_k}$ is the average of *M* imputed values.

This algorithm essentially utilizes the inferences in time trend of traffic volume using the observed values through the linear regression model while the nonnormal Bayesian drawing of values capture the statistical inference of the observed values. The effect of the algorithm is illustrated using a real data example in Figure 8 by showing before and after imputation. The data used is station data with 5-minute intervals for a day. Notice that the algorithm clearly captures the time trend as well as the statistical variability and fills in the missing values. Many other cases tested resulted in a similar outcome.

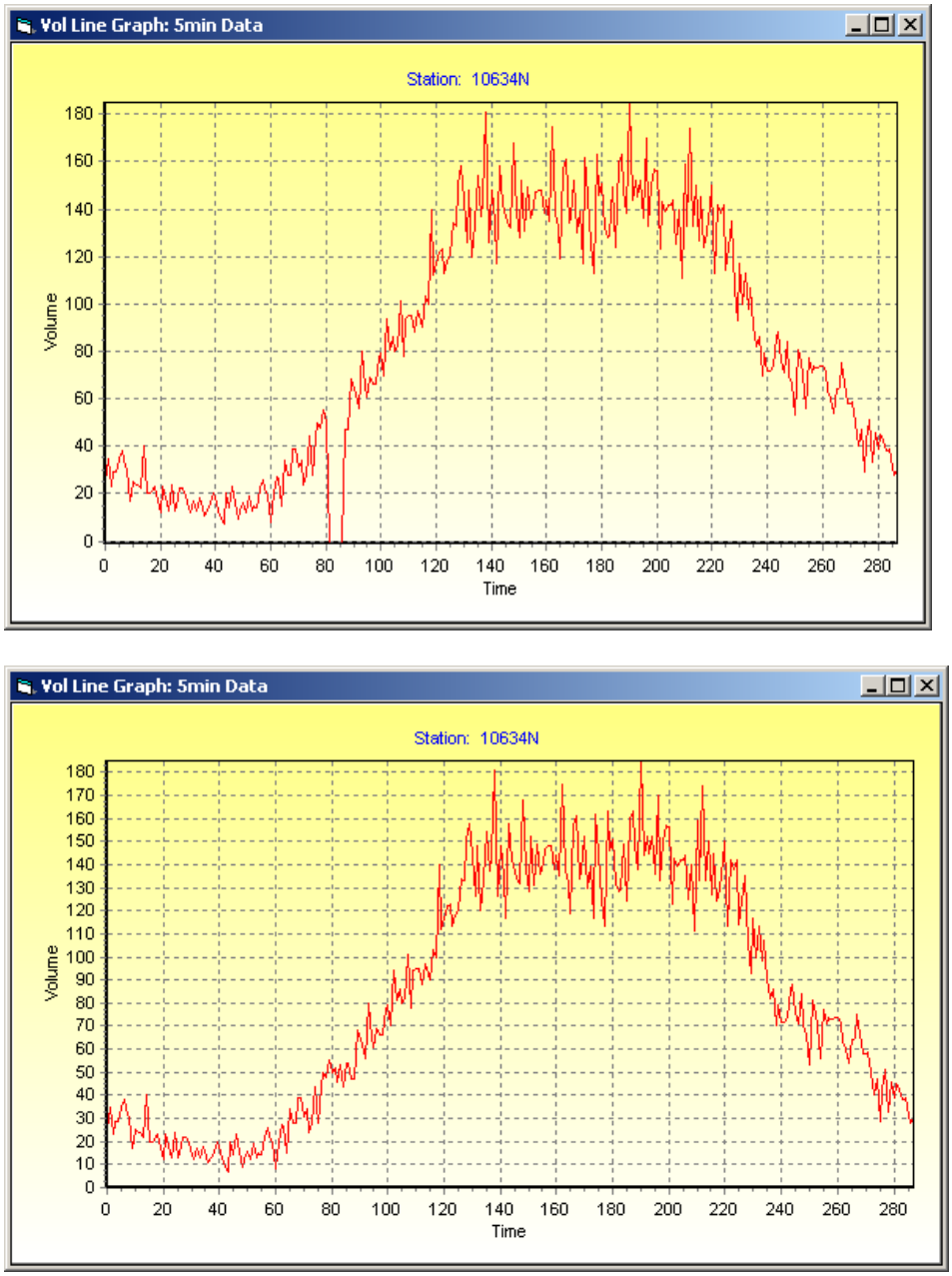


Figure 8: Effect of NBLR: before imputation (top) and after imputation (bottom)

4.3.4.3 Imputation of Block Missing Data Patterns

Block missing data refers to existence of a large amount of consecutive missing values in the data, such that neighboring values can no longer provide enough time trend inferences. In this case, the NBLR algorithm in Algorithm 2 cannot be used since the time trend inferences are not available. Therefore, some other inferences must be used. In traffic volume data, one can easily observe repeated patterns in the same day-of-week in surrounding weeks except for holidays and near holidays. For example, if a block of data is missing on Monday of 13th week of the year, the traffic during the missing block is likely similar to Mondays of 10th, 11th, 12th, 14th,15th and 16th weeks as long as the Monday is not holiday or near holiday. Based on these existing inferences, block missing data patterns are imputed using the following algorithm.

Algorithm 3: Block Level Nonnormal Bayesian Imputation
Step 1: Identify the beginning and end time of the block of missing data.
Step 2: Create an array of observed vectors using the same time block of the missing block on the same day-of-week from *M* previous weeks and *M* following weeks (*M* is usually a small number such as four or five), i.e., $B_{obs} = (B_{w_1}, B_{w_2}, ..., B_{w_{2M}})$ where B_{w_i} denotes the same time block of the volume data on the same weekday of previous or following weeks. If the same weekday of any of the chosen weeks is a holiday or near holiday, the data from that week is excluded.
Step 3: Using B_{0bs} draw *m* blocks by applying the NBI algorithm (Algorithm 1) and replace the missing block with the average of the *m* drawn blocks.
Again, the effectiveness of Algorithm 3 is illustrated using an example. Notice from Figure 5 that block of missing data (about six hours) was restored with high fidelity, which can be seen from the continuity of the data at the beginning and end of the day (or see another day like this one containing all “good” data).

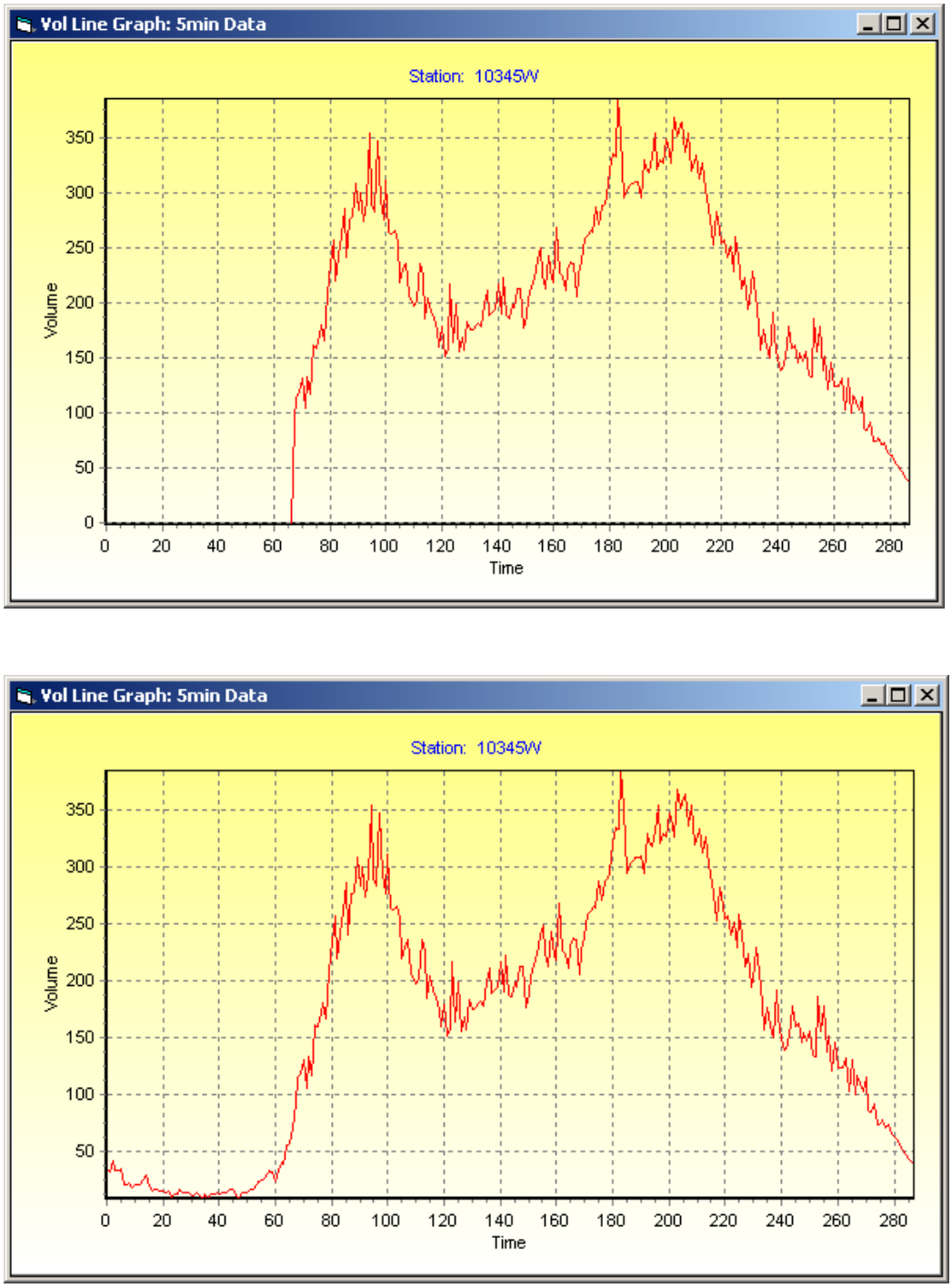


Figure 9: Effect of Block Imputation by Algorithm 3: top graph shows before block imputation and bottom graph shows after block imputation.

4.4 Implementation

This chapter describes implementation details of this project for two purposes. First, it is intended to serve as a record on how the actual data were produced. Second, it is intended to clarify what has been done and what has not been done so future developments and modifications can be made with reference to this work. This chapter will also include a description of software tools that have been developed for this project.

4.4.1 Continuous Count Data

Continuous count data from ATR stations are simply ordered lists of hourly traffic volume counts that consists of 12 entries for AM and 12 entries for PM per day. Each hour's data represents the total volume count of a station during the corresponding hour, which is the total volume count of individual loops for that station in an hour. The data are recorded seven days a week throughout the year on a continuous basis.

4.4.1.1 Continuous-Count Data Source

Traffic data has been supplied by TMC to the Data Center of TDRL through an automated on-line daily uploading. These data sets contain a binary form of volume and occupancy collected at 30-second intervals from all detectors that TMC manages. For each detector, there are two files consisting of 2,880 elements. One file contains volume and the other, occupancy. Since TMC manages 3,500 to 4,000 detectors (it varies over time), the total number of data files per day is between 7,000 to 8,000. For exchange and archive, this large number of files is zip-compressed into a single file that is then transmitted to the TDRL Data Center. The compressed file size is typically 15 MB (Mega Bytes); when uncompressed its size becomes about 32MB.

4.4.1.2 Detectors in Continuous Count Stations

For continuous stations, three prioritized detector sets are always defined according the equivalency relation of traffic flow. These detector sets are referred to as primary, secondary and tertiary detector sets denoting higher to lower priorities, respectively. In principle, the three detector sets must have the same amount of traffic flow in spatial relation. A lower priority detector set is used as an alternative detector set if the acceptance tests on the higher priority detector set fail. Detector identification numbers are expressed as either negative or positive integers. The positive numbers instruct the computing algorithm to add the detector volume to the station volume; the negative numbers instruct the algorithms to subtract the detector volume from the station volume. However, station volume must always be a non-negative integer.

4.4.1.3 Station Identification Database

The location of traffic measurement, that is the location of a station, may change over time. Likewise detectors assigned to a station may change. Since the detector locations and numbers as well as the stations themselves go through modifications, there is a need for a flexible means that could keep track of those changes, provide easy maintenance, and allow retrieval of any necessary combination of station information. For this purpose, a relation database (Microsoft SQL Database Engine™) was selected for the first choice of technology. This database was called the station identification database. During Task 1, this database was designed and developed to accommodate the required station management functions for both ATR and SC stations as well as for future applications such as a Geographical Information System (GIS). The database comprises two linked tables: Station Table and Detector Table (the detailed columns are shown in

Appendix A). The Station Table maintains the geographical data, names, identification numbers and who and when modified. The Detector Table maintains all detectors allocated for all stations that are linked to the Station Table. Since the main users and custodians of this database are located at the central office of Mn/DOT in St. Paul, the database must be accessible from remote locations with good security measures. Therefore, a web interface that allows only indirect access to the database was developed. The tools used for this web application are a Zope server, Python language and Java Scripts. Example screen captures of these web interfaces are shown in Figures 10 - 12. The development of this part of the project was completed during the summer of 2002.

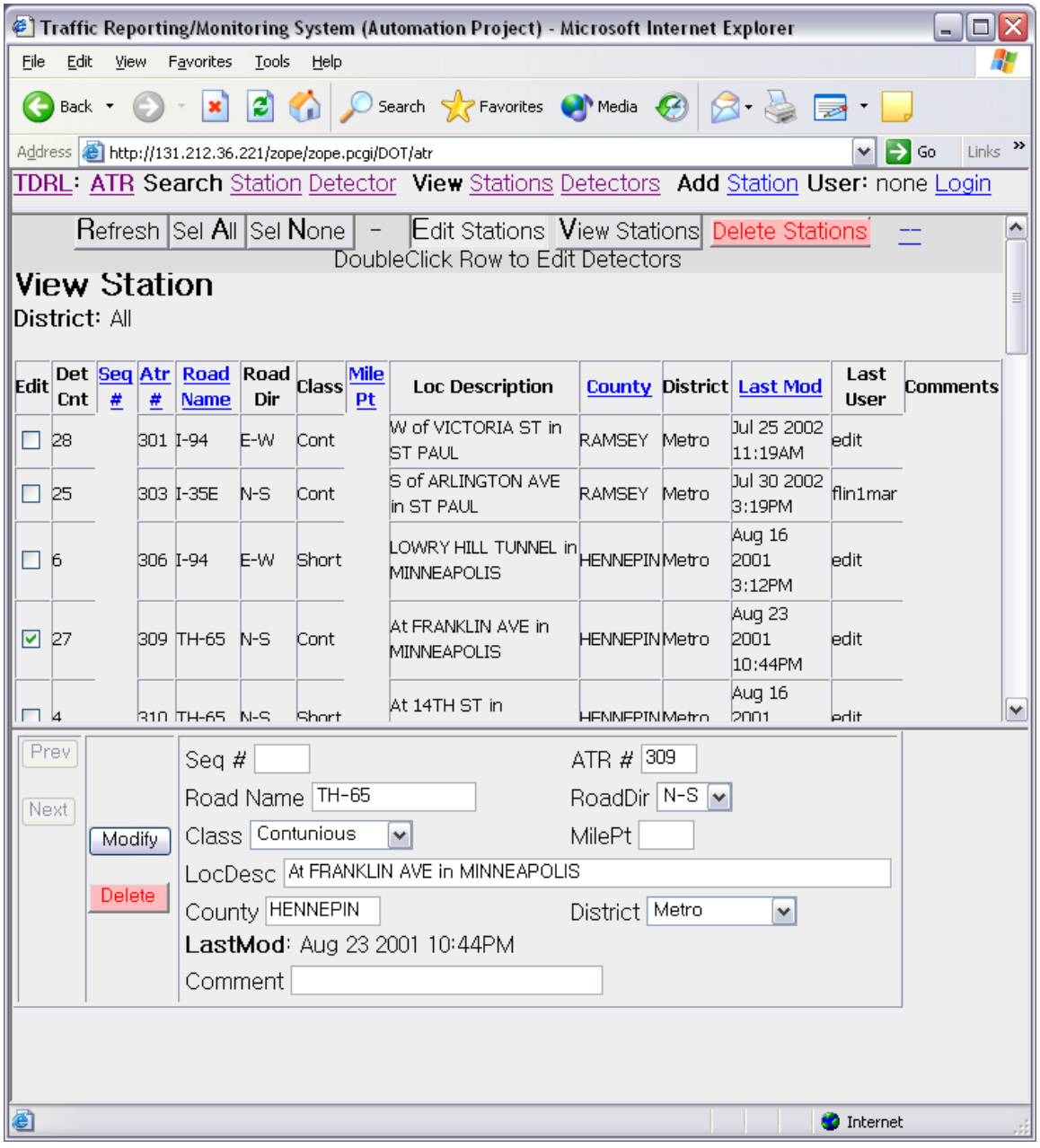


Figure 10: A sample screen capture of web interface: station table edit function

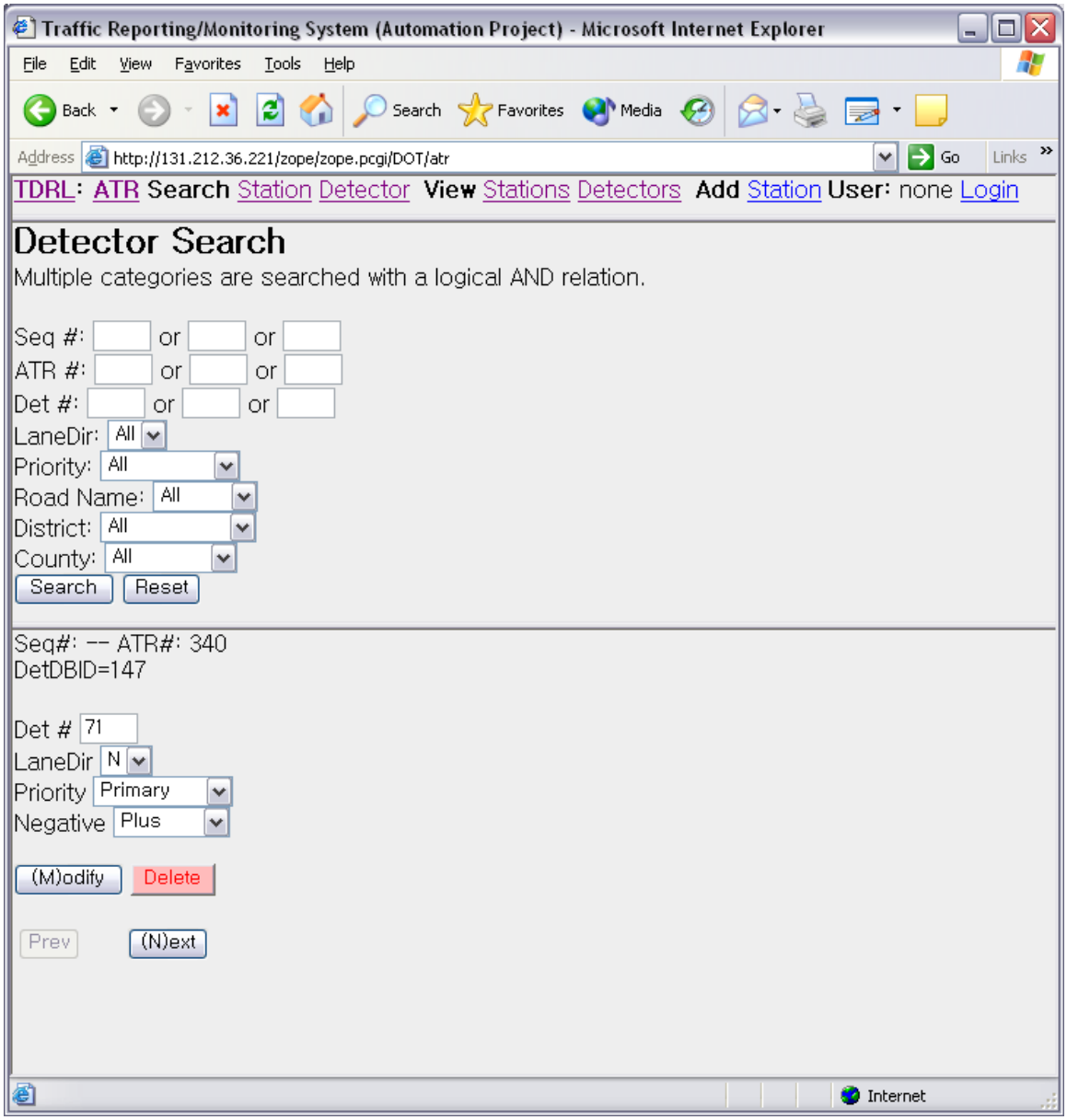


Figure 11: Sample screen capture of web interface: Detector edit

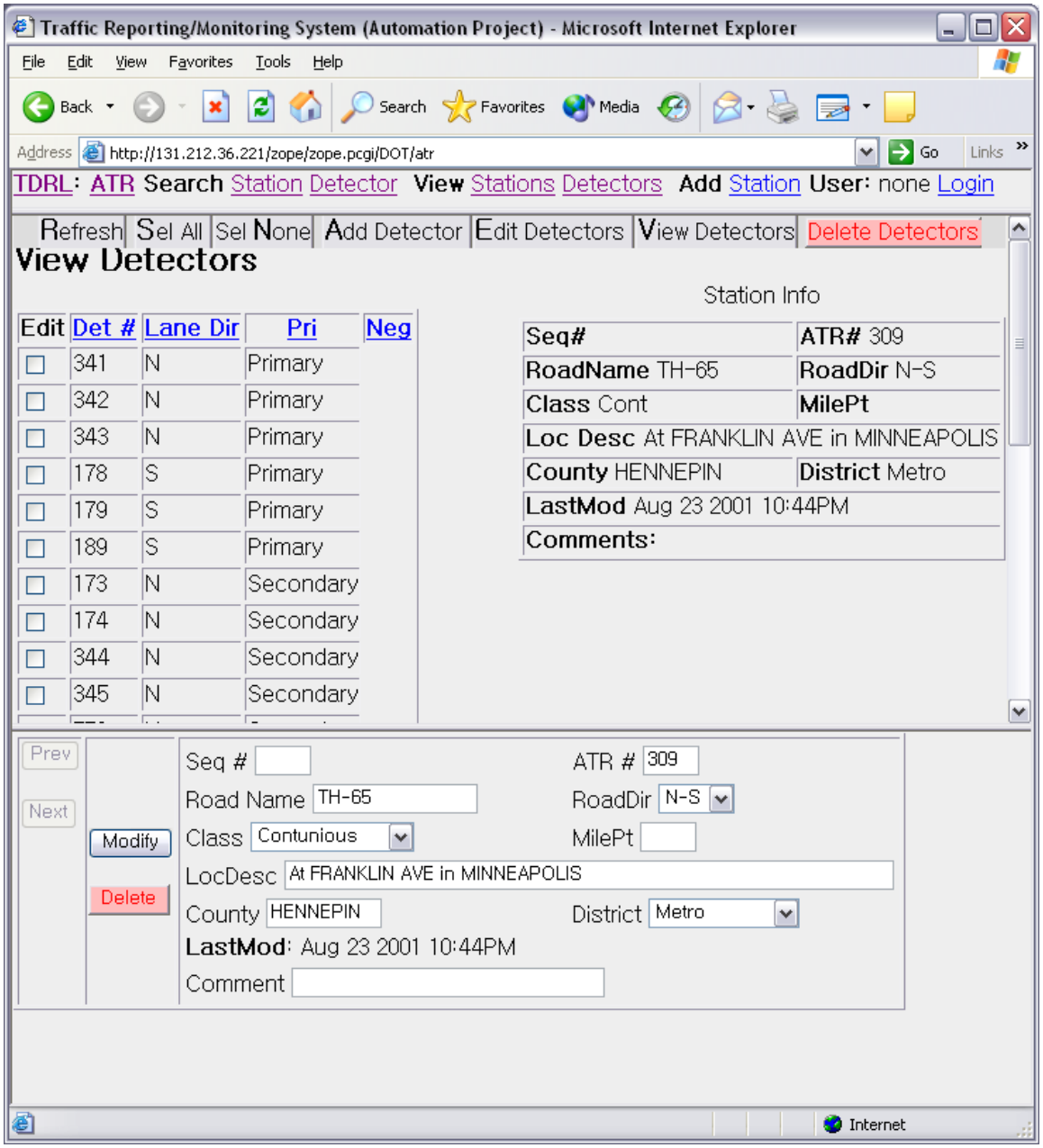


Figure 12: Sample screen capture of web interface: Selected station and detector view

Although this web application was developed with sophisticated web programming techniques and easy to use graphical user interfaces, in the final analysis it was concluded that it created additional burdens to the Mn/DOT users in training and learning and to the TDRL developers for user and database maintenance. This clearly goes against our initial spirit of creating simple and easy to maintain automated system. Therefore, for future version of this system, Mn/DOT and TDRL decided to develop a simple ftp based exchange of formatted text for the maintenance of station and detector information. This modification was planned to be carried out during the 2003-2004 fiscal year.

4.4.1.4 Data Format

The final output of continuous count data is formatted according to the existing SAS application input requirement for internal screening by Mn/DOT. All characters in the data file are ASCII characters. Each line contains a half-day of data for a station in one direction. Thus a full one-day amount of data in one direction occupies two lines: 12 hours per each row corresponding to AM and PM of the day. The field of each line and the digit positions are summarized in the table below. Considering two directions in most roads, a complete data for a single station per day would occupy four lines of data. This format was determined during the years that data were aggregated manually with the assistance of spreadsheets.

Table 3: Continuous Count Data Format

Digit Position	Number of Digits	Description
2	1	AM=1, PM=2
3-4	2	Month, 1-12
5-6	2	Day of the month, 1-31
7-8	2	Year
9	1	Day of the Week: Sun=1, Mon=2, Tue=3, Wed=4, Thu=5, Fri=6, Sat=7
10-12	3	Station ID*
13	1	Lane direction of the station, E,W,S,N,R
14-73	60	A set of five digits represents the hourly volume. Twelve of five digit sets (12*5=60) are consecutively concatenated in the order representing 1st to 12th hour depending on AM or PM.

* Presently, ATR ID is used as a Station ID.

Below two rows of data was taken from top two rows of a sample file."

```
210131002301E006620049800309002350027600897031060584005772040910388804217
220131002301E046780483805672069880712406576050020334802982033260217901497
210131002301W006310042600300003240058302301055300689606928050050441304565
220131002301W045650475705415058260664106847048970293602528023140184801073
```

As an example, the interpretation of the first line from the above data is illustrated in the following table:

Digit Position	Value	Meaning
2	1	AM
3-4	01	January
5-6	31	31st day

7-8	00	Year 2000
9	2	Day of the Week: Monday
10-12	301	Station ID = ATR ID
13	E	Lane Direction, East
14-73	00662 …	A set of five digits represents the hourly volume. Twelve four digits are consecutively listed

4.4.1.5 Log File Data Format

Along with the data file, a log file was produced to document missing data statistics and the choice of which detector set the algorithm selected. The log file consists of text readable ASCII codes, and they are mostly self-explanatory. The first line records the information on which day's data on what day it was processed, then it proceeds with station by station reporting of information on missing percent, directional volume differences, missing detector file, the hourly choice of detector sets and the hourly missing percent. The hourly choice of detector sets are denoted as: P=Primary, S=Secondary, and T=Tertiary. A sample log file can be found in Appendix B.

4.4.1.6 File Name Convention

Since the reported data are delivered to Mn/DOT electronically, a consistent file name convention was developed to denote which data type and the day of year it represents. For daily continuous count data and log files, a prefix "ATR" along with the date of the data is used as the file name.

File name format for daily ATR data: ATRyyyymmdd.dat

where yyyy denotes 4 digit year; mm denotes 2 digit month; and dd denotes 2 digit day. For example, for Feb 6, 2000, the data file should have the name "ATR20000206.dat" and the log file "ATR20000206.log."

To reduce the number of files, data sets are often packaged into a single file that may contain one or more weeks of data. The weekly data file name is denoted by appending a letter "w" followed by a numeric number that represents the number of weeks contained in that file. The date in the file name then represents the ending date of the data (mostly Sunday). A week is defined by seven days starting from Monday and ending after the final hour of Sunday. The following example further illustrates the name convention.

Example: ATR20020206w1.dat One week of data ending Feb 6, 2002.
ATR20020206w1.log Log file for ATR20020206w1.dat
ATR20020206w2.dat Two weeks of data ending Feb 6, 2002.
ATR20020206w2.log Log file for ATR20020206w2.dat

4.1.7 Software Developed

The software that computes the continuous count data was written in Microsoft Visual Basic with a few ActiveX tools. The code was relatively complex because it must handle unzipping, network file-transfer coordination, relational database access through network, missing file handling, calendar functions and the scheduled runs. However, the user interface is extremely simple as shown in Figure 13.

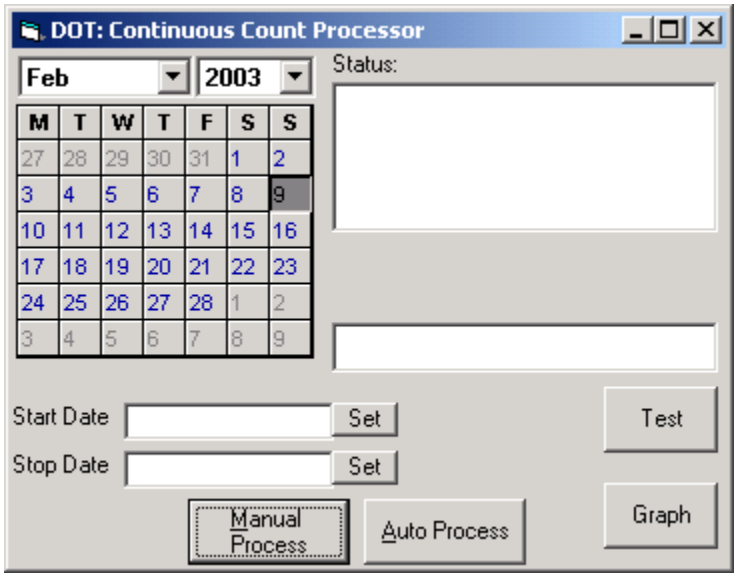


Figure 13 : Short duration count computation software

The code may run manually by entering the start and stop date or automatically by a scheduler. For manual entry, the dates should be entered by clicking the calendar buttons instead of typing to help eliminate typographical errors. However, a manual run should be used only if an error condition requires human intervention. During an automated run, the internal registry keeps track of which date was last completed so that it automatically determines which week to run next. It is presently scheduled to run daily to check whether the data from TMC arrived. If it finds enough data, then the computation process is activated.

An additional piece of utility software that can read and analyze the ATR data was developed since the ATR data in the form defined in Section 4.4.1.4 is hard to read. This software tool is named "ATRViewer" and includes the following functions:

- Reads multiple weeks of data
- Displays various forms of hourly graphs (line, bar, area, point)
- Calculates statistics
- Computes and plots daily volume
- Graphs hourly color grids and histogram
- Exports to an Excel file
- Converts and loads from binary source file

Figures 14-19 shows a sample screen of the functions listed above. Although the examples shown display only one week's worth of data, it can read an unlimited amount of data (such as a whole year) and can create the same plots and statistics.

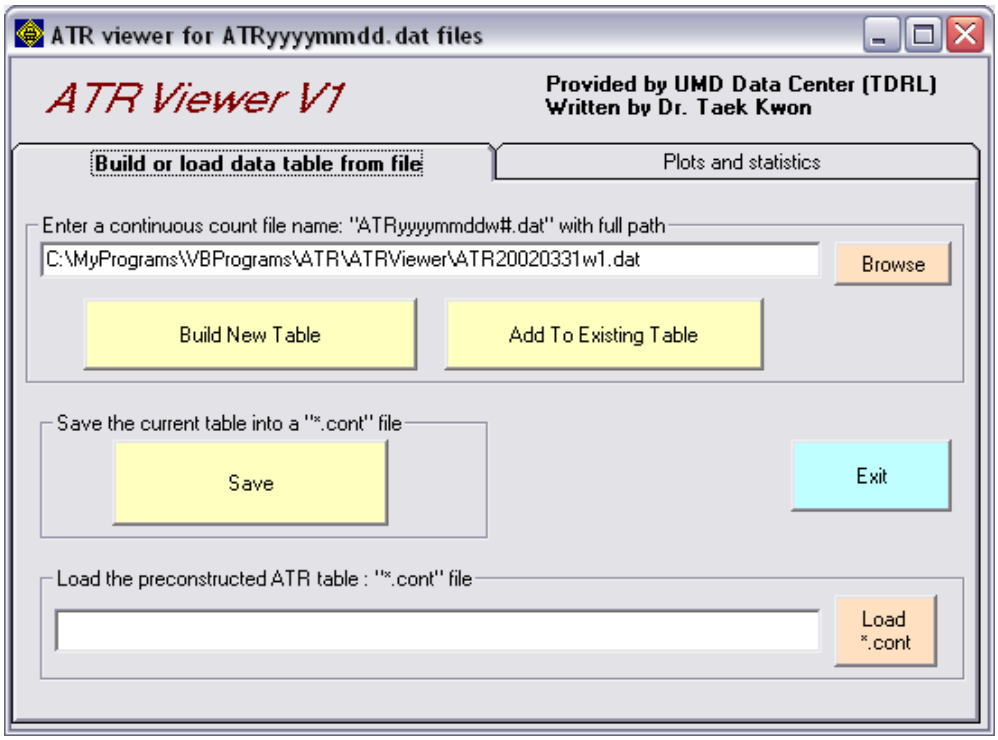


Figure 14: Screen Capture of the ATR Viewer Program

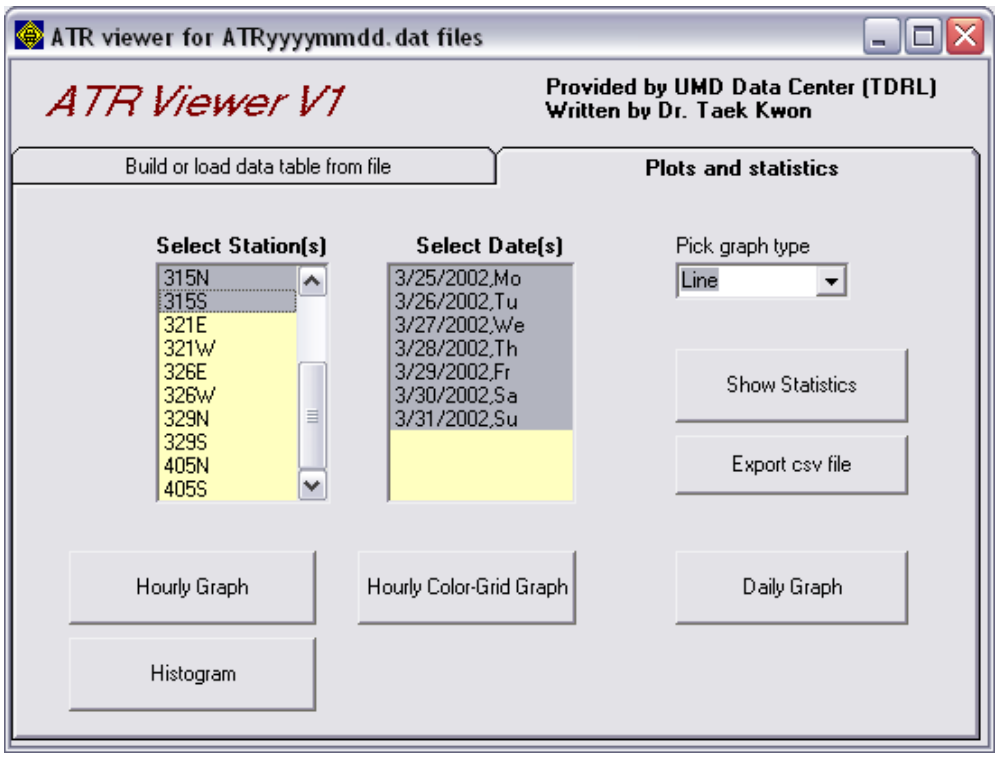


Figure 15: Plot and statistics tab

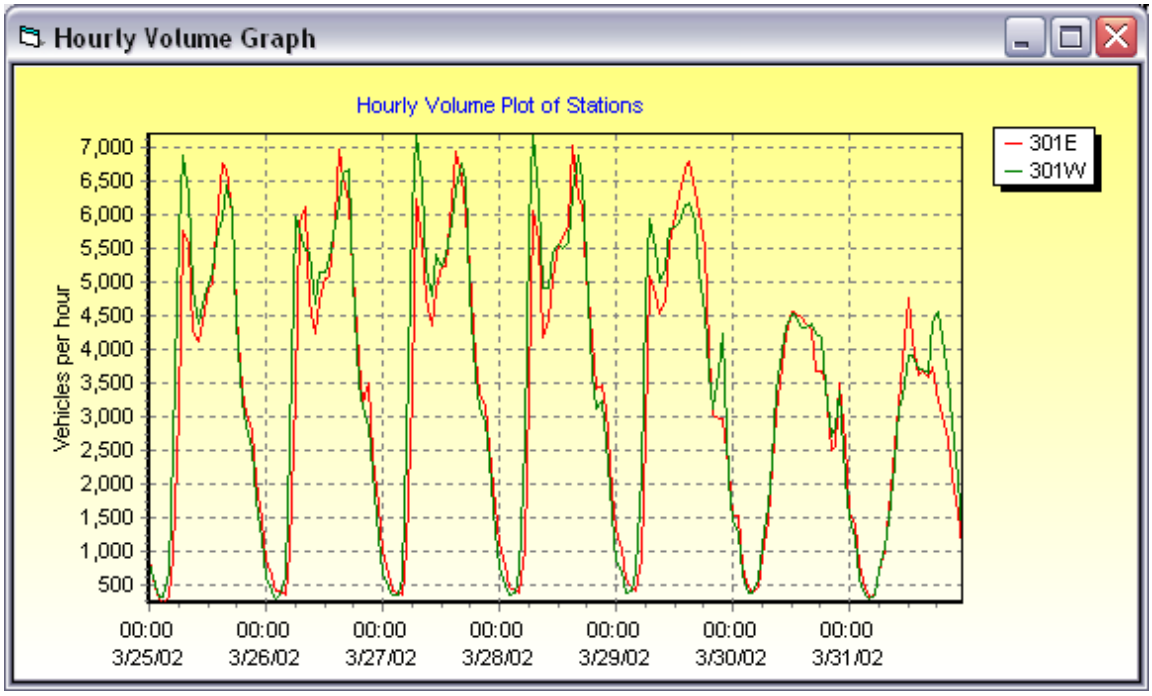


Figure 16: Line plot example

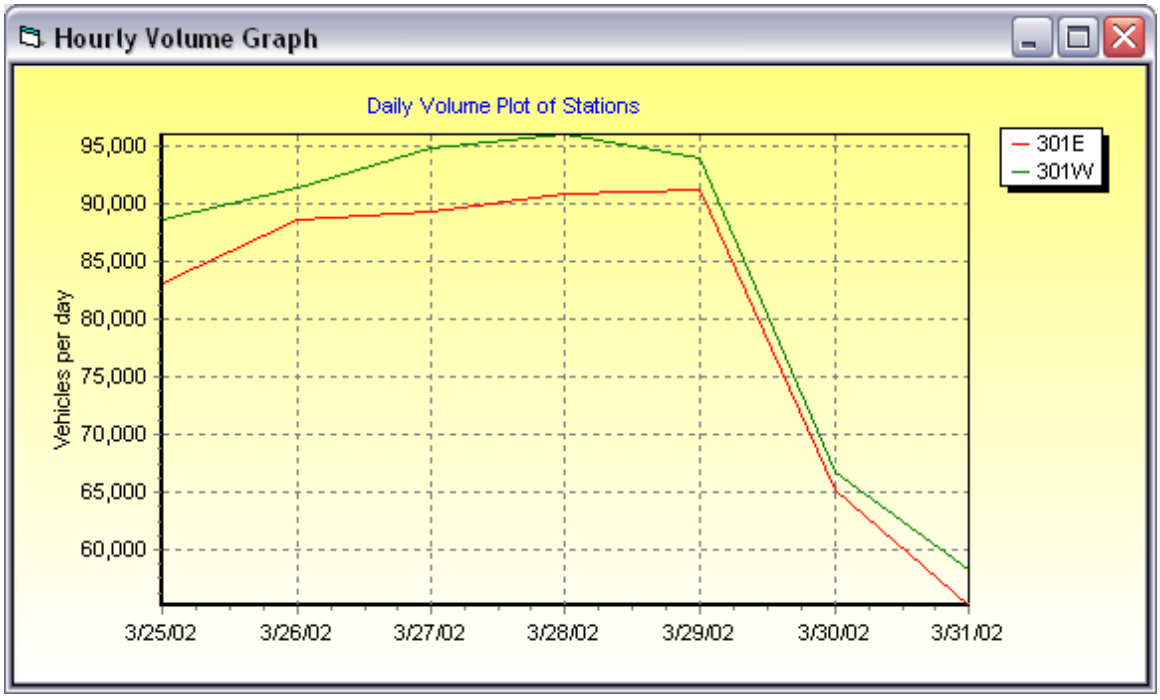


Figure 17: Daily volume plot example

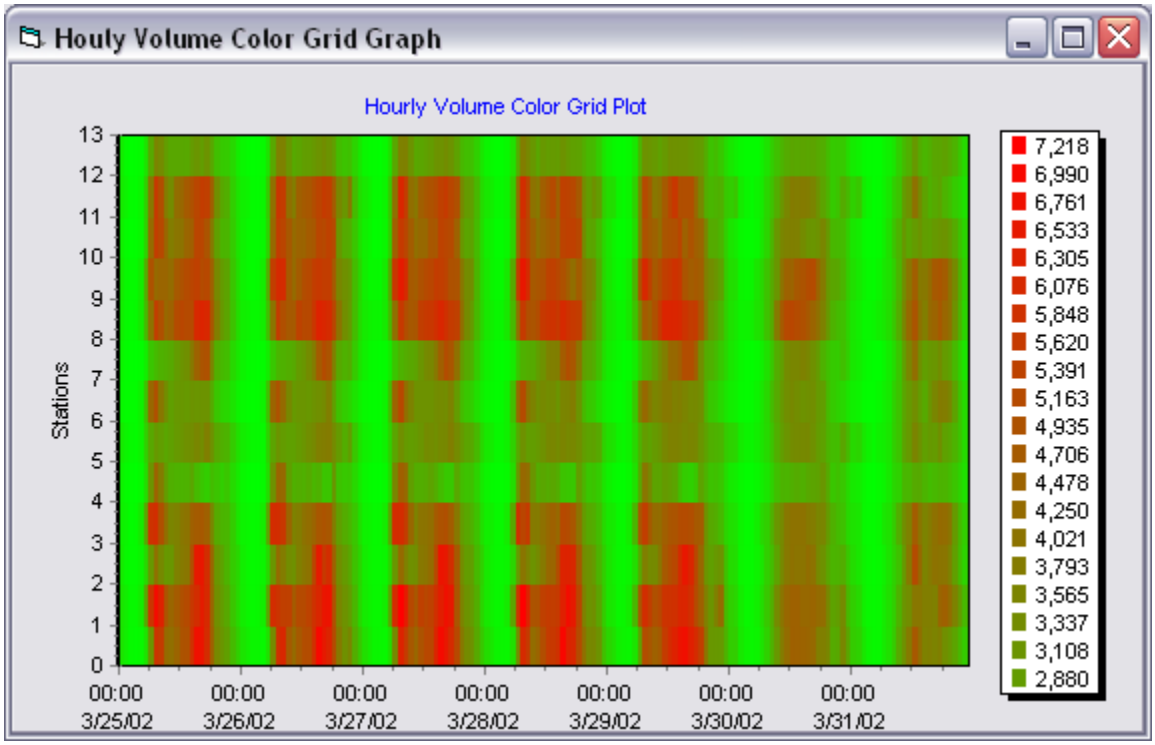


Figure 18: Hourly color grid example: Morning and afternoon high traffic times can be observed.

Station Stat

ATR ID	ADT	Min DT	Max DT	SD	AWDDT	AWEDT
301E	80,484	55,258	91,189	13,314	88,589	60,224
301W	84,247	58,276	95,983	14,121	92,951	62,488
303N	68,823	54,580	78,095	8,758	74,178	55,434
303S	70,277	55,538	79,172	9,322	76,045	55,858
309N	42,331	33,351	46,093	4,299	44,803	36,152
309S	53,911	45,230	58,041	4,309	56,347	47,822
315N	53,023	44,691	58,799	5,605	56,343	44,724
315S	52,755	43,002	59,490	5,807	56,153	44,258
321E	80,551	59,266	89,988	11,465	87,617	62,886
321W	78,900	58,872	88,036	10,919	85,489	62,427
326E	68,169	43,893	79,032	13,234	76,338	47,746
326W	69,722	41,923	81,150	14,645	78,680	47,328
329N	45,869	36,191	50,382	5,602	49,324	37,234
329S	49,767	39,555	54,674	5,541	53,151	41,306

Figure 19: Statistics of the selected period: ADT (average daily traffic), Min DT (minimum daily traffic), Max (maximum daily traffic), SD (standard deviation), AWDDT (average weekday daily traffic), AWEDT (average weekend daily traffic)

4.4.2 Short-Duration Count Data

This subsection describes the present version of AADT computation implemented for short-duration count stations.

4.4.2.1 Traditional Definition of Short-Duration Count In Mn/DOT

Short-duration count (SC) of a station is defined as a 24-hour (noon to noon) volume average computed over qualified three consecutive days, two 24 hour periods of noon to noon resulting 48 hours. In any given week, three qualified 48-hour periods are selected as from Monday noon to Wednesday noon (this period is denoted by the middle-date, Tuesday), from Tuesday noon to Thursday noon (middle-date=Wednesday), and from Wednesday noon to Friday noon (middle-date=Thursday). The qualified pool of dates for the short-duration count is typically selected from the period between *April 1 and November 1***.** During this period, dates with holidays, near holidays, detour, incidents, severe weather, and special events are excluded from the qualified pool of days to avoid any severe deviation from normal traffic patterns. These choices are made essentially to obtain a typical daily traffic count for a week day, from which the station's AADT can be estimated by seasonal/day-of-week adjustments. The adjustment factors for seasonal and day-of-week are derived from the ATR data on the same road or clustered ATRs exhibiting similar traffic characteristics as the SC station.

4.4.2.2 AADT Computation of SC Stations from ITS Traffic Data

Traffic data at a selected location is traditionally collected using portable vehicle counting devices such as pneumatic tubes by sampling typical days. On the other hand ITS traffic data is typically collected using pavement imbedded inductive loop detectors (not portable) at a much higher data-sampling rate (typically 30 seconds of samples) for real-time traffic monitoring and control. Also, the data is collected seven days a week, all year round. Therefore, it makes more sense to use the entire set of available data than to use a sampled set of just a few days as it was traditionally done for the computation of AADT using ITS traffic data. Based on this reasoning, the original task was modified to directly compute AADT from the TMC traffic data. Unfortunately, like other ITS traffic data, TMC data contains many missing values. If the amount of missing data is small, it does not present too much difficulty since they can be readily imputed. However, if the size of a missing data block is very large and thus only few good days are available for the entire year, imputation is more challenging.

4.4.2.3 New Station Definition Text Format

Although development and implementation of a relational database for managing detectors and stations (described in Section 4.4.1.3) has been completed, the final analysis indicates that the use of a simple text file is better for the personnel in Mn/DOT as discussed before. A simple text format for the SC station definition has been developed, which will be extended to the ATR stations in the future.

The line entry of the form has the following format:

StationID, DirCode, P, detP1, detP2, …, S, detS1,detS2,…, T, detT1,detT2, …, End

StationID is a unique identification number assigned to each station by Mn/DOT. DirCode denotes a direction code of a station that is determined by the direction of the road where the station is located. This code is represented by a number between 1 and 8 where the codes are defined in clockwise direction, i.e., 1=N, 2=NE, 3=E, 4=SE, 5=S, 6=SW, 7=W, 8=NW, 0= "All other directions such as reversible or both". The rest of fields can be more easily explained by an example. Let us define a station 321 direction 7 (west) with primary detectors at 843, 844 and 845, secondary detectors at 854, 855, 856, and 857, and tertiary detectors at 826, 827, 828 and 830. The line entry for this station would be written as:

```
321,7,P,843,844,845,S,854,855,856,857,T,826,827,828,830,End
```

Comments may be attached after the End statement or any line starting with the character ";" or the line left blank for legibility. The detailed editing rules and file name conventions are shown in Appendix C.

This format was developed to support a simple parser for programming purposes as well as for creating a human readable text format. Both objectives were accomplished and the present SC computing was implemented based on parsing of the detector lists written in this new format. The result was a significant performance increase against the database approach since the SC software no longer had to access the database to request the detector list each time it computed an AADT for a station. An ftp site has already been established from which Mn/DOT analysts can upload the station definition files. When the TDRL SC software is activated it runs using the most recent station definition file available from the ftp site.

4.4.2.4 Short-Duration Count Data Format

After computing the AADT for SC stations the program produces a final output following a certain format that can be directly fed into the Mn/DOT's TMS database. The format presently accepted by Mn/DOT is:

StationID, DirCode, EndingDate, AADT, "ValDays TMC"

The number of columns allocated for each field is:

7 columns, 2 columns, 11 columns, 7 columns, "Minimum 7 columns"

All fields are separated by a comma and are right justified. Null data is left as blank. The number of columns indicated for each data field includes spaces but excludes the separating comma. The meanings of the fields are:

- *StationID*: a unique sequence number defined for the station
- *DirCode*: direction code
- *EndingDate*: ending date of AADT computation period. Usually it is usually12/31/yyyy, but it can be also 10/31/yyyy, for example, if AADT was computed between 11/01/2001 – 10/31/2002, the ending date is 10/31/2002.
- *AADT*: Annual Average Daily Traffic (AADT) volume counts computed for one year ended by the *EndingDate*
- *ValDays*: the number of days that had useable and valid data for the AADT computing duration.
- *TMC*: It is a string constant that indicates "It was computed from the TMC data"
- "…": This field is a commenting area and is incorporated into the TMS database and shows up on analyst reports.

A sample data is shown in Figure 20. The resulting file name follows the format ADTSampleyyyy.txt where yyyy is the year of AADT.

```
 10069, 5, 06/26/2002,  41210, " 21 TMC"
 10182, 3, 10/31/2002,  37170, "350 TMC"
 10182, 7, 10/31/2002,  37095, "350 TMC"
 10286, 1, 10/31/2002,  44310, "322 TMC"
 10286, 5, 10/31/2002,  45375, "315 TMC"
 10287, 1, 10/31/2002,  41600, "329 TMC"
 10287, 5, 10/31/2002,  43332, "161 TMC"
 10288, 3, 10/31/2002,  33969, "294 TMC"
 10288, 7, 10/31/2002,  35226, "294 TMC"
```

Figure 20 : Sample AADT data formatted according to Mn/DOT specification

4.4.2.5 Detection of Missing and Incorrect Volume Counts

Before the imputation algorithm is implemented, the first step required is identification of missing and incorrect values. These missing data and incorrect values become candidates for imputation.

When a TMC traffic file is unzipped, it produces daily volume and occupancy files, each of which contains 2,880 values representing 30-sec samples of a single detector for a single day. In the data, all hardware errors are already flagged as a negative value during the data packaging process. These negative values become missing values in our algorithm. In addition, any volume counts greater than 39 per 30-second period are considered as incorrect values and are treated as missing values since such values are physically impossible. Yet another type of values screened are consecutive repeating values. In traffic data, there is a high probability of repeating 0 or 1 (or low number) during the low traffic hours such as 2:00 – 5:00 AM. However, the repeating is less likely to appear during the high traffic hours. Repeating of high numbers such as a number greater than 10 is highly unlikely to appear during any time of the day. In general, the probability that repeated numbers appears in a daily detector file diminishes as the volume count becomes larger.

Based on this principle, we can construct a probability model for the detection of incorrect data. Theoretically, its distribution should follow a Poisson distribution. However, it was not clearly observed in the real data. A simple but practical rule of detecting repeated values was established as follows. Repeated zeros or ones are considered normal during the low volume hours 2:00 – 5:00 AM. During any other period, if repeated values are observed more than four hours, it is considered as incorrect data and replaced with imputed values.

In addition to the repeating value problem, there are other types of incorrect count values that exist in loop data. When the threshold of loop detector sensitivity is set to a

wrong value, volume counts can be too high or too low. Very often mutual coupling causes over counting due to detection of adjacent lanes. In general, undercount or over-count problems are extremely difficult to detect just from the loop data alone. In this project, no attempts have been made to detect or correct over- or under- count problems in loops.

4.4.2.6 Implementation of Imputation

The basic premise of the overall imputation algorithm developed in this project was that missing data patterns (types classified in Section 4.3.2.2) supply recursive inferences to the next level as imputation moves from the Type-A missing data patterns and progress toward the Type-F missing data patterns. For example, after imputation of Type-A and Type-B missing data patterns there will be less Type-C missing data patterns, therefore more inferences are available for imputing Type-C missing data patterns, which would result in imputation with more information. The overall data processing cycle is implemented beginning with the proper identification of missing data types and then applying corresponding imputation algorithms. Figure 21 illustrates the steps implemented through a block diagram.

The imputation process starts with treating the detector-level random missing data cases, i.e., Type-A missing data patterns as shown in Figure 21. Since Type-A patterns are a class of random missing data patterns, the NBLR algorithm described in Section 4.3.4.2 was used for imputation. After extensive experiments, it was determined that up to 16 consecutive missing values of 30-second data can be effectively imputed using the NBLR algorithm. In the overall processing, Type B missing data patterns were not imputed since they are eventually imputed during the process of Type-C and D patterns.

After imputation of Type-A missing data patterns, the detector data was converted into station data with 5- minute interval. This was necessary to create a smaller memory requirement, so that a whole year of data could be loaded into the computer RAM and processed. Without this conversion, about 10 GB (giga bytes) of data must be loaded into RAM to process one year of data. Such a large memory-capacity is not presently available from the computers at the TDRL data center. Type-C missing data patterns were determined by less than six consecutive missing data points, which would correspond to 30 minutes. However, for future implementations 12 consecutive missing data points that correspond to one hour is recommended for Type-C missing patterns since 5 minute data can easily infer the time trend up to one hour. Imputation of Type-C patterns was implemented using the NBLR algorithm since Type-C patterns are random missing data patterns at the station level.

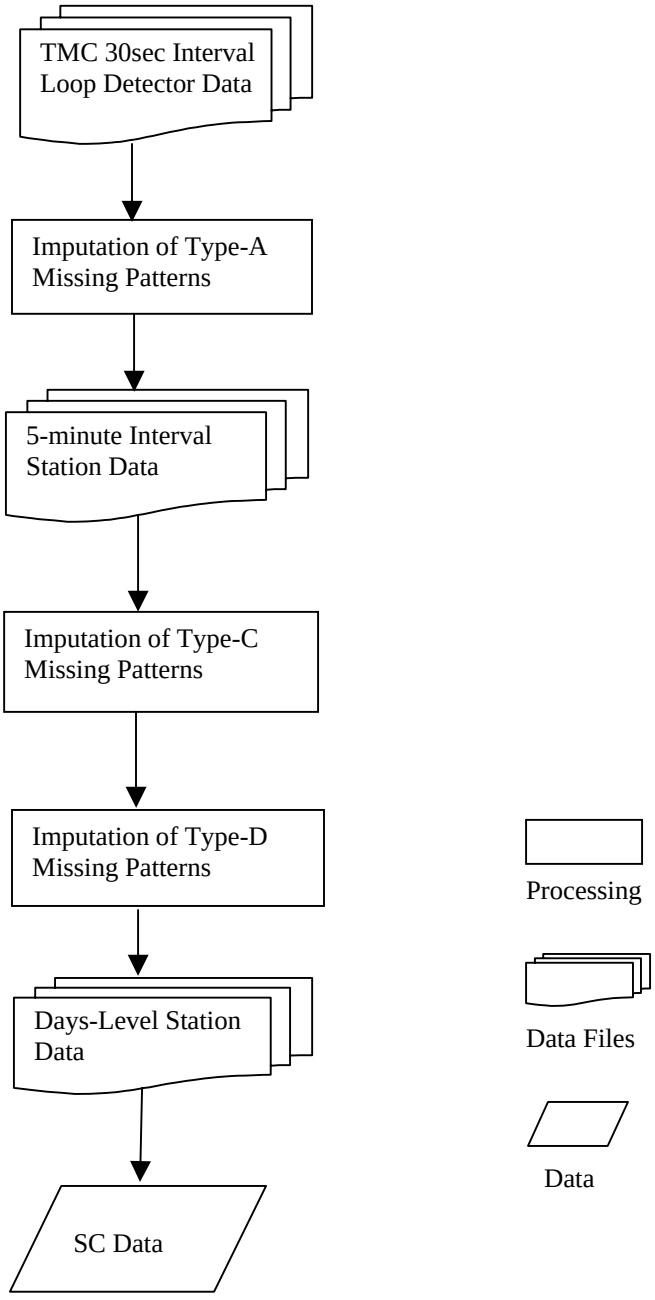


Figure 21: Block Diagram of Imputation Steps Implemented

Upon completion of Type-C imputation, block level imputations are applied to Type-D missing data patterns. Type-D missing data patterns were determined when the size of missing block is less than 60% of the day. The Algorithm 3 (Block Level Nonormal Bayesian Imputation) discussed in Section 4.3.4.3 was used for imputing the Type-D missing data patterns.

As shown in Figure 21, after completion of Type-D imputation day-level station data was produced for several purposes: for the final computation of AADT, days-level imputations, and also to provide a type of data similar to the ATR stations. In the present state of implementation (at the time of this writing), imputations up to Type-D missing data patterns were completed, but imputation of Type-E and F missing data patterns have not been implemented and left for future study due to limited time and the need for further studies.

4.4.2.7 Software Developed

The imputation software written implements the processing steps of the block diagram shown in Figure 21. This software does not have any interesting user interface since it was written for scheduled runs without manual commands or user interventions. However, an important utility tool that allows analysis of days-level data for any selected year has been developed for Mn/DOT analysts and is described in this section.

The days-level station data in Figure 21 contains information on daily traffic between the start date and end date of the year defined in a self-descriptive form. This effort was made to make the data file transportable between different operating systems and computer systems. The days-level station data is organized in the following order:

(1) a "magic code" used for software confirmation and identification of the days-level data type; the code is 56789yyyy where yyyy is the year of the ending date, e.g., for 2003 the magic code is stored as 567892003. (4 byte long integer)

(2) starting date of the year (8 byte date type)

(3) ending date of the year (8 byte date type)

(4) number of stations (2 byte integer)

(5) number of days (2 byte integer)

(6) an array of station numbers (4 byte array)

(7) direction code for the stations (2 byte integer array)

(8) two dimensional long (4 bytes) array of daily traffic count of the year; an array element x(i,j) represents total volume of ith station on jth day.

This days-level station data file is produced for every year as the SC automation program finishes the run. A tool utilizing this data named "Daily Traffic Data Analyzer", has been developed and is illustrated in Figure 22. Upon loading of data using this software tool, it immediately computes AADT (Average Annual Daily Traffic), AWDDT (Average Weekday Daily Traffic), AWEDT (Average Weekend Daily Traffic), PDT (Peak Daily Traffic of the Year), ValDays (number of valid days), SDs (Standard Deviations), and number of outliers. Users can select or double click on any of the stations available to see the graphs of daily traffic for the entire year. An example graph is shown in Figure 23. The graphs can be zoomed in or out to see the details by dragging a mouse on the region of interest. A zoomed example is shown in Figure 24. Using this graph, analysts can see how the traffic has been changed during the year or what the trends are. For example, one can see that Mondays have least traffic while Fridays have

the most traffic during weekdays. Another trend that can be noticed is that, during the summer months, traffic increases, but in December traffic is significantly decreased.

Using this tool, the users can also see the actual data by clicking the left axis and by selecting the data tab. The data screen example is shown in Figure 25. In addition, there are a number of graphical editing tools that are available for users which are not described here but can be easily learned by playing with the software. According to Mn/DOT analyst's comments, "This tool was an invaluable tool" for their traffic analysis. This tool is presently available for download from the TDRL web site (http://tdrl1.d.umn.edu). In the future, a Geographical Information System (GIS) tool that implements a traffic map will be developed to provide spatial map of daily traffic counts. This GIS application will be available as a web tool, such that the users can easily see the traffic estimates.

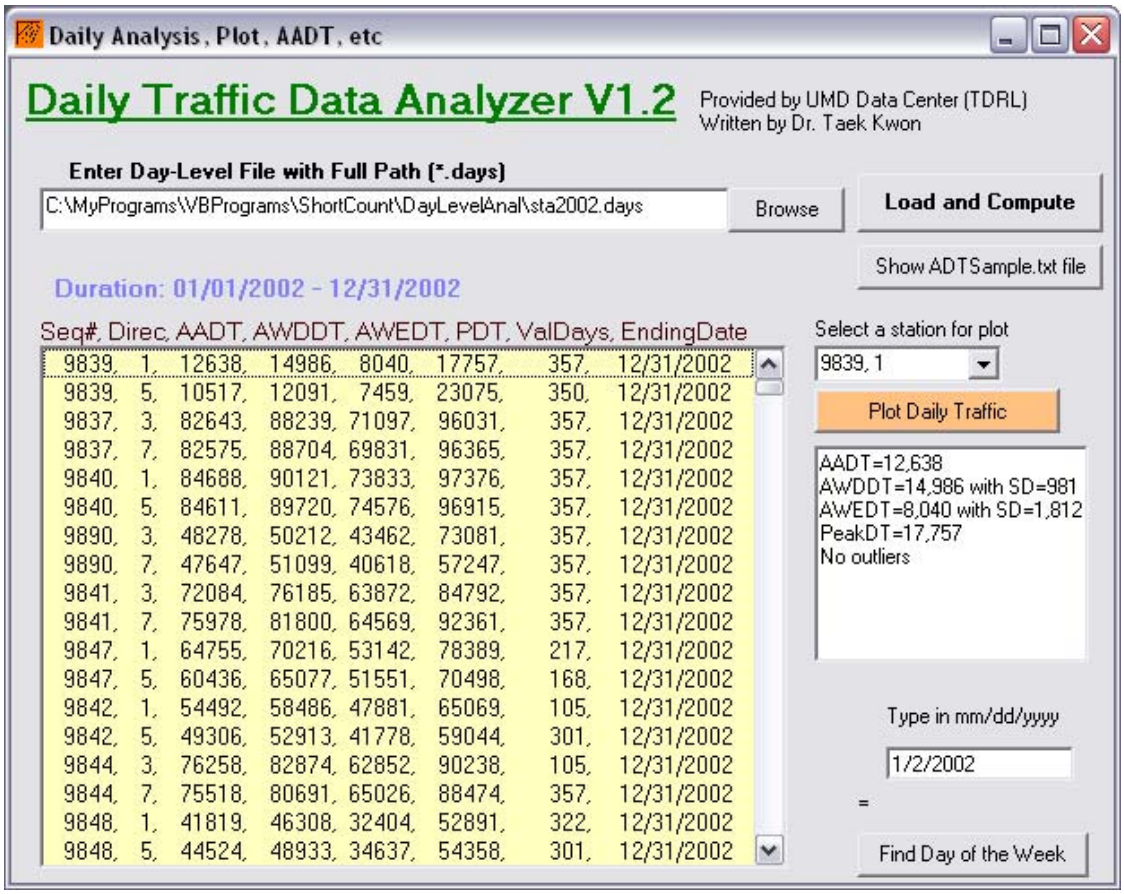


Figure 22: A sample screen of Daily Traffic Data Analyzer

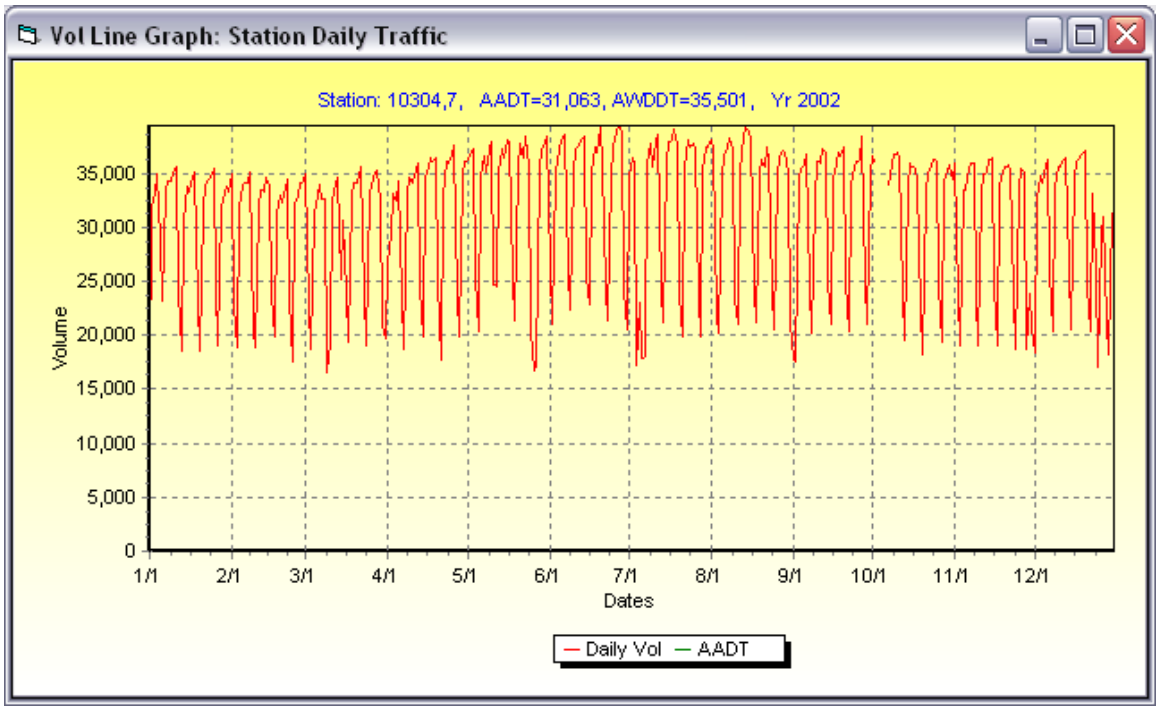


Figure 23: A sample graph of daily traffic, station 10304, NE, year 2002.

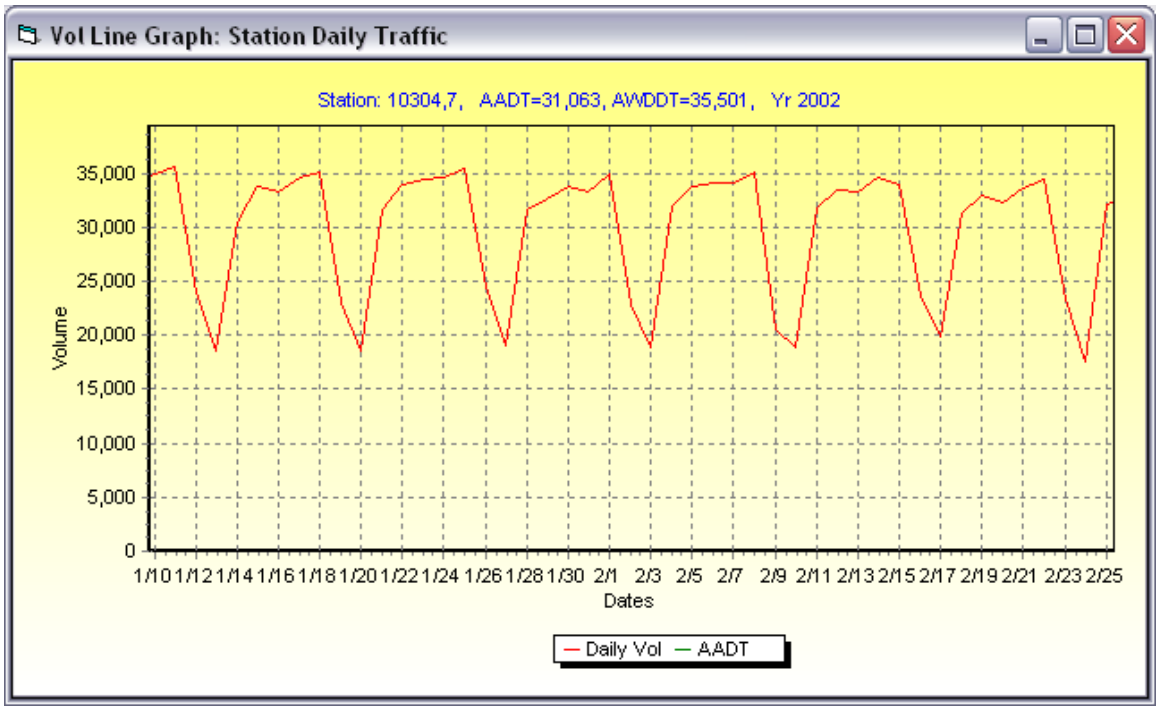


Figure 24: A zoomed in graph of Figure 19 by dragging a mouse on the region of interest.

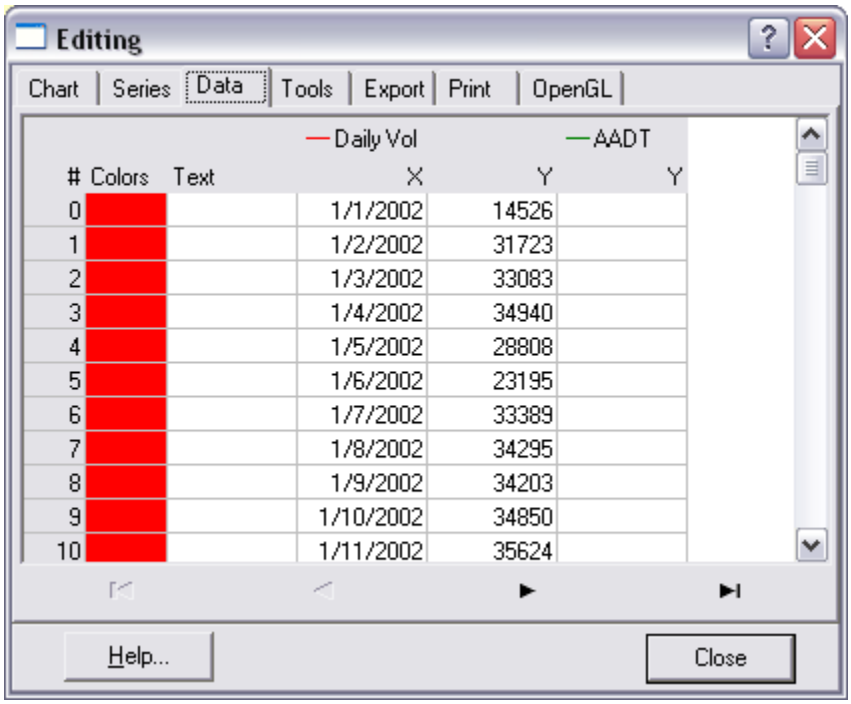


Figure 25: Graph editing tool that allows to see the actual data as well as various editing functions of the graph.

4.5 Conclusion and Future Work

This project was started with a goal of automating the continuous and short-duration count data for the portion of TMC traffic data. In the process, one important issue which the research team spent the most amount of time on was how to deal with the missing and incorrect data that exist in the TMC traffic data. First, all identifiable incorrect data points were simply treated as missing data since we do not know the amount of incorrectness. Thus the problem was simplified to dealing with only missing data. The finding is that missing data can be effectively imputed with the values that are very close to the real values if we utilize observable spatial and temporal relations of traffic flow. For utilization of spatial relation, this project introduced multiple redundant sets of detectors that are defined (or allocated) for each station by locating the detector sets that have an equivalency relation in terms of traffic flow. Since this approach is essentially equivalent to creating redundancy in data through availability of additional data from the vicinity of the primary detectors, it enabled replacement of missing data from the equivalent set of data. It should be noted that the use of spatial redundancy was possible because the TMC traffic sensor network was densely implemented (loops were installed at every 0.5 miles), which is one of the advantages of using ITS generated traffic data for traffic counting program. This strategy significantly improved the data quality by reducing the number of missing data when three sets of detectors (primary, secondary and tertiary) were assigned.

However, this approach alone did not produce the desired “completely healthy” traffic data that do not contain any missing data points. In some cases, all three sets of detectors contained missing data within the same time span. Therefore, there is a need for additional means to treat the missing data. Another important relation of traffic flow that was utilized was temporal relation of traffic flow. It was found that, except for the days with special events or holidays and near holidays, traffic patterns tend to repeat during the same day of week. Since this repetition does not occur precisely but in a statistical sense, algorithms that utilize Bayesian approach of quantifying uncertainty in temporal inferences (trends in this case) based on statistical properties of the data were developed. These algorithms successfully imputed the missing values when temporal inferences are available. In summary, the problems of missing data in ITS generated traffic data could be overcome through imputation based on temporal and spatial inferences that exist in the traffic flow.

Another finding from this project was that conventional way of using 48 hours or 72 hours of representative samples as a short-duration count and then adjusting it for AADT is no longer necessary for ITS generated traffic data. After spatial and temporal imputation of data, an ample amount of data was available for directly computing AADT and other summary statistics of traffic volume. Thus, the initial project tasks were modified to directly compute AADT rather than selecting short-duration samples.

There are some outstanding issues that require further exploration. One of them is estimating AADT when stations have absolutely no data for the entire year. It occurs in about 10 to 15 stations out of 483 stations every year. Since such stations do not have any data for imputation based on temporal inference, spatial inference such as the locations having equivalent traffic demand is the only available information that can be incorporated. Future work should study how spatial inferences in such cases can be

automatically incorporated to come up with a reasonable estimate of AADT. Having no data for the entire year suggests that the loops, controller, or communication links have been failing for the entire year, which could have been prevented through proper maintenance. Therefore, there is a need for developing an automatic notification system that reports suspected loop failures to Mn/DOT maintenance personnel or finding other ways of reducing long term failures.

Through this project, the research team learned that an automated system on this scale must be an evolving system. Better concepts and methods can be tested as the system is being developed, implemented, and used, as this was demonstrated in this project. That is, many initial concepts and methods in the proposal evolved into improved concepts and methods. It is expected that the three parties of the project team, TDA, TMC, and UMD Data Center, will continue to work together on improving the present system.

Appendix A: Station Identification Database

Station Table

Column Name	**Description**
StaDBID	DB generated index
SeqNum	Mn/DOT defined sequence number
ATRNum	Mn/DOT defined ATR number
RoadName	Road name that the station belongs to
RoadDir	Road direction at the station
Class	Continuous or Short-Duration
MilePoint	Station mile point
LocDesc	Location description generally used
County	County name that the station belongs to
District	District name that the station belongs to
LasdMod	Last modified date and time
LastUser	The user who last modified
Comments	Any comments on the station

Detector Tabl

Column	Description
StaDBID	Index of station table

DetDBID	Index of detector generated by database
DetNum	TMC assigned detector number
Priority	Primary, secondary, or tertiary
Negative	Negative signed detector (volume is subtracted)
LaneDir	Lane direction of the detector

Appendix B: Sample Log Data for Continuous Count Data

```
____----**** Processing 12/16/2002 ****----____  12/26/2002 12:50:43 PM
Processing: ATR 301 - 12/16/2002 - Seq 0
 ---Checking Primary---
  Data Missing:  E .03%   W .03%
  Direction Volume Difference =  8.9%
 ---Checking Secondary---
  Data Missing:  E .02%   W .03%
  Direction Volume Difference =  6.2%
 ---Checking Tertiary---
  Data Missing:  E .02%   W .03%
  Direction Volume Difference =  9.8%
 Selected det set:
 E   P   P   P   P   S   P   P   P   P   P   P   P
     P   P   P   P   P   P   P   P   P   P   P   P
 W   P   P   P   P   S   P   P   P   P   P   P   P
     P   P   P   P   P   P   P   P   P   P   P   P
 Hourly Missing %:
 E   0   0   0   0 .50   0   0   0   0   0   0   0
     0   0   0   0   0   0   0   0   0   0   0   0
 W   0   0   0   0 .67   0   0   0   0   0   0   0
     0   0   0   0   0   0   0   0   0   0   0   0

Processing: ATR 303 - 12/16/2002 - Seq 0
 ---Checking Primary---
  Data Missing:  N .03%   S .03%
  Direction Volume Difference =  3.8%
 ---Checking Secondary---
```

```
 Data Missing:  N .03%   S .03%
 Direction Volume Difference =  2.8%
---Checking Tertiary---
 Data Missing:  N  17%   S  22%
 Direction Volume Difference =  .36%
Selected det set:
N   P   P   P   P   S   P   P   P   P   P   P   P
    P   P   P   P   P   P   P   P   P   P   P   P
S   P   P   P   P   P   P   P   P   P   P   P   P
    P   P   P   P   P   P   P   P   P   P   P   P
Hourly Missing %:
N   0   0   0   0 .63   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0
S   0   0   0   0 .63   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0

Processing: ATR 309 - 12/16/2002 - Seq 0
---Checking Primary---
 Data Missing:  N .01%   S .01%
 Direction Volume Difference =   18%
---Checking Secondary---
 *unZip_1Det: Detector Not Found! D:\DOT\Traffic\20021216.traffic
95.v30
 Data Missing:  N .02%   S  25%
 Direction Volume Difference =   65%
---Checking Tertiary---
 Data Missing:  N  31%   S  31%
 Direction Volume Difference =  1.3%
Selected det set:
N   P   P   P   P   P   P   P   P   P   P   P   P
    P   P   P   P   P   P   P   P   P   P   P   P
S   P   P   P   P   P   P   P   P   P   P   P   P
    P   P   P   P   P   P   P   P   P   P   P   P
Hourly Missing %:
N   0   0   0   0 .28   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0
S   0   0   0   0 .28   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0
```

```
Processing: ATR 315 - 12/16/2002 - Seq 0
 ---Checking Primary---
  Data Missing:  N .03%   S .03%
  Direction Volume Difference =  1.6%
 ---Checking Secondary---
  Data Missing:  N .01%   S .01%
  Direction Volume Difference =  1.6%
 ---Checking Tertiary---
  Data Missing:  N .03%   S .03%
  Direction Volume Difference =  1
 Selected det set:
 N   P   P   P   P   S   P   P   P   P   P   P   P
     P   P   P   P   P   P   P   P   P   P   P   P
 S   P   P   P   P   S   P   P   P   P   P   P   P
     P   P   P   P   P   P   P   P   P   P   P   P
 Hourly Missing %:
 N   0   0   0   0 .21   0   0   0   0   0   0   0
     0   0   0   0   0   0   0   0   0   0   0   0
 S   0   0   0   0 .28   0   0   0   0   0   0   0
     0   0   0   0   0   0   0   0   0   0   0   0

Processing: ATR 321 - 12/16/2002 - Seq 0
 ---Checking Primary---
  *unZip_1Det: Detector Not Found! D:\DOT\Traffic\20021216.traffic
850.v30
  Data Missing:  E  33%   W .02%
  Direction Volume Difference =   100%
 ---Checking Secondary---
  *unZip_1Det: Detector Not Found! D:\DOT\Traffic\20021216.traffic
854.v30
  *unZip_1Det: Detector Not Found! D:\DOT\Traffic\20021216.traffic
855.v30
  *unZip_1Det: Detector Not Found! D:\DOT\Traffic\20021216.traffic
856.v30
  *unZip_1Det: Detector Not Found! D:\DOT\Traffic\20021216.traffic
857.v30
  Data Missing:  E .02%   W  100%
```

B-3

```
  Direction Volume Difference =  100%
 ---Checking Tertiary---
  *unZip_1Det: Detector Not Found! D:\DOT\Traffic\20021216.traffic
849.v30
  *unZip_1Det: Detector Not Found! D:\DOT\Traffic\20021216.traffic
850.v30
  *unZip_1Det: Detector Not Found! D:\DOT\Traffic\20021216.traffic
851.v30
  *unZip_1Det: Detector Not Found! D:\DOT\Traffic\20021216.traffic
853.v30
  Data Missing:  E  100%   W .01%
  Direction Volume Difference =  100%
 Selected det set:
 E   S   S   S   S   S   S   S   S   S   S   S   S
     S   S   S   S   S   S   S   S   S   S   S   S
 W   P   P   P   P   T   P   P   P   P   P   P   P
     P   P   P   P   P   P   P   P   P   P   P   P
 Hourly Missing %:
 E   0   0   0   0 .42   0   0   0   0   0   0   0
     0   0   0   0   0   0   0   0   0   0   0   0
 W   0   0   0   0 .21   0   0   0   0   0   0   0
     0   0   0   0   0   0   0   0   0   0   0   0
```

Appendix C: Format for Station Detector List Files

The detector list files for both the Short-duration Count (will be referred to as SC) stations and Continuous Count (will be referred to as ATR) stations must follow strict rule in order to allow the files to be used by the intended software.

Notation Convention:

yyyy	four digit number representing year, 0000-9999
mm	two digit number representing month, 01-12
dd	two digit number representing date, 01-31

File name:

For ATR stations, ATRDetsyyyymmdd.txt

Example) ATRDets20020101.txt

For SC stations, SCDetsyyyymmdd.txt

Example) SCDets20030109.txt

It is important to exactly use 8 digits for the year, month, and date along with the prefix as specified by above. The file name must not include any spaces. **This date should correspond to the date you are uploading the file to the TDRL server.** The reason for including the date information in the file is to keep track of changes may occur over time and to allow to go back and be able to rerun the program using the old detector lists. More importantly, the date information allows the developed software to recognize the most recent version of the detector lists and use them.

Primary, Secondary, and Tertiary Detector Sets

- For ATR stations, **all three prioritizes sets of detectors must be listed.**
- For SC stations, **only the primary set of detectors are required**, and the secondary and tertiary detector sets may be optionally added.

This and file names are the only difference between the ATR and SC stations.

Direction Code

The road direction at a station is numerically coded as follows:

- 1=N
- 2=NE
- 3=E
- 4=SE
- 5=S
- 6=SW
- 7=W
- 8=NW
- 0=All others such as reversible directions

Rules for the Entry of Detector List

- Any line starting with semicolon ";" is considered as a comment line and ignored by the software. The semicolon does not have to start from the first column. Comment area is especially useful for indicating the detector states and changes.
- Blank lines are considered as a comment line and ignored by the software.
- The detector list for a single station must be specified within a single line no matter how long the line is.
- The data entry line must start with the numeric station ID number and end with the "End" statement.
- All entries are **not case sensitive**.
- Any entries appended after the "End" statement is considered as a comment and ignored by the software.
- Each entry within the line must be separated by comma except for the comments.
- Spaces are allows after the commas, but not allowed between the numeric numbers. For example, "`,  24535`" is OK, but "`,  24  535`" will cause an error.

The Line Entry Format:

The primary detectors are listed after the letter “P”; the secondary detectors are listed after the letter “S”; and the tertiary detectors are listed after the letter “T”. The line format is:
StationID, DirCode, P, detP1, detP2, …, S, detS1,detS2,…, T, detT1,detT2, …, End
where

StationID: numeric number of the station ID
DirCode: numeric number representing the direction
P, S, T: indicates start of list of primary, secondary, and tertiary detectors
detP1, detP2, …: list of numeric numbers representing primary detectors
detS1,detS2,…: list of numeric numbers representing secondary detectors
detT1,detT2, …: list of numeric numbers representing tertiary detectors
End: indicates the end of the detector list for the station

Example Detector List File:

```
; Comments start with “;”
; Detector list for ATR stations
301,3,P,3176,3177,3178,3179,S,2638,2639,2640,2641,2642,T,2643,2644,2645,2646,3180,End
301,7,P,3218,3219,3220,3221,S,2658,3222,3223,3224,3225,T,2663,2664,2665,2666,3217,End
303,1,P,2393,2394,2395,2396,S,2397,2398,2399,2400,T,2389,2390,2391,-2392,2396,End
303,5,P,2457,2458,2459,2460,S,2450,2451,2452,2456,T,-2461,2462,2463,2464,End
309,1,P,341,342,343,S,173,174,344,345,570,T,335,336,337,338,-339,-340,End
309,5,P,178,179,189,S,94,95,176,177,T,-181,-182,183,184,185,186,End
315,1,P,494,495,496,S,266,267,525,1038,T,156,268,269,270,533,End
315,5,P,256,257,1003,S,1006,1007,1008,T,110,254,255,1002,End
321,3,P,846,847,850,S,837,838,839,841,T,849,850,851,853,End
321,7,P,843,844,845,S,854,855,856,857,T,826,827,828,830,End
326,3,P,793,794,795,S,1730,1731,1732,T,783,784,785,786,End   Comments can be here

; blank lines are allowed. Blank lines may improve readability.
326,7,P,788,789,790,S,1740,1741,1742,T,777,778,779,787,End
; comments can be added between the lines
329,1,P,339,340,S,-2130,2131,2132,T,335,336,337,338,-341,-342,-343,End
329,5,P,181,182,S,2243,2244,T,-178,-179,-180,183,184,185,186,End
405,1,P,1926,1927,1928,S,1929,1930,1931,T,1922,1923,1924,-1925,1928,End
405,5,P,1970,1971,S,1972,1973,1974,T,1972,-1975,1976,1977,1978,End
; comments can be added at the end as well.
; all of the above are valid format
```

CHAPTER 5: WEIGH-IN-MOTION PROBE

5. 1 Introduction

The WIM Probe was developed as a diagnostic tool for probing and analyzing WIM systems designed based on Kistler LINEAS sensors (Type 9195C1/C2). Using the WIM Probe, the user can directly sample the raw WIM signal coming from the BNC connectors of LINEAS sensors and analyze the noise level of the signal to determine the sensor status. The system supports two channels that can be used to test two parallel sets of LINEAS installation (one lane). The analysis software uses the sampled data and shows the weight computation processes of the raw signal. Each step of these computations can be compared with the field WIM system to confirm or determine possible cause of the problems. The system includes a notebook computer, data acquisition hardware, and software. The overall system was designed rugged for its use as a portable tester.

5.2 Hardware setup

The WIM Probe comprises of a notebook PC, interface box, and connections from LINEAS sensors as shown in Figure 26.

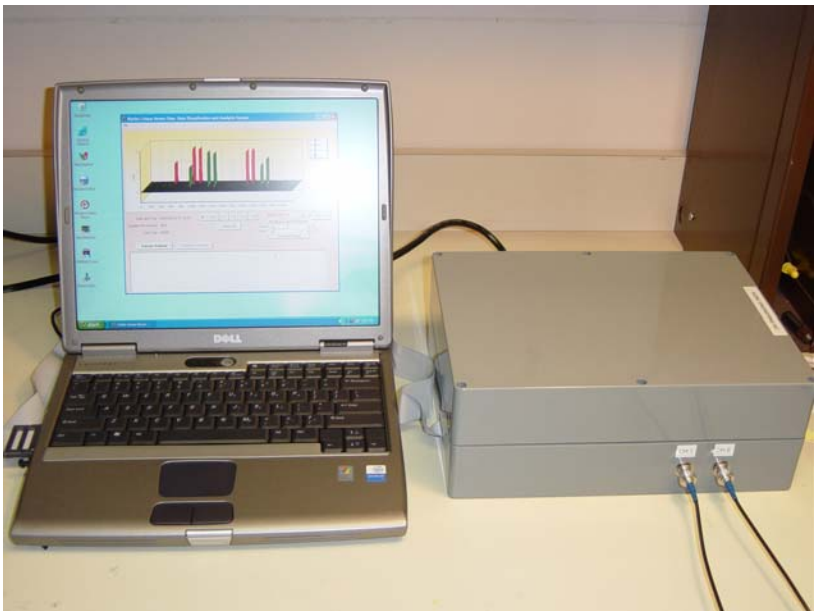


Figure 26: WIMDaqAnal Hardware setup

The user should follow a proper power-on procedure as described below

- Insert the PC-Card DAS16/16 to a PCMCIA slot of the notebook PC.

- Connect the edge connector of the ribbon cable to the DAS16/16 card (The white dot of the edge connector should face up).
- Turn on the notebook PC.
- Connect the power of the interface box.
- Connect the BNC connectors from LINEAS sensors to CH0 from the first set of sensors and to CH1 from the next set of sensors that vehicles cross (See Figure 27).

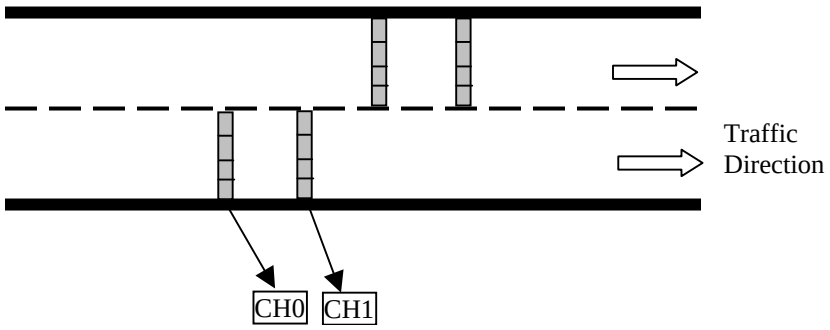


Figure 27: Connection of Channel 0 and 1

Note: After turning on the power, wait about 15 minutes for more accurate measurements. Most data acquisition boards and sensors take about 10-15 minutes to warm up and stabilize its signals. If the signal is sampled before the first 15 minutes, the signals could be unstable and slightly vary with time. This effect was particularly observed from the LINEAS sensors in the lab condition. It might be due to the initial electric charge condition and the subsequent electrical current contact with the piezoelectrical element of the LINEAS sensors, but giving about 15 minutes of warm-up time would be a simple, good practice.

5.3 Data Acquisition

Run **WIMDaqLT.exe** using the shortcut icon on the desktop or using Start → All Programs → WIMDaqAnal → **WIMDaqLT.exe**
Then Figure 28 should appear.

The software name "WIMDaqLT" was derived from "WIM Data Acquisition using Lap Top computer."

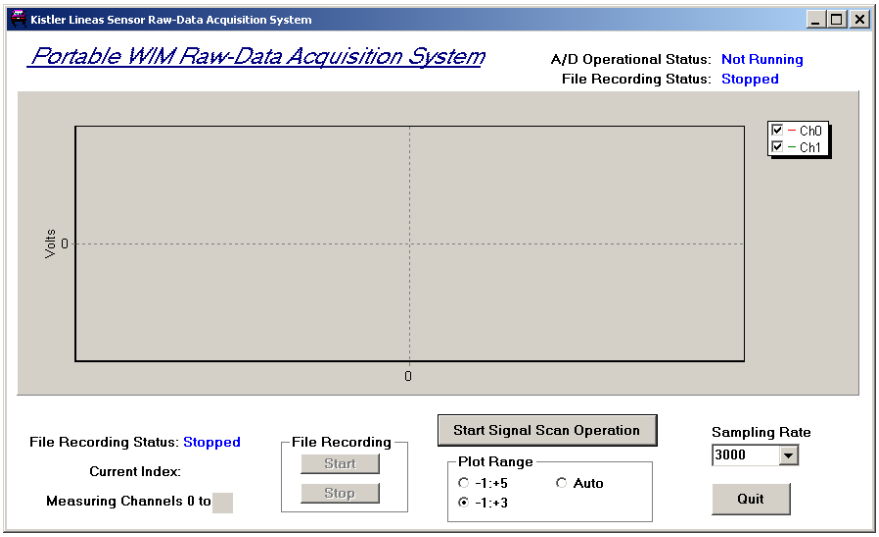


Figure 28: WIMDaqLT.exe initial screen

Follow the below steps:

1. First choose the sampling rate.
   The default value is set to 3,000 samples per second (S/s), which would be sufficient for WIM. You could select 4,090 S/s for more accurate measurements.
2. Press the "**Start Signal Scan Operation**" button.
   Data is always sampled from both channels (Ch0 and Ch1), but you can choose to display only one channel at a time or simultaneously by removing or setting the check marks on the graph legend. Once sampling starts, the start button is changed to a stop button and the label is changed to "**Stop Signal Scan Operation.**" At the same time, the Quit button and Sampling Rate combo buttons are disabled to prevent inadvertent errors. *In order to change the sampling rate or to quit the program, the signal sampling (scanning) operation must be stopped first.*
3. To begin recording the data, press the **Start** button on the File Recording group.
   The sampled raw data (binary format) is stored at "C:\Program Files\WIMDaqAnalysis\DataAcquisition\yyyymmdd" where yyyymmdd is the directory name automatically created based on the time of sampling date, i.e., year, month, and day. The raw data file is automatically created with the name hhmmss_SamplingRate.bin where hhmmss is the starting time of data acquisition. For example, if the sampling was started at 14:05:20 with 3,000 Samples/sec, the file name would be 140520_3000.bin. Each data file stores up to 20 seconds worth of data and another file is created for the next 20 seconds of data, and it repeats until the user presses the **Stop** button on the File Recording group. How many files have been recorded are shown in the File Recording Status with the format ##:## (number of files: number of seconds). O*nce recording starts, the signal scan operation button is disabled because data cannot be recorded without*

sampling. You must stop the recording first in order to stop the signal scan operation.

4. To stop recording data, press the **Stop** button on the File Recording group.

When the Stop button is pressed, the recording stops after completing saving of the data for current second in order to make sure that a proper boundary of the data is stored in the recorded data.

During at any time, the plot range can be selected. The choices include "–1V to +3V", "-1V to +5V", and Auto. Auto range allows plot of data between the maximum and minimum of the data range within the window. The x-axis labels denote sampling instances, i.e., for sampling rate 3,000 S/s, each window displays 3,000 points data using the actual measured voltages.

Quitting Sequence of the Program

It is important to follow a proper quitting sequence, i.e.,

1. Stop recording
2. Stop signal scan
3. Quit

After recording the data, ASCII conversion of data can be done using the Analysis tool. It produces a standard comma separated format, and the file can be directly loaded into MS Excel.

5.4 Data Analysis

To run data analysis software, run **DataAnal.exe** from the desktop or use Start → All Programs → WIMDaqAnal → **DataAnal.exe**

Step 1: Load the data file using the File menu "Open Data File"

Until the data is loaded, all control buttons are disabled. Once data is loaded, you will see an example screen shown in Figure 29.

There are a number of visualization tools available, which are summarized below:

- → move to next data window
- ← move to previous data window
- **->>** move two data windows forward
- **<<-** move two data windows backward
- **->|** move to end of data
- **|<-** move to beginning of data
- **"Show All"** show all data as a slide show
- Data Y-range may be changed using the slide bars or set to Auto.
- The number of data points to be displayed per window can be changed by setting the value at the Block Size text box and clicking the Set button. It is useful when details of signal need to be observed.

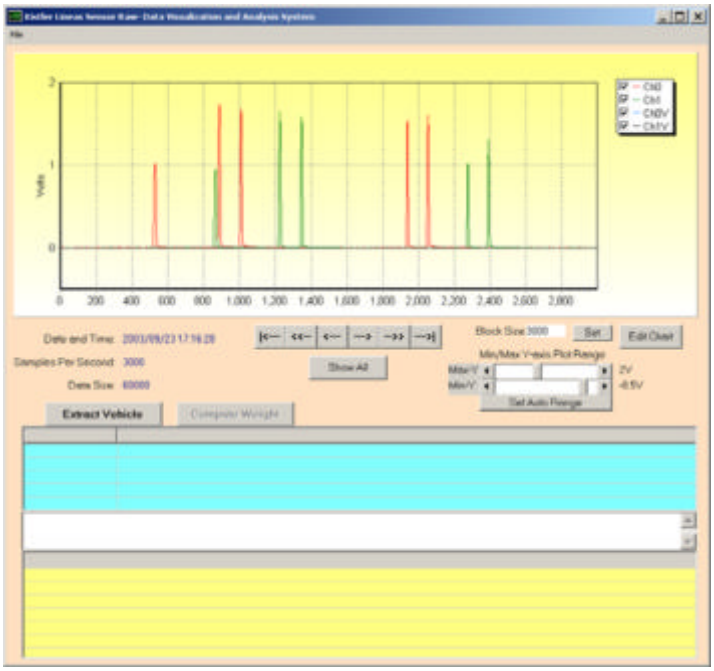


Figure 29: Data Analysis Screen

Graph legends

- Ch0 --- channel 0 raw data
- Ch1 --- channel 1 raw data
- Ch0V --- channel 0 raw data after wheel detection
- Ch1V --- channel 1 raw data after wheel detection

By removing or setting the check marks, you can choose which graph to view.

Edit Chart button

Cool things can be displayed using the various visualization tools by pressing the **Edit Chart** button (Figure 29). For example, by simply setting the check mark on the 3 Dimension as shown in Figure 30, you can almost see the vehicle as shown in Figure 31.

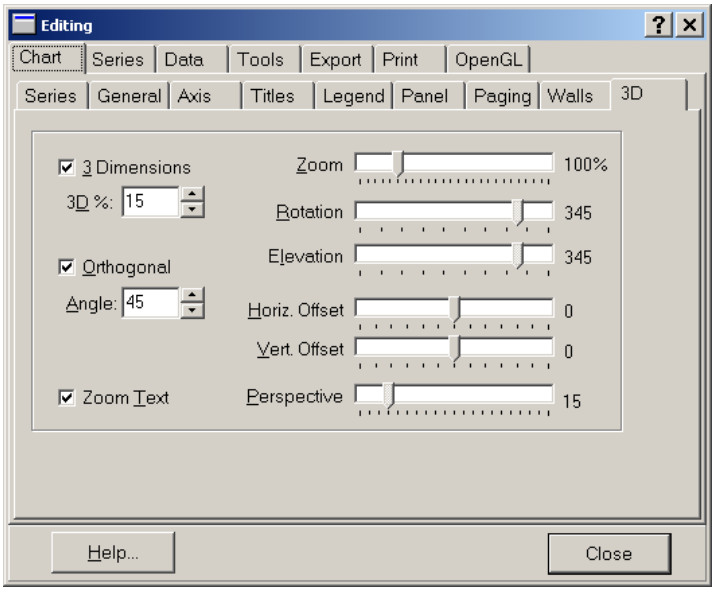


Figure 30: Chart Editing Tool

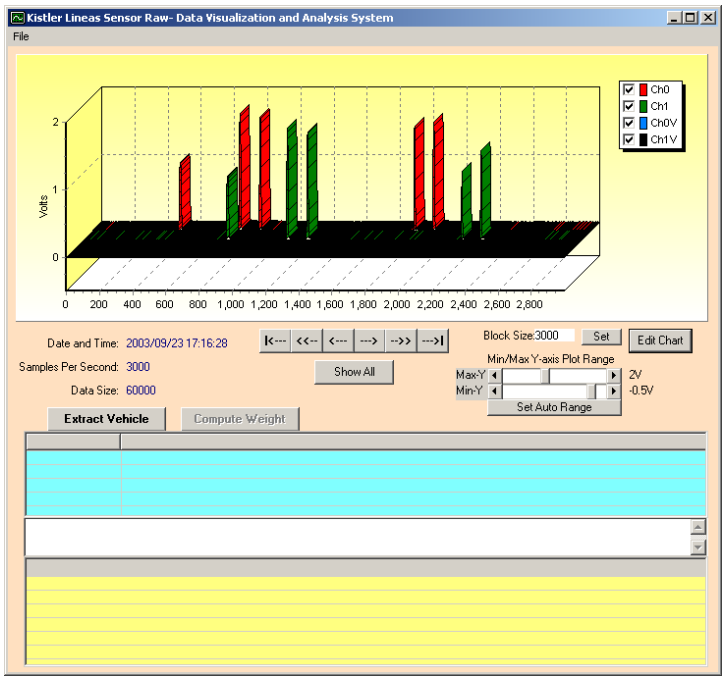


Figure 31: 3-D rendering using the Edit Chart tool

Step 2: Extract vehicle signals by pressing the "Extract Vehicle" button
In this step, a fairly complicated algorithm runs and determines which wheel signal belongs to which vehicle. The detected wheels are indexed for both channels and the measurements are displayed on the tables on the window (see Figure 32, the top and middle tables). It also produces intermediate data files in the directory "C:\Program Files\WIMDaqAnalysis\Analysis\". The file format of the output text files is shown in Table 4.

Table 4: Analysis Output Data Format

<table>
<tr><td>Speed.txt</td><td>Speed estimation using wheel footprint used by the algorithm. It is not the actual speed, but the computation of [0.17 * (Sample rate)/(Width count)] and is only produced to be used by the developers.</td></tr>
<tr><td>Wheels.txt</td><td>This file can be used by the general users. It provides all of the measurement values for each wheel and channel. This portion is also displayed in the text window. The data fields of this file are:
<table>
<tr><th>Data Field</th><th>Description</th></tr>
<tr><td>whIndex</td><td>The index that identifies wheel</td></tr>
<tr><td>BeginI</td><td>Sample instant index of the start of the wheel footprint</td></tr>
<tr><td>EndI</td><td>Sample instant index of the end of the wheel footprint</td></tr>
<tr><td>maxIndex</td><td>Sample instant index where maximum voltage is observed within the footprint of the wheel</td></tr>
</table>
</td></tr>
</table>

	maxValue	The voltage at the maxIndex
	Area	Sum of all voltage values under the curve of each wheel
	IdleLevel	The voltage level between wheels. If this level is not close to zero, the sensor has a hardware problem and requires investigation of the source of the problem.
	noiseLevel	The noise level is computed for the idle period by computing mean square root error of the signal level. This value is always positive and should be close to zero, otherwise it indicates that the sensor is experiencing a hardware problem.
VehWheels.txt	This file shows which wheels belong to which vehicle. Wheels are expressed using the wheel index found in wheels.txt file. Vehicles are indexed starting from 0. Each vehicle occupies one row with the corresponding wheel indices.	

Step 3: Compute vehicle weight, axle distances, and speed by pressing the "Compute Weight" button

When you press the "**Compute Weight**" button, based on the wheel analysis the algorithm computes speed, wheel weights, axle distances, and total weight. These values are computed using wheel data derived from Step 2. Since the data is available from both channels, two computational results are available. This data is shown in the bottom table of the screen as well as recorded as a text file "vehRecord.txt" in the directory "C:\Program Files\WIMDaqAnalysis\Analysis\" for future analysis. The data fields of the computed results are recorded as follows.

Table 5: WIM Parameter Columns

Data Field	Description
vehIndex	Vehicle indexed starting from 0
DateTime	The date and time of the vehicle crossed the sensor
mph	Speed in mph
WheelWeights	Wheel weights in pounds. Each wheel is separated by "-" For example, 5 axle vehicle is expressed as 4717-6067-5833-3847-5487. For screen, 4717=6067=5833=3847=5487.
TotalWeight	Sum of each wheel weights in pounds
axleDistances	Axle distances in feet. Each axle distance is separated by "-". For screen display, "=" is used in place of "-".

NOTE: If the content of any column is partially not visible due to not enough space on the screen, you can always place the screen cursor at the border of the title column of the table and drag it to adjust the column size.

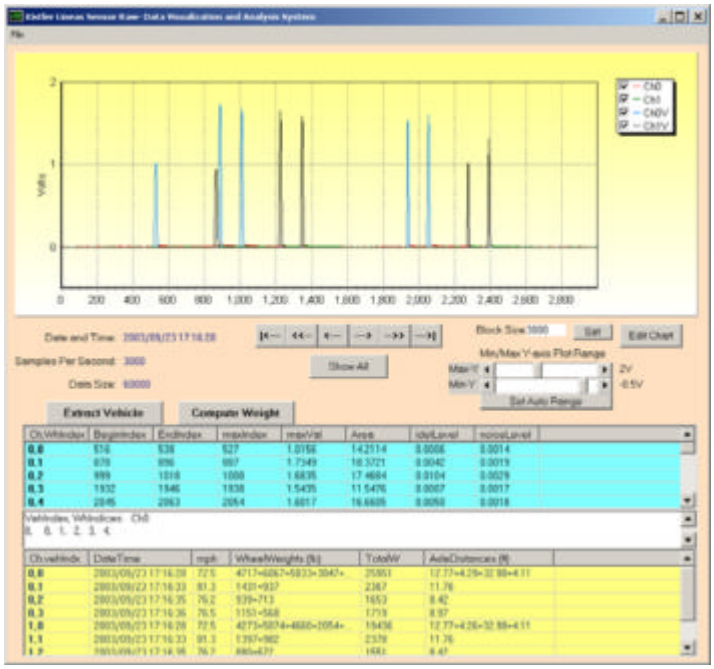


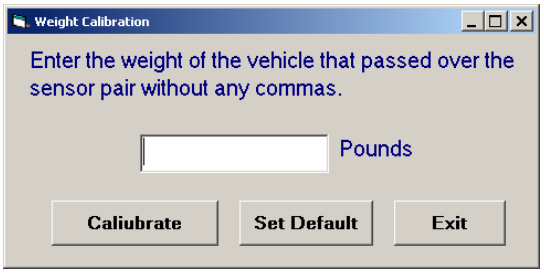
Figure 32: Completed window

Weight Calibration

The weight is computed using the calibration factor derived from a typical factory setting of the sensor sensitivity. This value is reasonably accurate and can be used without modification. However, if you wish to make more accurate measurement, you can change the calibration factor. Use the following steps.

Step 1: Collect data using a vehicle with a known weight. The software uses the first vehicle detected from the data set.

Step 2: Under file menu, select "**Calibrate Weight**" item, which will open up the following dialog window.

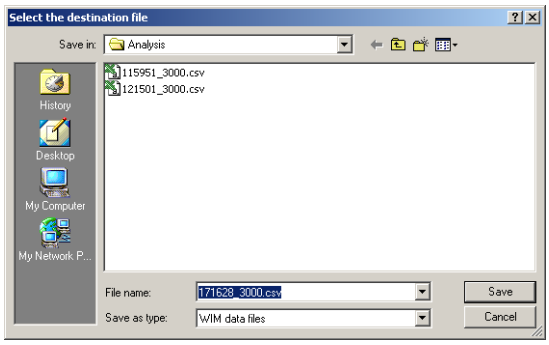


Step 3: Enter the known weight of the first vehicle detected

Step 4: Press the **Calibrate** button and then Exit.
Once the calibration is done, this factor is affected from the next weight computation. Since this calibration factor is kept in the Windows registry, its value is kept in computer even if you exit or reset the program. If mistakes were made, you can always go back to the factory default setting by clicking the “Set Default” button.

Conversion of Binary Raw Data to Excel Readable File

The raw binary data acquired from sensors can be converted into a comma separated values (CSV) file that can be directly loaded into a MS Excel. From the File menu select “Save As CSV File”, then the raw data presently loaded will be converted into a CSV file with the format “index, ch0 voltage, ch1 voltage”. Before the converted file is saved, the software provides a file dialog from which the destination file name and location can be typed in. Using the provided default filename would be simple and minimize the effort.



5. 5 Static Weight Test Tool

This tool is provided for testing LINEAS sensors before installing on the pavement. For use with this tool, **plug in the BNC connector of LINEAS to the Ch0 (or Ch1) of the interface box.** Then run "StaticWeight.exe". It will display the weight converted values sampled from the channel selected and actual voltage monitored as well as the ground level traced from the signal. It should be emphasized that this tool is not designed for measuring moving vehicles because it does not compute the footprint of wheels. It was only intended for a simple test of LINEAS sensors using your known weight by stepping on to the sensor before installation.

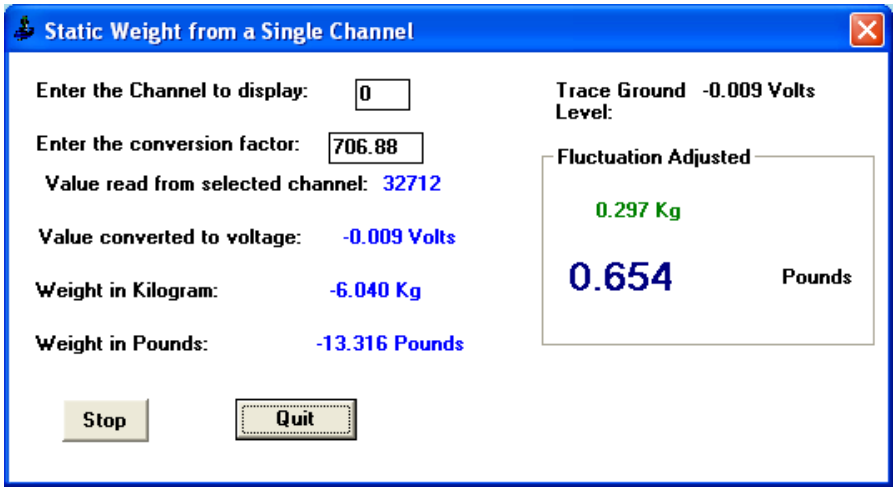


Figure 33: Static Weight Testing Tool

Weight computation is done using the conversion factor computed based on the sensitivity of LINEAS sensors provided by the Kistler data sheet. The default conversion factor computed from the Kistler data sheet is 706.88 as shown in Figure 33. You can edit the conversion factor while it displays the weights. Increasing this value will increase the weight values and decreasing it will decrease the weight values.

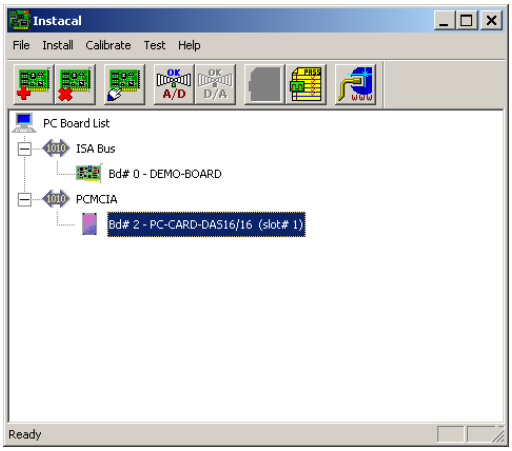
The example in Figure 33 was shown using Ch0. If you wish to test it using Ch1, connect the BNC from the sensor to Ch1 and simply type in "1" in the text box labeled "Enter the Channel to display".

The Start and Stop button works as a toggle button. To exit the program, press Stop and then the Quit button.

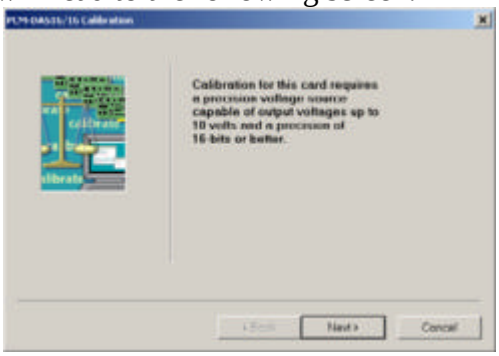
5.6 Calibration of DAS16/16 Card

The Das16/16 card was calibrated at the laboratory, but you can calibrate it if an accurate 16-bit resolution DC signal is available. The manufacturer recommends 16 bit accuracy of voltage signal for this calibration. The voltages required are +6V, -6V, +3V, -3V, +1V, -1V, +0.75V, and –0.75V. You should avoid this process if you don't know the circuit. Mistake can be very costly.
The steps are:
Step 1: Run InstaCal using

Start → All Programs → Measurement Computing → **InstaCal**



Step 2: Select Bd#2 under PCMCIA. Right click and select Calibrate→A/D, which will lead to the following screen.



Step 3: Click the Next button and follow the steps.
!!!Caution!!!: You need to open the interface box to apply the calibration voltage. The calibration voltage should be applied to Pin 2 (signal+) after carefully removing the yellow colored line and the (signal –) to Pin 1 and 3. Improper connection of voltage sources can damage the charge amplifier (expensive, don't do it if you don't know the circuit.)

5.7 Technical Data

A/D Maximum Sampling Rate: 100 K Samples per second
A/D Sampling Resolution: 16 bit
Daq. Card: PCMCIA-Card DAS16/16
Voltage Sampling Range: -5V to + 5V
Number of Channels: 2
Each Channel Sampling Rate: 3,000 or 4,090
Charge Amplifier: Kistler 5038A2Y43
Measuring Range: ±60,000 pC

Caution: Never apply any direct voltage source to the interface box BNC input.

REFERENCES

[1] Archived Data User Service (ADUS), "ITS Data Archiving: Five-Year Program Description," March 2000, Published by U.S. DOT, ADUS Program.

[2] Box, G.E.P., G.M. Jenkins, and G.C. Reinsel, *Time Series Analysis – Forecasting and Control*, 3rd ed., Englewood Cliffs, NJ; Prentice Hall, 1994.

[3] Chatfield, C., *The Analysis of Time Series – An Introduction*, 5th ed., London, UK; Chapman and Hall, 1996.

[4] Chen C., K. Petty, A. Skabardonis, and P. Varaiya, "Freeway Performance Measurement System: Mining Loop Detector Data," *Transportation Research Record 1748*, TRB, Washington, D.C., 2001, pp 96-102.

[5] Chen C., J. Kwon, J. Rice, A. Skabardonis, and P. Varaiya, "Detecting Errors and Imputing Missing Data for Single Loop Surveillance Systems," *TRB 82nd Annual Meeting CD-ROM*, Washington, D.C., Jan 2003.

[6] Cleghorn D., F. Hall, D. Garbuio, "Improved Data Screening Techniques for Freeway Traffic Management Systems," *Transportation Research Record 1320*, TRB, Washington, D.C., 1991, pp 17-31.

[7] Coifman B., "Using Dual Loop Speed Traps to Identify Detector Errors," *Transportation Research Record 1683*, TRB, Washington, D.C., 1991, pp 47-58.

[8] Dahlin C, "Proposed Method for Calibrating Weigh-in-Motion Systems and for Monitoring That Calibration Over Time," *Journal of the Transportation Research Board*: *Transportation Research Record 1364*, pp. 161-168, National Academy of Science.

[8] Daily D. J., *Improved Error Detection for Inductive Loop Sensors*, WA-RD 3001 Washington State Department of Transportation, May 1993.

[10] Edwards M. "How to build and profit," *Communication News*, Vol. 32, No. 11 (November 1995), 49.

[11] Fairhead, Neal. "Data warehousing," *Business Quarterly*, Vol. 60, No. 2 (January 1995), 89 - 94.

[12] FHWA, *Traffic Detector Handbook*, 2nd Edition, FHWA-IP-90-002, Research Development and Technology, Turner-Fairbank Highway Research Center, McLean, Virginia, July 1990

[13] Fogarty K. "Data mining," *Network World*, Vol. 11, No. 23 (June 1994a), 40-43.

[14] Gelman A., J. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis, Texts in Statistical Science*, Chapman & Hall/CRC, 1995.

[15] Gold D., S. Turner, B. Gajewski and C. Spiegelman, "Imputing Missing Values in ITS Data Archives for Intervals Under 5 Minutes," *TRB 80th Meeting CD- ROM*, Paper No. 01-2760, 2001.

[16] Goucher, G.C. and Mathews S.S., "A Comprehensive Look at CDF," NSSDC/WDC-A-R&S 94-07, NASA/Goddard Space Flight Center, August 1994.

[17] Gajewski B., Turner S., Eisele W., and Spiegleman C., "ITS Data Archiving: Statistical Techniques for Determining Optimal Aggregation Widths for Inductance Loop Detector," TRB 2000, Washington DC, Jan 2000.

[18] HDF: http://hdf.ncsa.uiuc.edu/

[19] Huffman, D. A., "A Method for the Construction of Minimum Redundancy Codes", Proceedings of the Institute of Radio Engineers, September 1952, Volume 40, Number 9, pp. 1098-1101.

[20] Jacobson L., N. Nihan, and J. Bender, "Detecting Erroneous Loop Detector Data in a Freeway Traffic Management System," *Transportation Research Record 1287*, TRB, Washington, D.C., 1990, pp 151-166.

[21] James, I.W., "The Inductive Loop Vehicle Detector: Installation Acceptance Criteria and Maintenance Techniques," California Department of Transportation, Sacramento Transportation Laboratory, Sacramento California, Federal Highway Administration, Washington, D.C., March 1976.

[22] NCHRP, "A guidebook for performance-based transportation planning," NCHRP Report 446, National Cooperative Highway Research Program.

[23] Kistler Instrument Corp., "Signal Processing Requirements for WIM LINEAS Type 9196," 20.218e 6.00 Kistler Application Note, Winterthur, Switzerland, 2000.

[24] Kwon T.M. and Dhruv N., "Unified Transportation Sensor Data Format (UTSDF) for Efficient Archiving and Sharing of Statewide Transportation Sensor Data," *Proc. of the Transportation Research Board 83nd Annual Meeting*, Washington D.C., Jan. 2004.

[25] Kwon T.M, Dhruv N., Patwardhan S., "Common Data Format Archiving of Large-Scale Intelligent Transportation Systems Data for Efficient Storage, Retrieval, and Portability," *Journal of the Transportation Research Board*: *Transportation Research Record 1836*, pp. 111-117, National Academy of Science, 2003.

[26] Kwon T.M.and Fleege S., "R/WIS Architecture for Integration and Expansion," *Journal of the Transportation Research Board*: *Transportation Research Record 1700*, pp. 1-4, The National Research Council, The National Academies, 2000.

[27] Ladaga J., "Let business goals drive your data warehouse effort," *Health Management Technology*, Vol. 16, No. 11 (October 1995), 26-28.

[28] Margiotta R, *ITS as a Data Resource: Preliminary Requirements for a User Service.* Report FHWA-PL-98-031, Federal Highway Administration, Washington, DC, April 1998.

[29] Oppenheim A.V., A.S. Willsky, Ian T. Young, *Signals and Systems,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.

[30] Little, R. J. A. and D. B. Rubin, *Statistical Analysis with Missing Data*, Wiley Series in Probability and Mathematical Statistics, 1987.

[31] Naidu, P.S., *Modern Spectrum Analysis of Time Series*, Boca Raton, FL; CRC Press Inc., 1996.

[32] Peeta S. and I. Anastassopoulos, "Automatic Real-Time Detection and Correction of Erroneous Detector Data Using Fourier Transforms for On-line Traffic Control Architectures," *TRB 81st Annual Meeting CD-ROM*, Washington, D.C., Jan 2002.

[33] Rubin, Donald B., *Multiple Imputation For Non-Response in Surveys*, Wiley Series in Probability and Mathematical Statistics, 1987.

[34] Schafer, J. L., *Analysis of Incomplete Multivariate Data*, Chapman & Hall/CRC Publication, 1997.

[35] Schmoyer, R., P. Hu, and R. Goeltz, "Statistical Data Filtering and Aggregation to Hour Totals of ITS Thirty-Second and Five-Minute Vehicle Counts," *TRB 80th Annual Meeting CD-ROM*, Paper No 1769, 2001.

[36] Smith B., D. Lewis, R. Hammond, "Design of Archival Traffic Databases: Quantitative Investigation into Application of Advanced Data Modeling Concepts," *Journal of the Transportation Research Board*: *Transportation Research Record 1836*, pp. 126-131, National Academy of Science, 2003.

[37] B.L. Smith, W. T. Scherer, J. H. Conklin, "Exploring Imputation Techniques for Missing Data in Transportation Management Systems," *Transportation Research Record 1836*, TRB, Washington, D.C., 2003, pp 132-142.

[38] Treinish, L.A., "Data Structures and 'Access Software for Scientific Visualization," A Report on a Workshop at Siggraph'90, Computer Graphics, 25, No. 2, April 1991.

[39] Treinish, L.A. and Goucher G.W., “A Data Abstraction for the Source-Independent Storage and Manipulation of Data,” National Space Science Data Center Technical Paper, NASA/Goddard Space Flight Center, August 1988.

[40] Treinish, L.A. and Gough M.L., “A Software Package for the Data-Independent Storage of Multi-Dimensional Data,” EOS Transactions, American Geophysical Union, 68, pp. 633-635, 1987.

[41] Tuner, S.M., Eisele, W.L., Gajewski, B.J., Albert, L.P., and Benz, R.J., *ITS Data Archiving: Case Study Analysis of San Antonio TransGuide Data.* Report FHWA-PL-99-024, Federal Highway Administration, Texas Transportation Institute, College Station, Texas, August 1999.

[42] Wall J., and D.J. Daily, “An Algorithm for the Detection and Correction of Errors in Archived Traffic Data,” *TRB 82nd Annual Meeting CD-ROM*, Washington, D.C., Jan 2003.

[43] Warner, R. M., *Spectral Analysis of Time-Series Data*, New York, NY; Guilford Press, 1998.

[44] Ziv J. and Lempel A., “A Universal Algorithm for Sequential Data Compression”, IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343, 1977.