DEPARTMENT OF TRANSPORTATION

Integrate RTMC Vehicle Classification Into the Current Detector Volume Data

Taek Kwon, Principal Investigator
Department of Electrical Engineering
University of Minnesota Duluth

NOVEMBER 2020

Research Project
Final Report 2020-31

To request this document in an alternative format, such as braille or large print, call 651-366-4718 or 1-800-657-3774 (Greater Minnesota) or email your request to ADArequest.dot@state.mn.us. Please request at least one week in advance.

Technical Report Documentation Page

1. Report No.	2.	3. Recipients Accession No.	
MN 2020-31			
4. Title and Subtitle Integrate RTMC Vehicle Classification into the Current Detector Volume Data		5. Report Date November 2020 6.	
7. Author(s) Taek M. Kwon		8. Performing Organization Report No.	
9. Performing Organization Name and Address Department of Electrical Engineering University of Minnesota Duluth 271 MWAH, 1023 University Drive Duluth, MN 55812		10. Project/Task/Work Unit No. CTS #2020018 11. Contract (C) or Grant (G) No. (c) 1003325 (wo) 125	
12. Sponsoring Organization Name and Address Minnesota Department of Transportation Office of Research & Innovation 395 John Ireland Boulevard, MS 330 St. Paul, Minnesota 55155-1899		13. Type of Report and Period Covered Final Report 14. Sponsoring Agency Code	
15. Supplementary Notes http://mndot.gov/research/reports/2020/202031.pdf			
16. Abstract (Limit: 250 words) Collection of vehicle classification data is considered an essential part of traffic monitoring programs. The objective of this project is to integrate the raw classification data generated by the Minnesota Department of Transportation (MnDOT) Regional Transportation Management Center (RTMC) into the existing volume data managed by the Traffic Forecasting and Analysis (TFA) Section under the Office of Transportation System Management (OTSM). RTMC manages a large number of traffic sensors in the Twin Cities' freeway network and continuously collects a huge amount of traffic data. Recently, it added Wavetronix radar sensors, from which length-based classification and speed data are generated in addition to typical volume and occupancy data generated by loop detectors. This project integrates this classification data into the existing TFA volume data, which could save cost and time for TFA in the future by using existing classification data. The project team also integrated the RTMC speed data for the locations where it was available. The final deliverable of this project was a software tool called detHealth_app, from which users can retrieve classification and speed data in addition to volume/occupancy data in multiple formats including Federal Highway Administration (FHWA) format. The detHealth_app program was thoroughly tested and has been successfully used by MnDOT TFA.			
17. Document Analysis/Descriptors Traffic Surveillance, Vehicle length, Speed data, Computer programs		18. Availability Statement No restrictions. Document available from: National Technical Information Services, Alexandria, Virginia 22312	
19. Security Class (this report) Unclassified	20. Security Class (this page) Unclassified	21. No. of Pages 55	22. Price

INTEGRATE RTMC VEHICLE CLASSIFICATION INTO THE CURRENT DETECTOR VOLUME DATA

FINAL REPORT

Prepared by:

Taek M. Kwon
Department of Electrical Engineering
University of Minnesota Duluth

November 2020

Published by:

Minnesota Department of Transportation
Office of Research & Innovation
395 John Ireland Boulevard, MS 330
St. Paul, Minnesota 55155-1899

ACKNOWLEDGMENTS

The author would like to thank the Minnesota Department of Transportation (MnDOT) for providing financial and technical support for this implementation project. Special thanks are extended to the Technical Liaison John Hackett and Project Coordinator Katie Fleming for helping successfully complete the project. Thanks are also extended to the following members of the Office of Transportation System Management (OSTM) at MnDOT for their involvement in the Technical Advisory Panel (TAP) meetings and invaluable feedbacks: Darin Mertig, Christina Prentice and Benjamin Timerson. Finally, yet importantly, the author would like to thank Doug Lao at the Regional Transportation Management Center (RTMC) for providing assistance in speed and classification data formats and troubleshooting help for accessing the RTMC Intelligent Roadway Information System (IRIS) server.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MnDOT	Minnesota Department of Transportation
FHWA	Federal Highway Administration
AASHTO	American Association of State Highway and Transportation Officials
RTMC	Regional Transportation Management Center
IRIS	Intelligent Roadway Information System
OTSM	Office of Transportation System Management
TFA	Traffic Forecasting and Analysis
AADT	Annual Average Daily Traffic
COV	Conservation of Vehicles
CC	Continuous Count
SC	Short-Duration Count
GUI	Graphical User Interface
CSV	Comma Separated Values
IT	Information Technology

EXECUTIVE SUMMARY

Collection of traffic volumes by vehicle classification is an essential part of traffic monitoring [1-4]. This collection method is not only affected by the volume of each category but also by the categorization system. The objective of this project is to integrate vehicle classification data into the existing volume database.

The Regional Transportation Management Center (RTMC) at the Minnesota Department of Transportation (MnDOT) maintains a large number of traffic sensors on the Twin Cites' (St. Paul and Minneapolis) freeway network. Traffic sensors mostly comprise inductive loop detectors but more Wavetronix radar detectors have recently been added. This detector provides speed and classification in addition to typical volume and occupancy data. The MnDOT Traffic Forecasting and Analysis (TFA) office wants to use these available data and integrate them into the existing volume database created from the previous project [5]. The main benefit of this integration would be that MnDOT TFA could obtain speed and classification data without new sensor installations or human resource for data collection. Moreover, since RTMC publishes its data online and encourages its uses in other offices, use of RTMC classification data by TFA is a win-win for both offices. Thus, a proposal to integrate the RTMC classification data to the existing volume database was submitted, and this project was approved and started.

The database system developed in the previous project [5] was based on a client/server architecture, and it was to take advantage of the premise that all heavy computations are performed at the server side and only light computations are needed in the client side. This allows less development time and complexity codes in the client side. It also has an advantage that the client and server software can be evolved independently with a standardized interface established between them. However, one critical issue MnDOT TFA raised was that maintaining a database server would be expensive in terms of human resources, i.e., someone has to manage the database and take care of the server maintenance software and hardware. For example, if the detector-health database housed in the hard disk at the server side crashes, someone has to restore the database from the crashed hard disk. Demand of high availability and reliability of the server may also require use of a virtual machine (VM) or a cloud service, which would add a recurring cost. In small offices such as MnDOT TFA, adding an information technology (IT) personnel or using a VM would be difficult to justify. After a few meetings and discussions, the research team and MnDOT TFA decided to convert the current server-based system to a server-less, stand-alone system in which a single program installed on a user's personal computer (PC) independently provides all of the required data services, i.e., without receiving data from the database server. The task of conversion was carried out in this project.

To remove dependency from the server while providing equivalent data services, the application program now must include both the server and client functions in a single large program. Fortunately, when the server program was developed, it also created a structured directory tree along with daily comma separated value (CSV) files that exactly matched with the daily entries of the corresponding database tables for uses in backup and restoration. The research team was able to adapt this utility to the stand-alone model and created new types of data queries based on structured CSV files. Although

the sequential nature of CSV files was a concern for search efficiency, it worked really well for the current program because network-wide data were only retrieved one day at a time according to the date selected by user. Because each CSV file contained a single day of data for the entire network and its filename was encoded with the date of traffic, retrieval performance was comparable to that of the relational database since it used a search from the efficient hierarchical tree of the operating system (OS) file.

The original name of the client-side program called “detHealth_app” in the previous project was retained in the stand-alone model, as was the method of organizing related data functions by Windows tabs. The graphical user interfaces (GUIs) of the original “detectors” and “r_node” tabs were unchanged in the new version, except that all data functions now run without retrieving data from the database server. According to MnDOT TFA, the “COV (conservation of vehicles)” tab was not useful in the original version, and thus it was replaced with a new function providing annual average daily traffic (AADT) for short-duration count stations. After completing the conversion from the original application tabs developed in the previous project, the main purpose of this project, length-based classification, was implemented in a new tab named “LEN Class.” During the same period, MnDOT TFA also asked the research team to integrate the speed data available from RTMC Wavetronix sensors. Speed data applications were written and integrated to the “Volume” tab because of the same GUI and input requirements, and then its name was changed to “Volume/Speed.” In addition, a new tab named “Bulk FHWA” was created, which implemented batch functions for generating a large amount of volume, speed, and classification data in TMG-2016 FHWA format.

In summary, this project successfully converted the detHealth_app software to a stand-alone program from a server-based program and then integrated speed and classification data to the existing volume data. The “Volume/Speed” and “LEN Class” tabs were created for retrieving or exploring data one station at a time. For generating data for a large number of stations, the “Bulk FHWA” tab was created for retrieval of all three types of data in FHWA format. The software was delivered to MnDOT TFA on time and has been thoroughly tested and debugged. Since then, the software has been successfully used by MnDOT TFA for applications in real traffic monitoring.

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

In the previous research project entitled "Improve Traffic Volume Estimates from MnDOT's Regional Traffic Management Center," the research team created a relational database server that houses detector-health parameters and traffic volume data for the entire vehicle detectors in the Twin Cities' freeway network managed by the Regional Transportation Management Center (RTMC) [5]. The system was designed as a typical client/server architecture in which client programs installed on users' personal computers (PCs) remotely access the server to query and receive detector-health and traffic volume data. An application called detHealth_app was developed as the main client program as a part of the project. The current project was created as an extension of this work and was proposed to integrate RTMC vehicle classification data into the current volume database.

RTMC provides length-based vehicle classification that classifies vehicles into four types: motorcycle, short vehicle, medium vehicle, and long vehicle. Classification data are available at the roadway locations where Wavetronix traffic sensors (microwave radar detectors) have been installed. MnDOT Traffic Forecasting and Analysis (TFA) under the Office of Transportation System Management (OTSM) wants to use this existing data by integrating them into the current volume database. The four-category length-based classification is not the Federal Highway Administration (FHWA) 13-category classification system [1-4] but rather one of the recognized classifications by FHWA since the publication of TMG-2013 [2], i.e., the data files with extension ".LEN" that conform the length-based classification format defined in TMG-2013 are allowed to be submitted to the FHWA Travel Monitoring Analysis System (TMAS) [2]. For managing quality control as well as TMAS submissions of volume and length-based classification data, MnDOT TFA has been working with the company High Desert Traffic (HDT) LLC. HDT manages an online traffic-data management system called Jackalope to which MnDOT LEN and "VOL" files along with other traffic information are uploaded. One of the outcomes of this project has been to make uploading the FHWA formatted data files produced by detHealth_ap to the Jackalope system feasible.

MnDOT RTMC manages the Twin Cities' freeway network including all traffic controls, incidents and maintenance of traffic detectors. The detectors produce real-time data every 30 seconds for 24 hours a day, 7 days a week, all year around, accumulating a huge amount of data. For applications requiring daily volumes, directly using this data is inconvenient since it requires downloading of 30-second data and then conversion of the data to daily volumes. In particular, issues arise if the 30-second data contain missing data in some time slots, i.e., how the missing data should be counted toward the total volume of the day is an important issue. In the previous project, this drawback was addressed by creating a database server that houses daily volumes of all RTMC detectors along with detector diagnostics (quality) data, i.e., it provides daily volumes along with percent of missing data information as well as 13 different detector-health parameters [5]. Availability of relational database allowed efficient search and retrieval of a large amount of daily volumes in both temporal and spatial relations.

The database server created in the previous project was installed on a desk space in the TFA section, which was physically insecure and later became an issue. For example, one of the temporary employees at MnDOT TFA accidently disconnected the RJ45 network connector in the server to use the connection for a laptop and then did not connect back to the server. This caused inaccessibility to the Internet and failures of raw-data downloads by the server as well as unavailability of the server itself to clients for a few days and created data-loss problems. Near the end of the project, MnDOT TFA also raised several issues related to server management. The first one was its location and security to control physical accesses to the server, i.e., only authorized persons should have physical access to the server. This would require a security-controlled room and an access policy. The other issues included how to ensure high availability, high reliability, and security from cyberattacks. MnDOT TFA, information technology (IT) offices and the University project team had multiple meetings and discussed potential solutions. The following were some of discussions and solutions. High reliability may be obtained through database duplications and load sharing. High availability may be provided through implementation of the database and the server software in an online virtual machine (VM) or cloud. Online security is a challenging issue but using solutions provided by the VM provider and MnDOT firewall may be sufficient. However, all of these efforts would require extra costs and human resources, which would not be easily justifiable or obtainable for a small office like MnDOT TFA. At the end, the most critical issue was who would maintain the database and the server and how would they be maintained when MnDOT takes over the server on completion of the project. For example, who will restore the database when it crashes, who will upgrade the server software when changes are needed, who will redesign the database tables when new types of data must be added or removed, and last, but importantly, who will pay for all of these extra efforts and cost? The conclusion reached in the last Technical Advisory Panel (TAP) meeting was that adding an IT specialist who can maintain the database in VM and manage the server software for TFA was too expensive and not justifiable. Consequently, MnDOT TFA and the University research team agreed to eliminate the server and convert the current server-based system to a server-less, stand-alone application program. More specifically, a new application program would be developed with the goal that it should provide equivalent data services of the client/server model but without needing to connect and retrieve data from the server.

Conversion to a stand-alone application program from a server-based architecture was carried over to this project. To remove dependency from the server while providing equivalent data services, the application program now requires inclusion of both the server and client functions in a single large program. Fortunately, when the server program was created, it also used a structured directory along with daily comma separated value (CSV) files that match the daily database table entries for backup and restoration. The research team was able to adapt this utility to the stand-alone model by adding data retrieval codes based on CSV files. The only potential drawback is that the sequential nature of CSV files is inherently inefficient for random searches. However, it worked efficiently for detHealth_app because network-wide data is only retrieved for one day at a time according to the date selected by the user, and each CSV file under the hierarchical directory tree of the operating system (OS) contains a single day of data for the entire network. Since CSV filenames were encoded with the date of traffic under a directory tree organized by temporal relations, the search uses the benefit of a hierarchical search algorithm implemented in the OS, providing a fast search.

Another type of retrieval frequently used in detHealth_app was retrieval of historical data for a station typically comprised of few detectors. This type of retrieval function was implemented using real-time computation of daily data from the raw 30-second detector data. The RTMC Intelligent Roadway Information System (IRIS) server allows a query of raw volume and occupancy data (30-second data) for any detector from any date for one day at a time. Because the amount of historical data retrieved from a few detectors for a period in common applications is relatively small, such as one year, the time required for retrieval and computation of daily volumes or health parameters for a station is short enough to feel real time.

In this project, the original name of the client-side program called detHealth_app was left unchanged, and the basic method of using a Windows tab for implementing a similar set of functions remained the same. The same graphical user interfaces (GUIs) of the original "Detectors" and "r_node" tabs were carried over to the new version, except that all utility functions now work without retrieving data from the detector-health server. The COV tab in the original version was not very useful, according to MnDOT TFA, and thus it was changed to a new application that provides annual average daily traffic (AADT) for short-duration count stations. After restoring the original retrieval functions, the main purpose of this project, length-based classification, was implemented in a new tab named "LEN Class." All retrievals of classification queries were implemented using real-time computation, and the details were described in Chapter 3. During the project period, MnDOT also asked the University research team to integrate the RTMC speed data to detHealth_app. The "Volume" tab was first changed to "Volume/Speed" and speed-data retrieval functions were integrated in that tab. In addition, a new tab named "Bulk FHWA" was created to provide batch runs for producing a large amount of volume, speed, and classification data in TMG-2016 format. All new functionalities implemented in this project were detailed in Chapter 3.

The original proposal was to integrate RTMC length-based classification data into the existing volume database. However, this had to be changed because the database server was removed from the MnDOT TFA office due to the reasons described above. The title of the project remained the same but tasks were slightly modified, and the revised tasks are described in the next section.

1.2 REVISED PROPOSED WORK

The original project title was "Integrate RTMC Vehicle Classification to the Current Detector Volume Data" and no change was needed. The proposed work was comprised of four tasks in which Tasks 1 and 2 were slightly modified due to the reasons described in Section 1.1. The remaining tasks, Tasks 3 and 4, were allocated for writing the final report and unchanged. The only changes required were removal of the word "database" in the task descriptions since the original database server was removed. Due to such a minimal text change required, the change was resolved through a Technical Advisory Panel (TAP) meeting and not through an official contract amendment. The new version of tasks is shown below.

Task 1: Integrate RTMC Vehicle Classification Data to the Detector Volume and Health-Parameter Data

1.1 Extract the length-based classification from the RTMC IRIS server and integrate them into the current detector volume and health-parameter data.

1.2 Modify the current program and allow generation of past length-based classification data. Since this integration must be done for the detectors in the entire Twin Cities' freeway network, it is expected to require handling of a large amount of data and storage. With this integration, each detector will potentially have three types of traffic information: volume, classification, and detector-health parameters.

Task 2: Develop a Software Tool for Retrieval and Reporting of the RTMC Classification Data

2.1 Develop an easy-to-use software tool for data retrieval, analysis, and reporting applications that MnDOT TFA can use.

2.2 Define the types of data retrievals, applications, and reporting formats by working together with MnDOT TFA and then develop and deliver the desired software tool.

Task 3: Compile Report, Technical Advisory Panel (TAP) Review and Revisions

3.1 Prepare a draft report, following MnDOT publication guidelines, to document project activities, findings, and recommendations. This report will need to be reviewed by TAP, updated by the University to incorporate technical comments, and then approved by the technical liaison (TL) before this task is considered complete.

3.2 Schedule a TAP meeting to facilitate discussion of the draft report.

Task 4: Editorial Review and Publication of Final Report

4.1 During this task, the approved report will be processed by MnDOT's contract editors. The editorial review will ensure it meets publication standards. Work with the editor to address editorial comments, so this task can be finished while the contract is still active.

CHAPTER 2: CONVERSION TO SERVERLESS, STAND-ALONE DATA SYSTEM

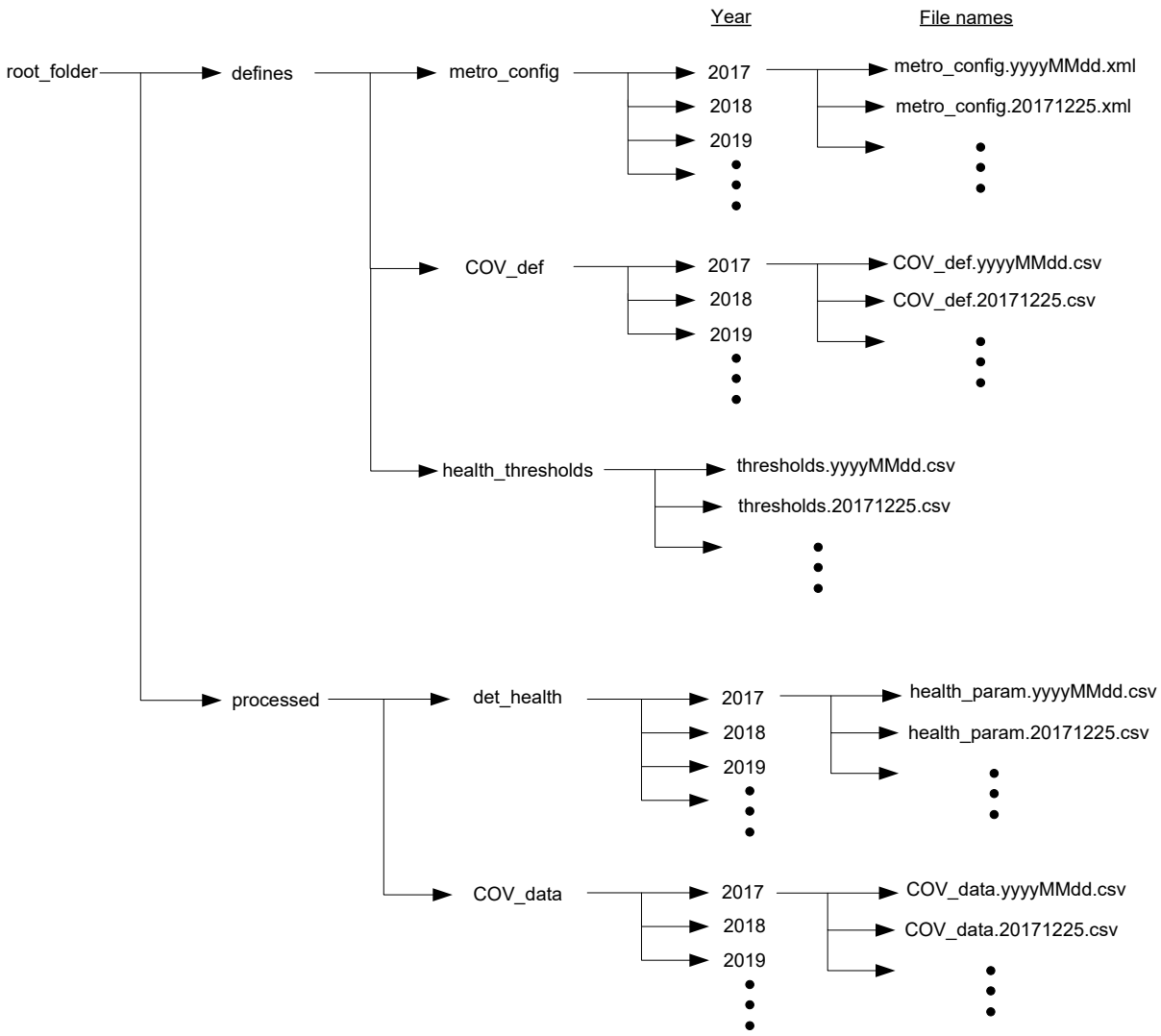
In the client/server system developed in the previous project, the server was managed by a software tool called *detHealth_daily* [5]. This tool downloads the raw traffic data from the IRIS managed by RTMC, computes detector-health parameters, and then loads them to the database. For the relational database engine, a free version of MySQL was used. In order to protect from catastrophic failures, *detHealth_daily* archived the health-parameters and volume data in CSV files on a network drive (NAS) and then loaded the same data to the database. Thus, if the database server experienced a non-recoverable failure, it would then be rebuilt using the archived CSV files in the network drive. Since the CSV files precisely match column-by-column with the tables in the database, rebuilding or restoring the database would be simple using MySQL scripts. For the conversion required in this project, this archived csv files were critical and served as the data source. The ultimate goal of this migration was that the user should not be able to tell the difference between the server-based and the stand-alone system, i.e. the user should not feel the difference of this conversion and migration. This chapter describes details on the software conversion and migration process.

2.1 DIRECTORY TREE

Figure 1 shows the directory tree that was used by the server software, *detHealth_daily*, for creating backup files, which is also used by the new versions of *detHealth_app*. In this structure, the root folder is divided into two folders, i.e., "*defines"* that houses all forms of defines files and "*processed*" that houses all processed results. The root folder is not fixed and can be selected or programmed by user using the "Settings" menu in *detHealth_app.* The default root folder was set to the following location:

"C:\Users\user_name\AppData\Roaming\Bulldog\detHealth_app"

It should be noted that processed and defines folders are divided into specific data types which are then further organized by year folders, except for the health_thresholds folder. The reason that the health_thresholds folder was not organized by year was that this folder contains only a small number of files and does not grow rapidly in the future. All file names are coded with an 8-character date stamp that includes year, month, and day in eight letters, i.e., "yyyyMMdd" format for easy recognition and search where "yyyy" is a four digit year, "MM" is a two digit month and "dd" is a two digit day between 01 and 31. This eight-character string representing a date stamp will be referred to as a date string in this report.



* If metro_config folder does not exist, the xml file downloaded is temporarily saved to g_AppPath

* The health_thresholds folder does not have year subfolders.

Figure 1: The root and processed/defines directory tree

2.2 DATA PROCESSING

The data processing algorithms developed in the previous project can be found in Chapter 2, and their implementation steps were documented in Chapter 3 of the final report [5]. The current project used the same algorithms, but implementation steps were modified to accommodate the server-less environment in which data must be self-generated by the application software. Below summarizes the steps involved in producing daily volume and detector health-parameters in the new *detHealth_app* developed in this project.

2.2.1 Download metro_config.xml (Step 1)

RTMC publishes information on their entire traffic detectors in a form of compressed xml file with a filename, *metro_config.xml.gz*, on their file-download site (IRIS server). This file is a well-structured xml file compressed in gzip, which is a utility commonly available under UNIX or Linux operating systems (OS's). When *detHealth_app* runs at the first time on a day, it downloads the *metro_config.xml.gz* file from the RTMC file-download site, uncompresses it, adds a date-stamp (date string) in the filename, and saves the file to a year folder in "~*root_folder**defines**metro_config*\". The filename conforms the following format:

metro_config.yyyyMMdd.xml

where yyyyMMdd is an eight-digit date string. The folder for storing this file can be set by user using the "Settings/Parameters" menu, and its default folder is created at "~\root_folder\defines\metro_config\". In the server implementation, this step occurs once daily and archives daily metro_config.xml files posted by RTMC. In the new *detHealth_app*, this step is need-based and only activated when the user runs the application, since most users would not need this file every day for the entire year.

2.2.2 Derive three equivalent detector sets for each r_node (Step 2)

After downloading the *metro_config.xml* file for a day, the program analyzes each corridor and r_nodes from the xml file to determine equivalent detector sets. For each r_node where its node type is Station, equivalent upstream and downstream stations with their volume relations described in Section 2.3 of the previous project final report [5] are derived. This result is saved as a CSV file with a date string attached as:

COV_def.yyyyMMdd.csv

This file is then saved in the folder "~\root_folder \defines\COV_def\" and the date string should match with the corresponding *metro_config. yyyyMMdd .xml* filename. The data processing is done by one corridor at a time, and the detailed algorithm and data formats can be found from the previous project report [5].

2.2.3 Compute all detector-health parameters (Step 3)

All detector-health parameters are computed using the raw volume and occupancy data queried and received from the IRIS server. For this computation, threshold values of the health-level classifier are first read in from the most recent define file. The threshold define files are stored in "~\root_folder\defines\health_thresholds\" and has a filename given by:

thresholds.yyyyMMdd.csv

The final detector-health data file is then created as a CSV formatted file that exactly matches with the columns defined in the corresponding database table. The filename follows the same date string convention with "health_param" prefix, which is shown below.

health_param.yyyyMMdd.csv

This file is save in the default folder, "~\root_folder\processed\det_health_param\", or in a user-defined folder. This CSV file is loaded when user presses the "Load" button in the Detectors tab of the detHealth_app program.

2.2.4 Apply COV rule-checks for applicable r_nodes (Step 4)

After completing computations of all detector-health parameters as described in Section 2.2.3, health level (HL) is first adjusted based on COV relations within a single r_node. The first character of COV_ap column represents the result of this step. There are two cases, when n_type is 'Station' but more than one detector per lane exist and when n_type is 'Entrance' where redundancy of detectors exists. The rules for using spatial relations in this step were described in the previous project final report [5].

2.2.5 Adjust HL according to COV tests for r_nodes with main-lane stations (Step 5)

In Step 4, COV was used within a single r_node when redundant detectors with equivalent traffic flow are available. In Step 5, the COV principle is applied in three equivalent traffic-flow stations in main lanes. The three equivalent stations are the current station, an equivalent upstream station, and an equivalent downstream station. How to add or subtract certain detector volumes to maintain an equivalency relation between the three stations was described in Section 2.4 of the previous-project final report [5], and the actual detector volume relations among associated detectors are defined in the corresponding "COV_def.yyyyMMdd.csv" file. The computed data according to "COV_def.yyyyMMdd.csv" is saved in a COV_data file with the same date string included in the COV_def filename, i.e.,

COV_data.yyyyMMdd.csv

This file is archived in "~\root_folder\processed\COV_data\" as default.

To test if the stations meet the requirement of COV rule, another intermediate data file is produced. For similarity measurements, normalized difference ratios called *diff_ratio* are computed. Detectors are considered healthy if the resulting *diff_ratio* is less than 0.05, and they are listed under the column heading *good_dets* [5]. For example, if *diff_ratio(up_vol, cur_vol)* is less than 0.05, all detectors involved in the current and upstream stations are considered to be candidates for HL upgrade. To save *diff_ratio* data, another intermediate CSV file is created and its filename starts with "COV_diffRatio" followed by a date string, i.e.,

COV_diffRatio.yyyyMMdd.csv

This file is stored in the "~\root_folder\processed\COV_data\" folder. The detectors listed under the column heading, *good_dets,* are the detectors of HL to be upgraded to H, since they satisfy the COV rule.

In order to keep track of which detectors are upgraded following the main lane COV rules, the detectors upgraded are listed in the **COV_upgradeDets.yyyyMMdd.csv** file in which each line is the state before upgrade. The detailed information on this file is described in [5].

The 5-step processing described above was implemented in the stand-alone *detHealth_app* program. It should be noted that the step that loads the produced detector-health parameters to the detector-health database is omitted, since the database is no longer used.

2.3 SPEED AND LENTH-BASED CLASSIFICATION

MnDOT RTMC provides 30-second volume and occupancy data for all detectors through their IRIS server. For the locations where Wavetronix radar detectors are installed, speed and classification data are additionally available to volume and occupancy. This section describes the raw form of these detector data.

2.3.1 RTMC detector data query

RTMC detector data are queried using hyperlinks, and the request string must conform the following format:

http://data.dot.state.mn.us:8080/trafdat/yyyy/yyyyMMdd/DetID.EXT

where yyyy is a four digit year string, yyyyMMdd is a date string, DetID is the detector ID, and EXT is an extension determined by the traffic data type. The data types related to previous and current projects are summarized in Table 1.

Table 1: Extensions for Different Traffic Data Types

Extension	Data Type Description
.v30	30-second volume
.c30	30-second occupancy
.s30	30-second speed
.vmc30	30-second motocycle volume --- length less than 8 feet
.vs30	30-second short vehicle volume --- length (8 – 20) feet
.vm30	30-second medium vehicle volume --- length (20 – 43) feet
.vl30	30-second long vehicle volume --- length longer than 43 feet

As an example, a hyperlink for querying 30-second speed data for "detector ID = 6908" on July 23, 2020 would look like:

http://data.dot.state.mn.us:8080/trafdat/2020/20200723/6908.s30

which would return the data in a binary format. Data can be retrieved in a JASON (JavaScript Object Notation) format for the same data by adding an extension "json". For example,

http://data.dot.state.mn.us:8080/trafdat/2020/20200723/6908.s30.json

The IRIS server only allows for requesting one detector, one type of data, and one day at a time.

2.3.2 Speed data

Speed data is available for the detector IDs assigned for Wavetronix radar sensors. The time resolution of speed data is 30 seconds, producing 120 data points per hour, 2,880 data points per day. The 30-second speed data is retrieved by ".s30" extension. The returned data quantity is 2,880 bytes, each byte representing the average speed of all vehicles passed through the detector in mph for the corresponding 30-second timeslot. In all implementations, JASON format was not used, instead binary formats were used to improve the bandwidth efficiency of the remote data retrievals.

2.3.3 Length-based classification data

Vehicle classification data also comes from Wavetronix radar sensors. The Wavetronix default classification scheme categorizes vehicles into four types based on the total vehicle length, which are:

- Motorcycles, less than 8 feet
- Short vehicles, 8 – 20 feet
- Medium vehicles, 20 – 43 feet
- Long vehicles, longer than 43 feet

To retrieve this classification data from the RTMC IRIS server, the file extension for each type shown in Table 1 is used. For example, data retrieval for motorcycles for “detector ID = 6908” on July 23, 2020 would need the following hyperlink.

http://data.dot.state.mn.us:8080/trafdat/2020/20200723/6908.vmc30

As before, the data can only be retrieved one-detector, one-type, and one-day at a time from the IRIS server. Consequently, it requires four requests to retrieve four vehicle types per detector for a single day. The time resolution of the returned data is 30 seconds, receiving 2,880 data points per data type and detector on a day. Data for each 30-second time slot represents population of the vehicle type matching the extension on the detector for 30 seconds. Since each data point is returned as a byte (8 bits), count of the matching vehicle type is ranged between 0 and 254. Number 255 (hex FF) is reserved for missing data.

CHAPTER 3: IMPLEMENTATION OF DATA TOOLS

This chapter describes implementation of detHealth_app related to this project. Present version includes six tabs providing different functionalities. Among them, tabs related to this project are Detectors, Volume/Speeds, LEN Class, and Bulk FHWA tabs. This chapter will focus on describing how they are implemented. Information on how to use them are available from the detHealth_app user manual.

3.1 DETECTORS TAB

After all data processing steps are completed as described in Section 2.2, the computed health parameters are stored in the folder tree shown in Figure 1. Since data is no longer available from a central database, retrieval utilizes the CSV files in the folder tree. In the server implementation, health parameters for the whole RTMC detectors were daily computed and then stored in the database to allow any combinational query for retrievals. This would be unnecessary and too much for the stand-alone program because it is only used by an individual on his or her own PC. A new strategy implemented was not to daily compute health parameters but only based on demand-by-user, i.e., detector health parameters are produced only if user requests and if the data requested was not available on the user PC. The final GUI is nearly identical to the client software of the client/server model, and a screen capture is shown in Figure 2. Notice that only two more buttons were added in the options of loading the data, which are explained next.

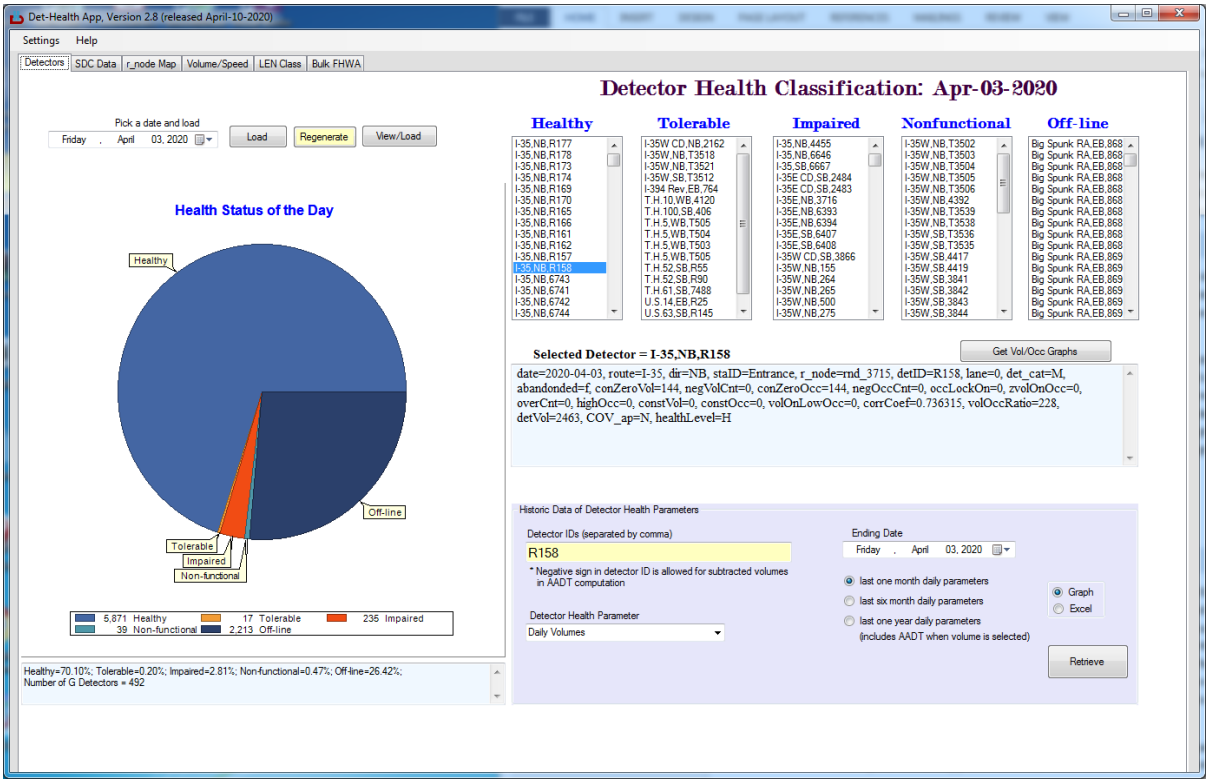


Figure 2: An example screen of detector health classifier and parameter retrieval functions

Users can still pick any past date from the calendar function and press the **Load** button. However, since past data are not saved for every day in the stand-alone version, the Load function first searches and figures out if the data for the user-selected date already exists or not. If the data does not exist in the folder tree, the software will then ask if generation of data for the selected date can be started or not ("Yes" or "No" choice in a dialog box). Selecting "Yes" will generate the data followed by loading the produced data, creating a GUI similar to Figure 2. Generation of detector health parameters for a single day for the entire RTMC detectors (about 7,800) takes about 6 to 10 minutes, depending on the Internet traffic and load conditions of the IRIS server. If data for the user-selected date already exists in the folder tree, loading it from would feel like an instant, taking less than a second.

To accommodate information on what dates of data are available in the local file system, a new button called "**View/Load**" was added. Pressing this button opens the "\health_param" folder which is organized by year. Figure 3 shows an example that contains three days of data in the year 2020 folder. Selecting a filename that contains the desired date string and pressing the Open button will load the data, displaying a screen similar to Figure 2. If no file with the desired date string exists, it means that the data for the date does not exist in the file system, and it must be generated by selecting the date from the calendar tool and then pressing the "Load" button. This "View/Load" function is new and added for the stand-alone version.

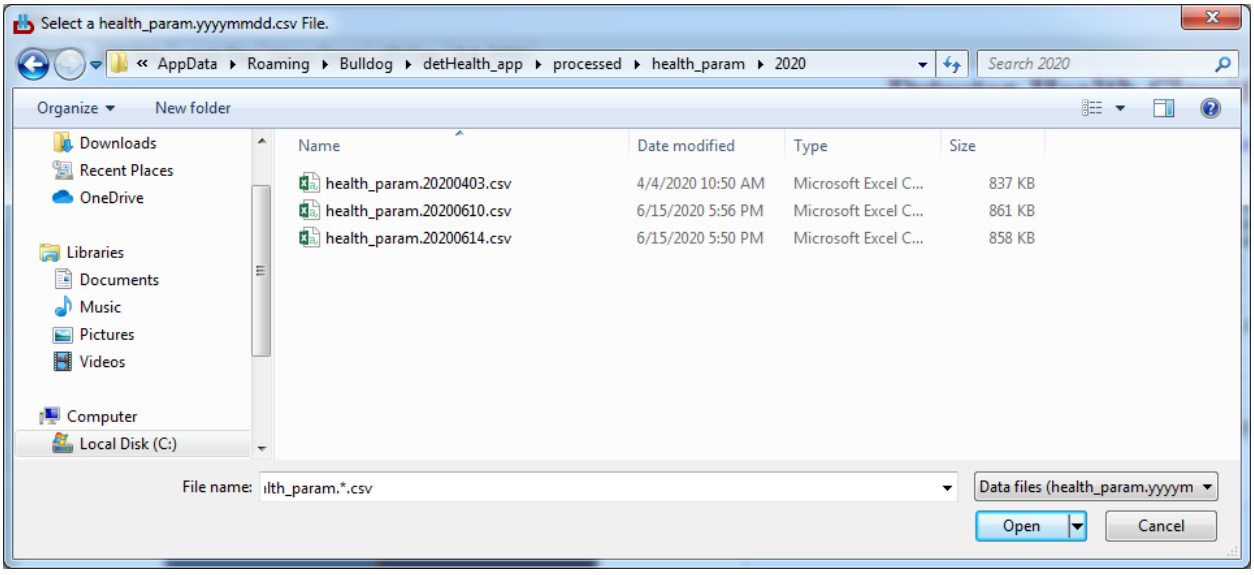


Figure 3: An example of health_param year directory

Another function included is a regenerate function implemented by the "**Regenerate**" button. This function allows regeneration of existing data for the selected date. It was necessary to regenerate and refresh the existing data when RTMC corrects or adds the raw detector data.

The purple group box in the lower right corner in Figure 2 provides retrieval of historical data of the health parameters for the detector IDs listed in the textbox labeled "Detector IDs". This function was implemented using a real-time computation of detector health parameters from raw volume and occupancy data retrieved from the RTMC IRIS server.

3.2 VOLUME/SPEED TAB

Traffic volume and speed data are critically important and frequently used in traffic monitoring and other transportation applications. The "Volume/Speed" tab shown in Figure 4 provides retrieval of both data in various formats and options. In a raw form, RTMC volume data are records of number of vehicles per every 30 seconds. For MnDOT TFA, hourly or daily volumes are more useful than the raw 30-second volumes. Consequently, retrieval of hourly and daily volume group boxes are provided with choices of hourly and daily traffic volumes. The data retrieval period is always defined by the number of past dates from the ending date of the retrieval period. If a pair of 30-second raw volume and occupancy data is needed, the "Get Vol/Occ Graphs" button in the "Detectors" tab can be used, which provides line graphs and scatter plots.

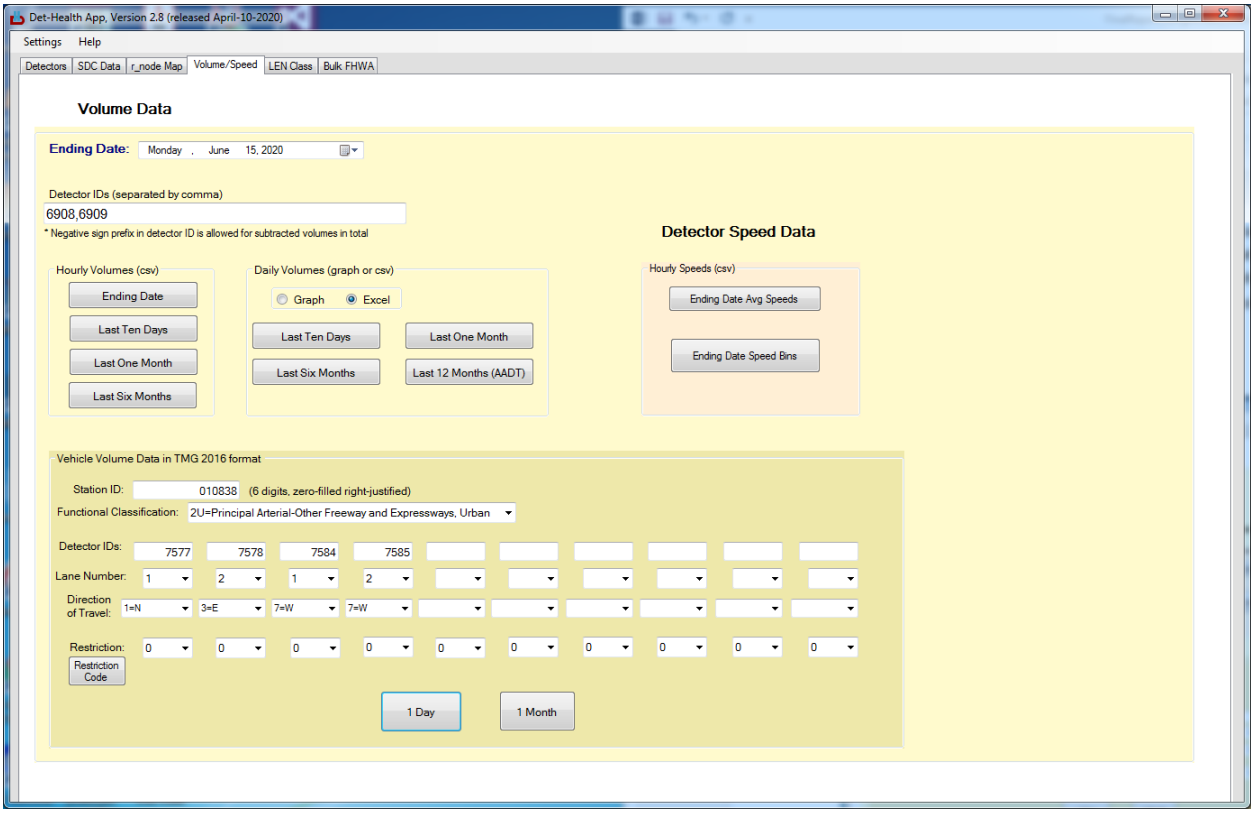


Figure 4: GUI of "Volume/Speed" tab

The user selectable retrieval options for hourly volumes are single day (ending date), last ten days, last one month, and last six months from the ending date. The output data is formatted in CSV and then directly loaded as an Excel spreadsheet, from which user can save the file with a user preferred filename and directory. Table 2 shows an example of hourly volume data for one day loaded to Excel. The columns comprise date, hour, hourly volumes of detectors, total volume (hourly total of all detector

volumes in that row), missing-data percent of each detector within the corresponding one-hour time slot. The missing percent column provides information on quality of the data in each hour.

Retrieval options of daily traffic volumes include last ten days, last one month, and last 12 months from the ending date. In daily CSV data, only difference is absence of the hourly column and then addition of the total miss % column. Table 3 shows an example of daily volumes retrieved for past 10-days from the ending date 6/15/2020. If the "Excel" option is selected and then the "Last 12 Months (AADT)" button is pressed, computed AADT (average-of-average, often called AASHTO method [1-3]) is computed and appended at the last row of the CSV file.

Table 2: Hourly Volumes For Detectors 6908 And 6909 On June 15, 2020

date	hour	6908	6909	Total Vol	6908-mis%	6909-mis%
6/15/2020	0	109	130	239	0	0
6/15/2020	1	63	86	149	0	0
6/15/2020	2	58	68	126	0	0
6/15/2020	3	68	83	151	0	0
6/15/2020	4	137	130	267	0	0
6/15/2020	5	359	421	780	0	0
6/15/2020	6	582	790	1372	0	0
6/15/2020	7	745	1005	1750	0	0
6/15/2020	8	840	1024	1864	0	0
6/15/2020	9	852	958	1810	0	0
6/15/2020	10	872	1056	1928	0	0
6/15/2020	11	1012	1134	2146	0	0
6/15/2020	12	1084	1225	2309	0	0
6/15/2020	13	1034	1221	2255	0	0
6/15/2020	14	1169	1429	2598	0	0
6/15/2020	15	1368	1839	3207	0	0
6/15/2020	16	1367	1786	3153	0	0
6/15/2020	17	1242	1625	2867	0	0
6/15/2020	18	916	1195	2111	0	0
6/15/2020	19	671	817	1488	0	0
6/15/2020	20	501	600	1101	0	0
6/15/2020	21	422	479	901	0	0
6/15/2020	22	300	349	649	0	0
6/15/2020	23	187	233	420	0	0

Table 3: Daily Volumes For Detectors 6908 and 6909 On June 6-15, 2020

date	6908	6909	total	6908-mis%	6909-mis%	total-mis%
6/6/2020	12243	15406	27649	0	0	0
6/7/2020	11139	13710	24849	0	0	0
6/8/2020	15420	19515	34935	0	0	0
6/9/2020	15064	18516	33580	0	0	0
6/10/2020	14702	19596	34298	0	0	0
6/11/2020	15843	20426	36269	0	0	0
6/12/2020	16598	20816	37414	0	0	0
6/13/2020	13291	16610	29901	0	0	0
6/14/2020	11681	14315	25996	0	0	0
6/15/2020	15958	19683	35641	0	0	0

Retrieval of daily volume data includes an option to generate its output as a graph. Graphs often provide effectiveness in inspection of data such as spotting of an abnormal traffic pattern. Figure 5 shows a graph of daily volumes for detectors 6908 and 6909 for one year from the ending date 6/15/2020. Please note that both detector had 100% missing near July 21, 2019 and then a significant drop of volumes starting March 28, 2020, at which the Minnesota governor (Tim Walz) issued a stay-at-home order to reduce spread of COVID-19. Notice that the traffic volume dropped abruptly with the order and then gradually recovered. In the graph, AADT of the two detectors (combined) is provided in the graph title (Figure 5), which is 35,013. The line graphs display daily volumes in the top plot area while missing percent data are displayed in the bottom plot area. Lines can be removed from graph by unchecking the item in the graph legend. The Edit button provides many editing functions of the graph. For example, data points can also be removed by unchecking the Visible option in the Points tab.

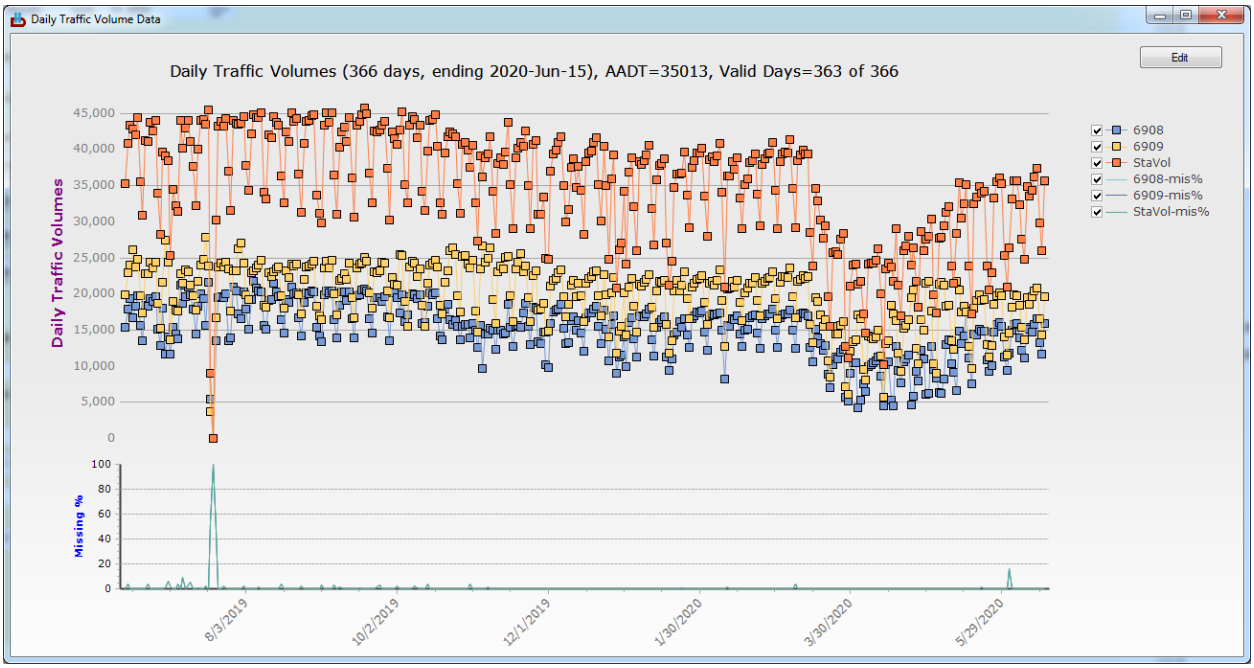


Figure 5: Plot of past 12 months from ending date 6/15/2020 for detectors 6908 and 6909

Integration of speed data was not originally included in the proposal of this project. However, TFA later learned that RTMC actually provides not only length-based vehicle classifications but also speed data online from the Wavetronix radar sensors. Thus, TFA wanted this speed data to be integrated in detHealth_app as well. Since the same user entries as the volume data are required, the speed data functions were conveniently added to the "Volume" tab using a groupbox labeled "Hourly Speeds (csv)" and then the tab name was changed to "Volume/Speed" (Figure 4). It should be noted that RTMC speed data is not binned data but an average within 30-second time slots, i.e., an average of all vehicle speeds in that 30-second time slot on the lane the detector is located. This raw speed data is retrieved and converted to hourly data when the button labeled "Ending Date Avg Speeds" is pressed. An example of hourly average speed data is shown in Table 4. The speed of each hour is computed by averaging the speeds of the 120 30-second timeslots. The column labeled "Avg Spd" provides an average speed of all lanes (detectors) in that hour. Missing data information is provided similarly to that of volume data.

Table 4: Hourly Average Speed Data for Detectors 6908 and 6909 on June 15, 2020

date	hour	6908	6909	Avg Spd	6908-mis%	6909-mis%
6/15/2020	0	58	63	60	38.3	37.5
6/15/2020	1	59	64	62	58.3	48.3
6/15/2020	2	60	64	62	60	56.7
6/15/2020	3	60	63	62	55.8	53.3
6/15/2020	4	60	64	62	30.8	34.2
6/15/2020	5	63	67	65	3.3	5.8
6/15/2020	6	62	67	64	0	0
6/15/2020	7	60	65	62	0.8	0
6/15/2020	8	57	65	61	0	0
6/15/2020	9	57	65	61	0	0
6/15/2020	10	56	64	60	0	0
6/15/2020	11	57	64	60	0	0
6/15/2020	12	56	64	60	0	0
6/15/2020	13	57	63	60	0	0
6/15/2020	14	56	62	59	0	0
6/15/2020	15	49	54	52	0	0
6/15/2020	16	44	48	46	0	0
6/15/2020	17	57	62	60	0	0
6/15/2020	18	59	66	62	0	0
6/15/2020	19	60	68	64	0	0
6/15/2020	20	59	67	63	0.8	0
6/15/2020	21	57	65	61	3.3	1.7
6/15/2020	22	57	64	60	3.3	5.8
6/15/2020	23	57	64	60	18.3	13.3

The standard speed data format accepted by FHWA TMAS is binned data, i.e., a histogram. Since RTMC speed data is not binned as described above, the closest binned data that can be estimated is binning the volume of each 30-second time slot using its 30-second average speed. For example, if the average speed and volume of a 30-second timeslot were recorded at 65mph and 13, respectively, then the volume count 13 can be added to the bin that belongs to 65mph. This estimation assumes that the speed variations within the 30-second time slot is within the range of the corresponding bin limits. Pressing the "Ending Date Speed Bins" button produces this estimated binned data, and an example is shown in Figure 6. The columns comprise detector ID, Date, Hour, and 22 speed bins. The bins are not programmable and spaced at 5mph except for the first and last bins. It starts from the range [0, 20) mph, and then subsequent bins are incremented by a 5mph interval, i.e., [20, 25) mph, [25, 30) mph, etc. The 22nd bin is the last bin and it is allocated for the speed range [120, ∞) mph. The precise bin

ranges are specified in Table 6. According to Figure 6 and many examples reviewed, speed distributions computed using 30-second averages looked very similar to those distributions computed using individual vehicle speeds. It appears to support the assumption that speed variations within the corresponding 30-second time slot mostly stays within the range of the corresponding bin, although a ground truth verification would be needed.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	detID	Date	Hour	0->20	20->25	25->30	30->35	35->40	40->45	45->50	50->55	55->60	60->65	65->70	70->75	75->80	80->85	85->90	90->95	95->100	100->105	105->110	110->115	115->120	120->	Total
2	6908	6/15/2020	0	0	0	0	0	0	2	3	18	53	26	5	1	0	1	0	0	0	0	0	0	0	0	109
3	6908	6/15/2020	1	0	0	0	0	0	0	2	14	18	18	7	3	1	0	0	0	0	0	0	0	0	0	63
4	6908	6/15/2020	2	0	0	0	0	0	0	2	7	19	20	7	3	0	0	0	0	0	0	0	0	0	0	58
5	6908	6/15/2020	3	1	0	0	0	0	0	1	12	19	24	10	1	0	0	0	0	0	0	0	0	0	0	68
6	6908	6/15/2020	4	0	0	0	0	0	0	0	12	47	58	17	1	2	0	0	0	0	0	0	0	0	0	137
7	6908	6/15/2020	5	0	0	0	0	0	0	0	5	44	189	106	14	1	0	0	0	0	0	0	0	0	0	359
8	6908	6/15/2020	6	0	0	0	0	0	0	6	0	102	313	154	7	0	0	0	0	0	0	0	0	0	0	582
9	6908	6/15/2020	7	0	0	0	0	0	0	0	15	416	296	14	2	2	0	0	0	0	0	0	0	0	0	745
10	6908	6/15/2020	8	0	0	0	0	0	0	0	156	552	132	0	0	0	0	0	0	0	0	0	0	0	0	840
11	6908	6/15/2020	9	0	0	0	0	0	0	10	203	522	117	0	0	0	0	0	0	0	0	0	0	0	0	852
12	6908	6/15/2020	10	0	0	0	0	0	0	9	232	584	47	0	0	0	0	0	0	0	0	0	0	0	0	872
13	6908	6/15/2020	11	0	0	0	0	0	0	12	192	727	77	4	0	0	0	0	0	0	0	0	0	0	0	1012
14	6908	6/15/2020	12	0	0	0	0	0	0	23	236	701	124	0	0	0	0	0	0	0	0	0	0	0	0	1084
15	6908	6/15/2020	13	0	0	0	0	0	12	13	213	661	127	8	0	0	0	0	0	0	0	0	0	0	0	1034
16	6908	6/15/2020	14	0	0	0	0	18	9	56	238	744	104	0	0	0	0	0	0	0	0	0	0	0	0	1169
17	6908	6/15/2020	15	24	19	7	65	129	99	204	421	336	57	7	0	0	0	0	0	0	0	0	0	0	0	1368
18	6908	6/15/2020	16	0	29	103	128	152	257	235	165	263	35	0	0	0	0	0	0	0	0	0	0	0	0	1367
19	6908	6/15/2020	17	0	0	0	10	31	13	59	166	694	269	0	0	0	0	0	0	0	0	0	0	0	0	1242
20	6908	6/15/2020	18	0	0	0	0	0	3	0	118	459	320	16	0	0	0	0	0	0	0	0	0	0	0	916
21	6908	6/15/2020	19	0	0	0	0	0	0	0	10	278	339	43	1	0	0	0	0	0	0	0	0	0	0	671
22	6908	6/15/2020	20	0	0	0	0	1	0	0	30	259	198	13	0	0	0	0	0	0	0	0	0	0	0	501
23	6908	6/15/2020	21	0	0	0	0	2	1	0	112	197	85	24	0	0	1	0	0	0	0	0	0	0	0	422
24	6908	6/15/2020	22	1	0	0	0	0	0	7	57	182	49	4	0	0	0	0	0	0	0	0	0	0	0	300
25	6908	6/15/2020	23	0	0	0	0	0	0	6	26	116	31	8	0	0	0	0	0	0	0	0	0	0	0	187

Figure 6: Screen capture of speed bin data loaded on Excel for detectors 6908 and 6909 on June 15, 2020

Volume data may be produced in an FHWA VOL format that conforms TMG 2016. The details of the VOL format that detHealth_app generates is provided in Appendix A. This operation is controlled using the groupBox labeled “Vehicle Volume Data in TMG 2016 format” and it is shown in Figure 7. Data may be generated for one day from the Ending Date (date picker) or one month from the ending date. The user is also required to enter Functional Classification of the road, lane number, and direction of travel. Figure 8 shows one-day TMG2016 VOL data generated for four detector entries under Station ID 010838. The output is a text file, and one detector occupies one line per day. FHWA format requires entries of hourly volumes, and each line represents a single day entry for a single lane (detector) that comprises 24 hourly volumes. It should be noted that this utility is designed for retrieving data for one station at a time. If data for a large number of stations are needed, the “Bulk FHWA” tab should be used.

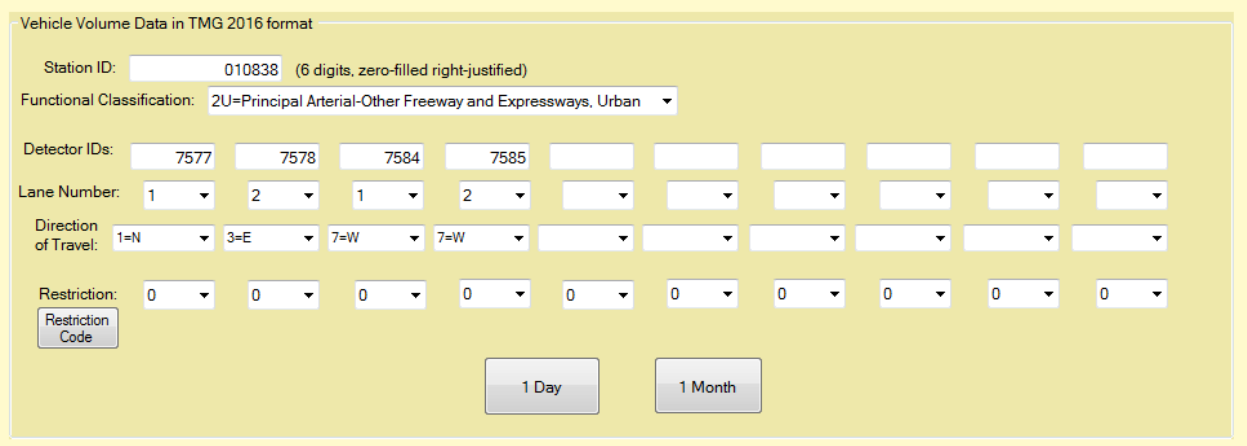


Figure 7: Interface used for generating FHWA VOL data

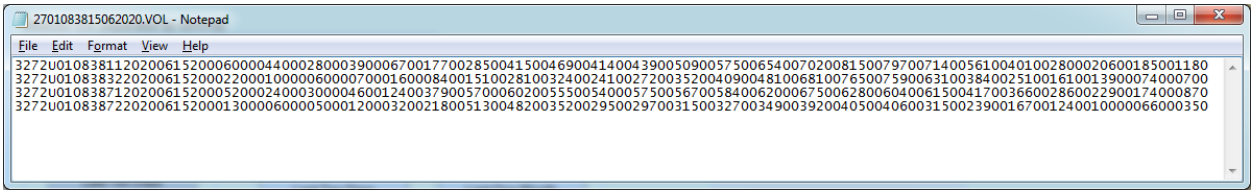


Figure 8: FHWA VOL data produced for a single day using Figure 7 entry.

3.3 LEN CLASS TAB

RTMC provides length-based vehicle classification through its IRIS server for the locations where Wavetronix radar sensors are installed. This data consists of volumes of four vehicle classes (see Section 2.3.3 for details) in 30-second time intervals.

To implement length-based classification in detHealth_app, a new tab named “LEN Class” was added, which is shown in Figure 9. The tab name includes “LEN” to indicate that it provides the TMG “LEN” formatted data, accepted by FHWA TMAS. This utility was designed to generate classification data in two formats, CSV and FHWA. The choice of hourly or daily time intervals for CSV outputs was implemented using two separate groupBoxes, i.e., “Hourly Classification (csv)” and “Daily Classification (csv),” respectively.

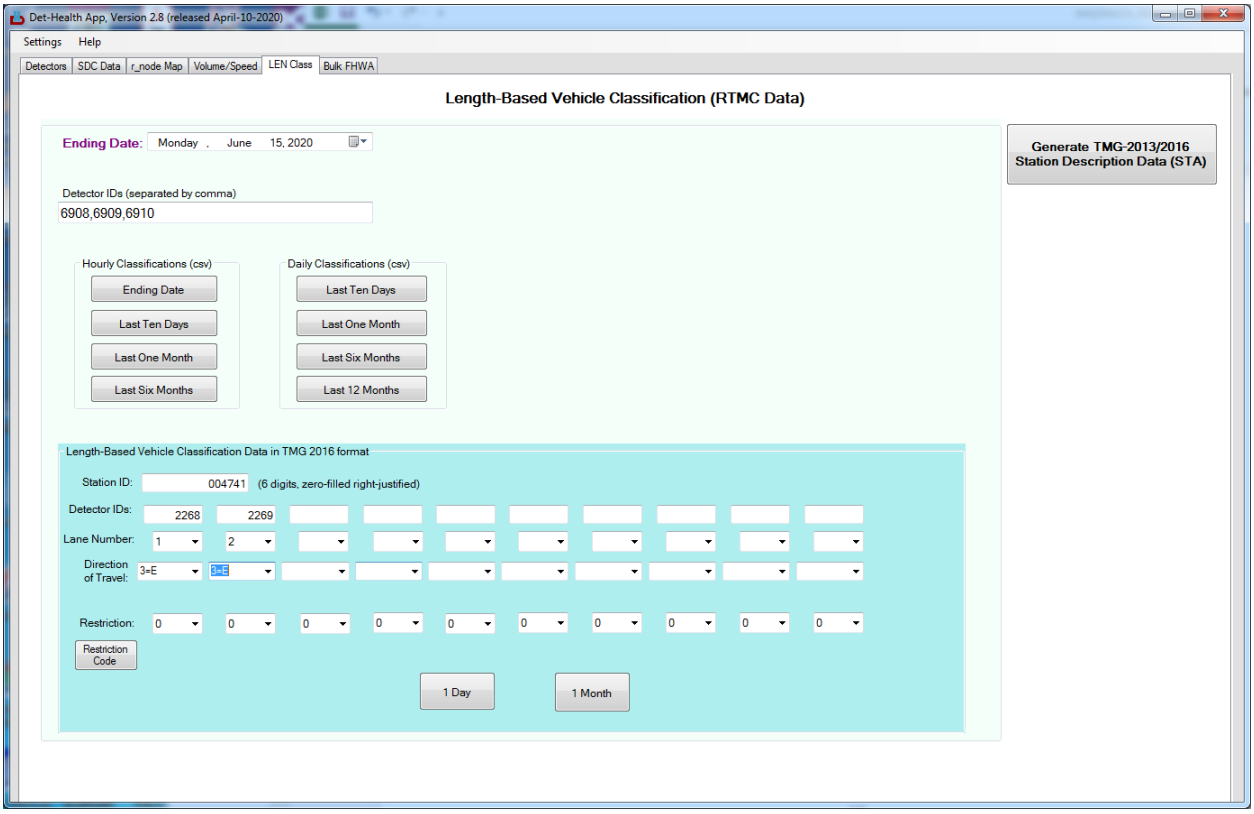


Figure 9: GUI of the "LEN Class" tab

For both hourly and daily CSV files, the columns have the same format, except that hourly CSV includes one more column labeled "hour". Each row in hourly CSV is the class volume of the corresponding hour of the day while each row in daily CSV is the class volume of the corresponding day. The columns start with "date" followed by "hour" for hourly CSV (none for daily CSV), and then each detector occupies six columns. The format is defined using the following string.

date, hour, (detID-mot, detID-sho, detID-med, detID-lng, detID-vol, detID-mis%)

where the parenthesis repeats for each detector, *detID* denotes detector ID and the first four columns are volumes of each type (motorcycle, short, medium, and long), followed by detID-vol that is the total volume of all types. The *detID-mis%* columns denote the percent of data missing in that time interval for the lane (detector). For example in hourly data, if *detID-mis%* column is 6.8 then it means 6.8 percent of all data combined in that hour is missing. Figures 10 and 11 show examples of hourly and daily classification CSV files loaded to Excel, 24 hours and 10 days for detector IDs 6908 and 6909, respectively.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	date	hour	6908-mot	6908-sho	6908-med	6908-lng	6908-vol	6908-mis%	6909-mot	6909-sho	6909-med	6909-lng	6909-vol	6909-mis%
2	6/15/2020	0	0	92	7	10	109	0	0	117	11	2	130	0
3	6/15/2020	1	0	50	4	9	63	0	1	66	10	9	86	0
4	6/15/2020	2	0	40	9	9	58	0	0	50	10	8	68	0
5	6/15/2020	3	1	39	7	21	68	0	0	55	10	18	83	0
6	6/15/2020	4	1	90	11	35	137	0	0	82	29	19	130	0
7	6/15/2020	5	0	235	78	46	359	0	0	281	105	35	421	0
8	6/15/2020	6	1	358	143	80	582	0	0	482	245	63	790	0
9	6/15/2020	7	1	409	217	118	745	0	0	664	263	78	1005	0
10	6/15/2020	8	2	470	239	129	840	0	0	689	258	77	1024	0
11	6/15/2020	9	0	496	233	123	852	0	1	596	267	94	958	0
12	6/15/2020	10	2	516	209	145	872	0	3	668	275	110	1056	0
13	6/15/2020	11	1	597	274	140	1012	0	0	728	284	122	1134	0
14	6/15/2020	12	1	643	273	167	1084	0	2	797	290	136	1225	0
15	6/15/2020	13	0	627	256	151	1034	0	0	761	341	119	1221	0
16	6/15/2020	14	1	660	340	168	1169	0	1	914	393	121	1429	0
17	6/15/2020	15	1	814	422	131	1368	0	0	1261	491	87	1839	0
18	6/15/2020	16	8	840	396	123	1367	0	2	1239	457	88	1786	0
19	6/15/2020	17	0	822	321	99	1242	0	0	1221	348	56	1625	0
20	6/15/2020	18	1	681	185	49	916	0	1	884	260	50	1195	0
21	6/15/2020	19	0	501	122	48	671	0	0	611	174	32	817	0
22	6/15/2020	20	1	400	57	43	501	0	0	451	115	34	600	0
23	6/15/2020	21	0	347	49	26	422	0	0	377	73	29	479	0
24	6/15/2020	22	0	250	16	34	300	0	0	274	49	26	349	0
25	6/15/2020	23	0	164	10	13	187	0	0	173	42	18	233	0

Figure 10: Hourly classification data loaded to Excel for detectors 6908 and 6909 on June 15, 2020

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	date	6908-mot	6908-sho	6908-med	6908-lng	6908-vol	6908-mis%	6909-mot	6909-sho	6909-med	6909-lng	6909-vol	6909-mis%
2	6/6/2020	13	9318	2307	605	12243	0	8	11459	3458	481	15406	0
3	6/7/2020	13	8834	1844	448	11139	0	4	10451	2936	319	13710	0
4	6/8/2020	19	9666	3922	1813	15420	0	12	13260	4889	1354	19515	0
5	6/9/2020	19	9510	3711	1824	15064	0	5	12156	4983	1372	18516	0
6	6/10/2020	14	9588	3505	1595	14702	0	10	13351	4952	1283	19596	0
7	6/11/2020	46	9767	4173	1857	15843	0	16	13647	5241	1522	20426	0
8	6/12/2020	30	10526	4219	1823	16598	0	14	14406	5134	1262	20816	0
9	6/13/2020	13	10202	2440	636	13291	0	11	12736	3330	533	16610	0
10	6/14/2020	5	9376	1791	509	11681	0	4	11069	2860	382	14315	0
11	6/15/2020	22	10141	3878	1917	15958	0	11	13441	4800	1431	19683	0

Figure 11: Daily classification data loaded to Excel for detectors 6908 and 6909 on June 6-15, 2020

In addition to CSV formatted data, the “LEN Class” tab includes a function that outputs the data in TMG 2016 LEN format. Figure 12 shows the final groupBox implementation of this utility, which is located at the lower half portion of the “LEN Class” tab. To generate the FHWA formatted data, the user must supply information on station ID, detector IDs, lane numbers of each detector, traffic direction of travel on each detector, and a restriction code. A station with up to 10 lanes can be entered. If each direction is separately generated, it would allow generation of data up to 20 lanes at a time. The duration of the data period can be either one day or one month from the ending date. The final LEN formatted data is a text file, and an example of one-day data loaded in Notepad is shown in Figure 13. Note that each line of the LEN format represents class volumes of the corresponding hour and detector. Since this station has

only two detectors, a single day data consists of 48 rows. The details of the FHWA LEN format implemented by detHealth_app is provided in Appendix B.

It should be noted that this utility was not designed for generating a large amount of data for many stations in TMG-2016 LEN format but only for exploring one station at a time. For generating data for many stations, utilities in the “Bulk FHWA” tab should be used.

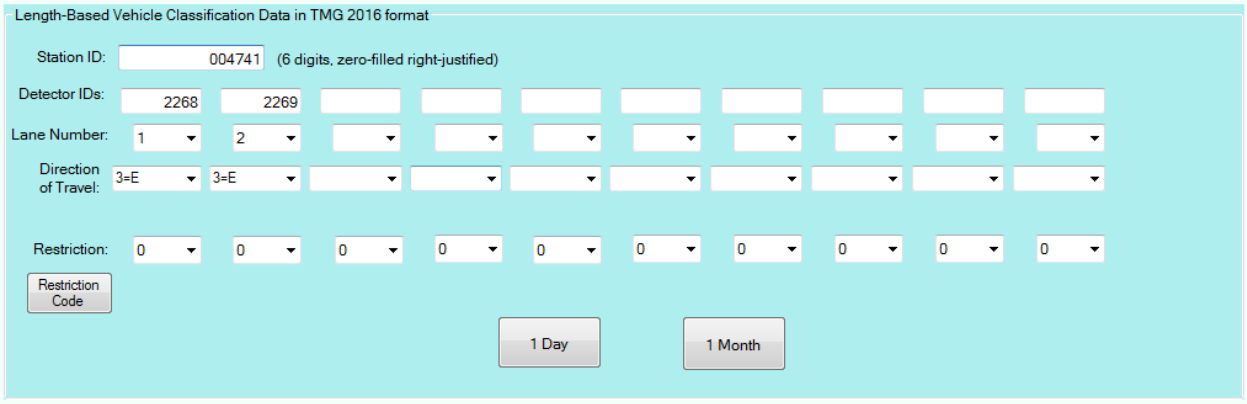


Figure 12: GUI for retrieving length-based classification in TMG .LEN file

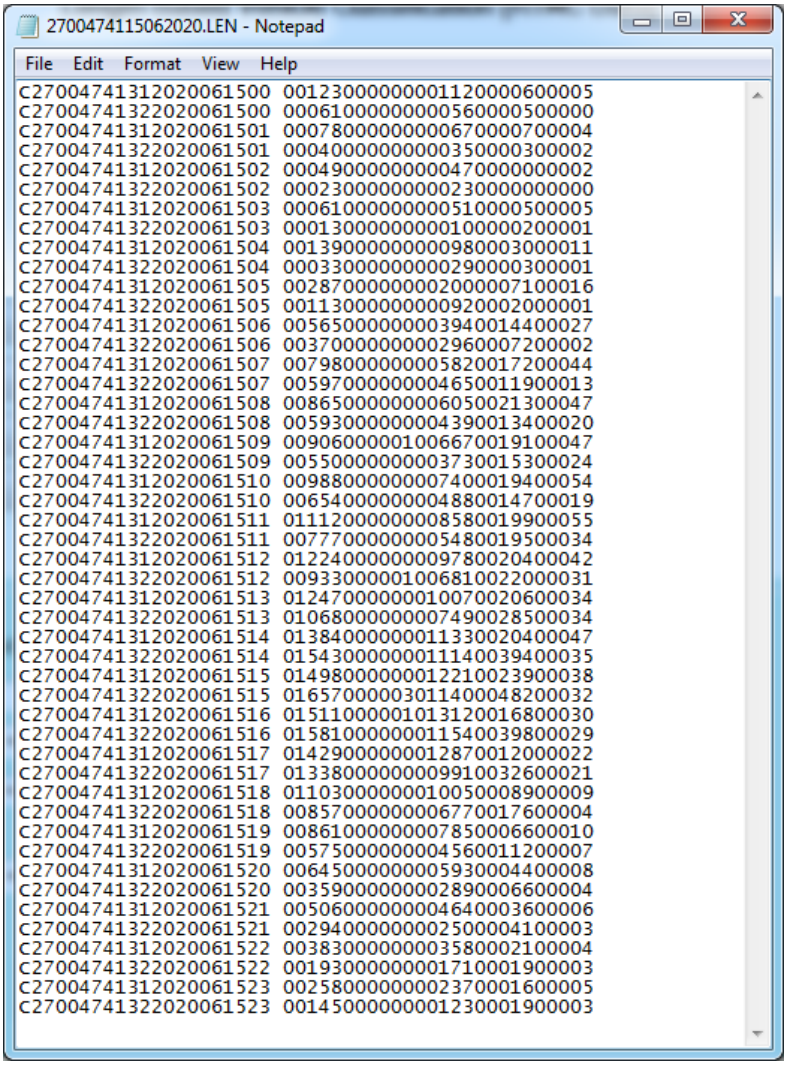


Figure 13: Content of .LEN file for a single day for two detectors loaded in Notepad

3.4 BULK FHWA TAB

This tab was designed for retrieving a large amount of data for many stations (or sequence numbers) in TMG-2016 format, i.e., for producing a large amount of data for many stations with a single click. Figure 14 shows the GUI of the "Bulk FHWA" tab. The top groupBox is used for generating TMG LEN data, the middle groupBox is used for TMG SPD data, and the bottom groupBox is used for TMG VOL data.

To use this utility, the first step required is to create a station define file or select an existing one if it was already created. It is a text file, and the columns of LEN and SPD define files are shown in Table 5. The VOL define files have a slightly different format, and it is described later.

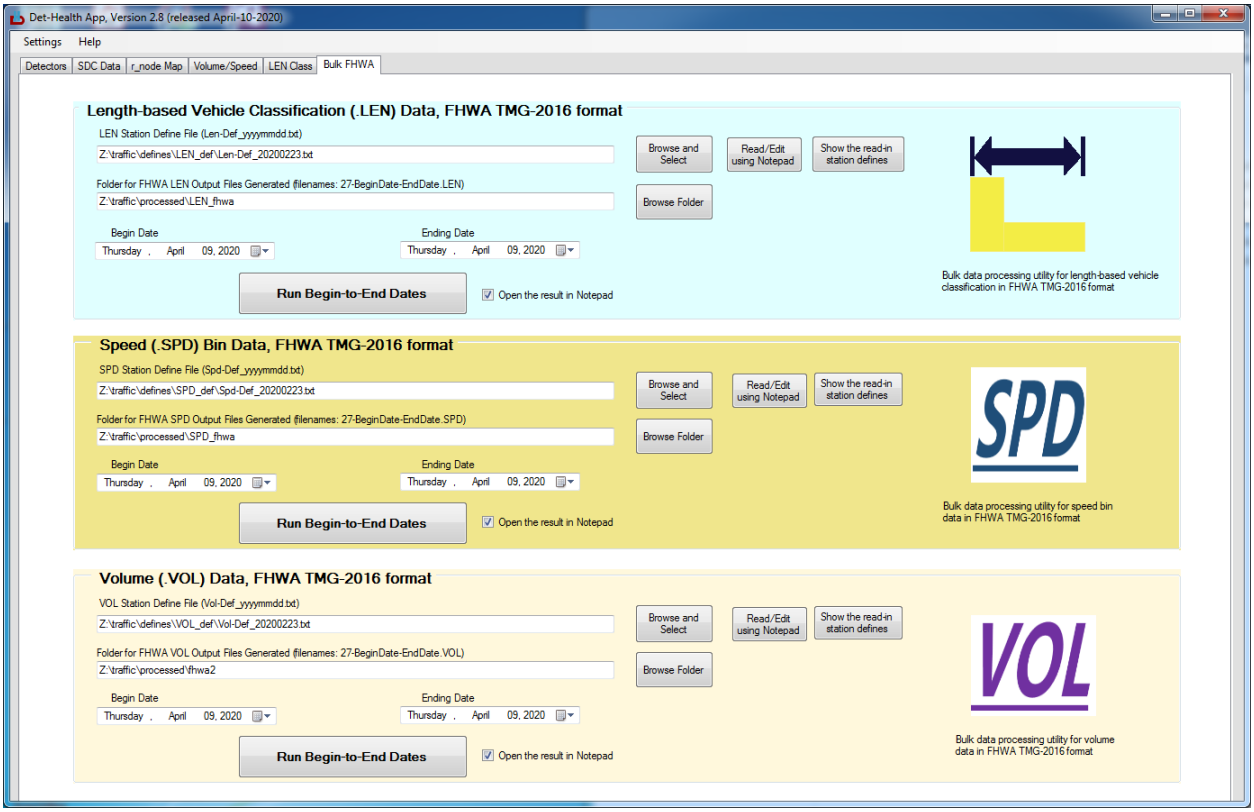


Figure 14: GUI of "Bulk FHWA" tab

Table 5: Column Formats For Station Definition Files Of LEN And SPD Data

Column name	Description
Station_ID	Station ID not exceeding 6 digits
Direction_Code	1=N, 2=NE, 3=E, 4=SE, 5=S, 6=SW, 7=W, 8=NW
City	T=Twin Cities, R=Rochester
P	Primary (all detectors are primary for the current version)
(detector IDs)	List of detector IDs separated by comma
lanes	A separating word used to indicate lane numbers. Case not sensirive
(lane numbers)	Lane numbers corresponding to the detector IDs in the same order
End	End of a record word. Comments can be added after this field.

Filenames for station-define files have a different prefix for each type for clear distinction. The user is allowed to change only the * portion of the filename as shown below.

Len-Def*.txt	for LEN station define file
Spd-Def*.txt	for SPD station define file
Vol-Def*.txt	for VOL station define file

For the wildcard field *, any string can be used but date of the station defines created is recommended for tracking the creation date. An example of LEN station define file is shown in Figure 15 which has a filename “Len-Def_20200223.txt” denoting that it was created on February 23, 2020. The top commented portion (lines starting with “;”) of the text describes the details of the format.

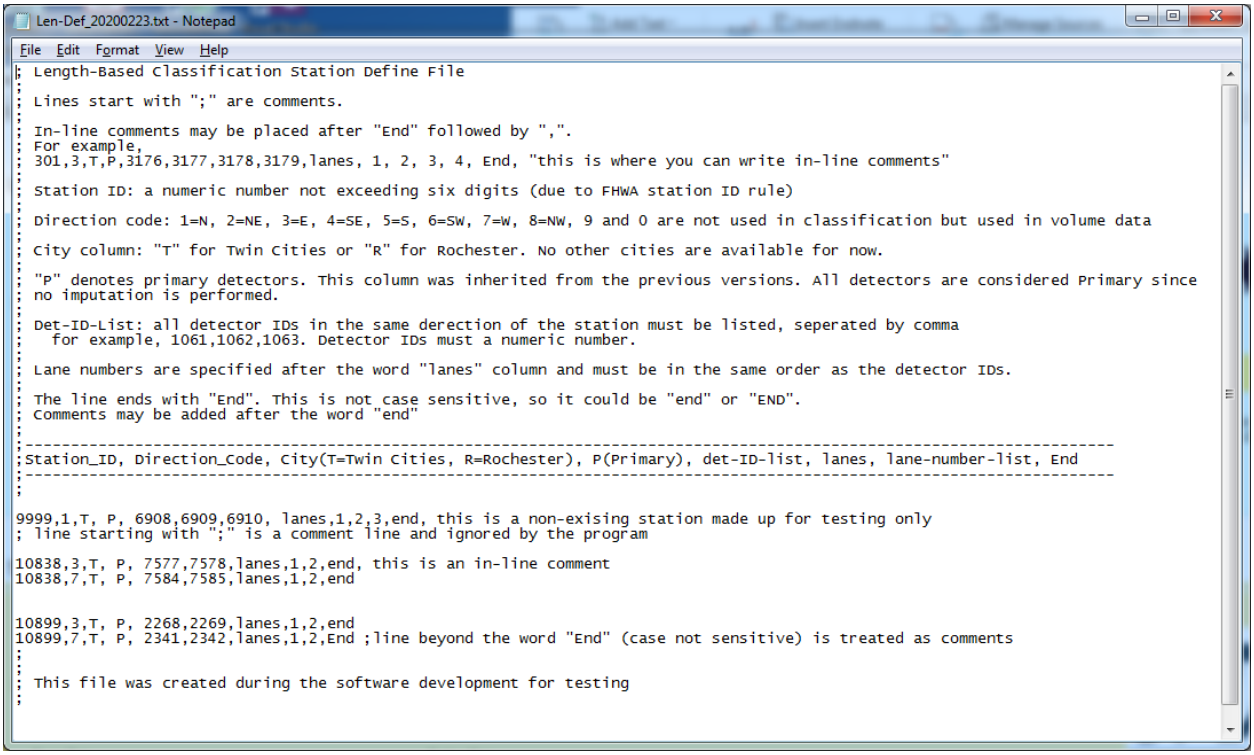


Figure 15: An example of LEN station define file

FIGURE 16: Three buttons used to select, edit, and verify LEN station define file.

For managing station define files, three buttons are used, which are described below. The screen segment of GUI is shown in Figure 16.

"Browse and Select" button: This button is used to browse and select a prepared station define file using a file select dialog. Once a define file is selected, its path is displayed in the textbox left.

"Read/Edit using Notepad" button: Pressing this button loads the station define file on the left textbox to Notepad. Once it is loaded in Notepad, the station defines can be reviewed, edited and saved. This button was created to provide a simple means of editing station defines.

"Show the read-in station defines" button: Pressing this button shows how the program actually reads in the station defines by displaying the read-in result in a Notepad. This utility is used to verify that the program recognizes and reads all stations correctly. Figure 17 shows an example screen. Note that the format is different from the define files and the first line of the text shows which define file was read in, followed by one blank line. Next, each directional station definition is shown, but displaying the lane number inside a parenthesis next to its detector ID. Comments and blank lines are not displayed

because they are not read-in by the program. The line ends with the word “End”. The format was intentionally made different from a real define file to differentiate that this is a read-in define file.

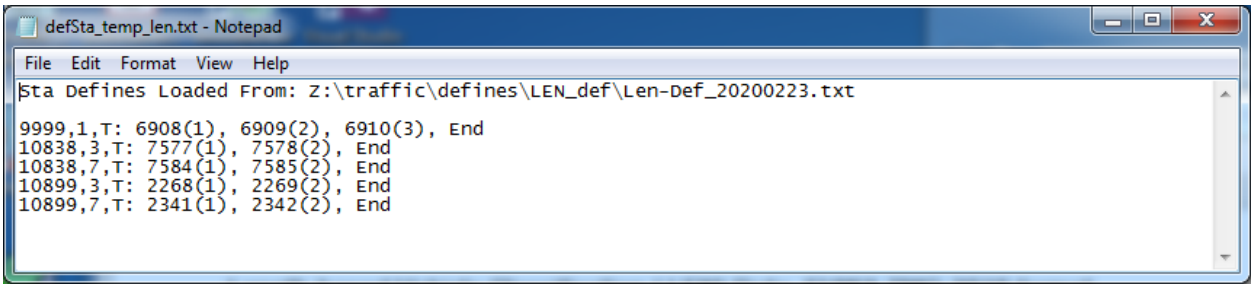


Figure 17: The read out when the “Show the read-in station defines” button is pressed

Once a station define file is prepared and the folder for the output file is set using the provided folder browser, pressing the “**Run Begin-to-End Dates**” button produces the final TMG-2016 LEN file. Before it runs, it checks the begin and end dates to make sure that the begin date is older than the end date. There is also an option to display the final output file in Notepad using the check mark labeled “Open the result in Notepad” for all three types of data.

The speed data can be retrieved using the same approach described above for the LEN data, i.e., a define file for speed data should be prepared and then the “Run Begin-to-End Dates” button in the speed groupBox is pressed.

Note on speed bins: The speed bins produced by detHealth_app are not true speed bins but bins of 30-second average speeds retrieved from RTMS data. The bins were calculated using *(30-second average speed) x (30-second volume)*. It uses 22 speed bins spaced by 5mph, and its complete ranges are shown in Table 6. The range [A, B) denotes A≤ speed < B mph. How these bins are used to create a file in TMG 2016 SPD format is described in Appendix C.

Table 6: Speed Bin Ranges

Bin number	Speed range
Bin 1	[0, 20) mph
Bin 2	[20, 25) mph
Bin 3	[25, 30) mph
Bin 4	[30, 35) mph
Bin 5	[35, 40) mph
Bin 6	[40, 45) mph
Bin 7	[45, 50) mph
Bin 8	[50, 55) mph
Bin 9	[55, 60) mph
Bin 10	[60, 65) mph
Bin 11	[65, 70) mph
Bin 12	[70, 75) mph

Bin 13	[75, 80) mph
Bin 14	[80, 85) mph
Bin 15	[85, 90) mph
Bin 16	[90, 95) mph
Bin 17	[95, 100) mph
Bin 18	[100, 105) mph
Bin 19	[105, 110) mph
Bin 20	[110, 115) mph
Bin 21	[115, 120) mph
Bin 22	[120, ∞) mph

The column format of VOL station define file is slightly different from LEN and SPD define files, and thus its column defines are separately shown in Table 7. An example of VOL station define file is shown in Figure 18 by opening in a Notepad. The difference from LEN and SPD defines is that a VOL define file includes a column for two-letter *Functional Classification Code* after the P (primary) column.

Table 7: Column Formats Of Station Definition Files Of VOL Data

Column name	Description
Station_ID	Station ID not exceeding 6 digits
Direction_Code	1=N, 2=NE, 3=E, 4=SE, 5=S, 6=SW, 7=W, 8=NW
City	T=Twin Cities, R=Rochester
P	Primary (all detectors are primary for the current version)
Functional Classification Code	FHWA functional classification of the road in two characters. The first character is a numerical digit and the code is given below. 1=Interstate 2=Principal Arterial -- Other Freeways and Expressways 3=Principal Arterial -- Other 4=Minor Arterial 5=Major Collector 6=Minor Collector 7=Local The second character can be either “R” for rural and “U” for urban. Example: 1R=Rural Interstate, 4U=Urban Minor Arterial, 7U=Urban Local
(detector IDs)	List of detector IDs separated by comma
lanes	A separating word used to indicate lane numbers. Case not sensirive
(lane numbers)	Lane numbers corresponding to the detector IDs in the same order
End	End of a record word. Comments can be added after this field.

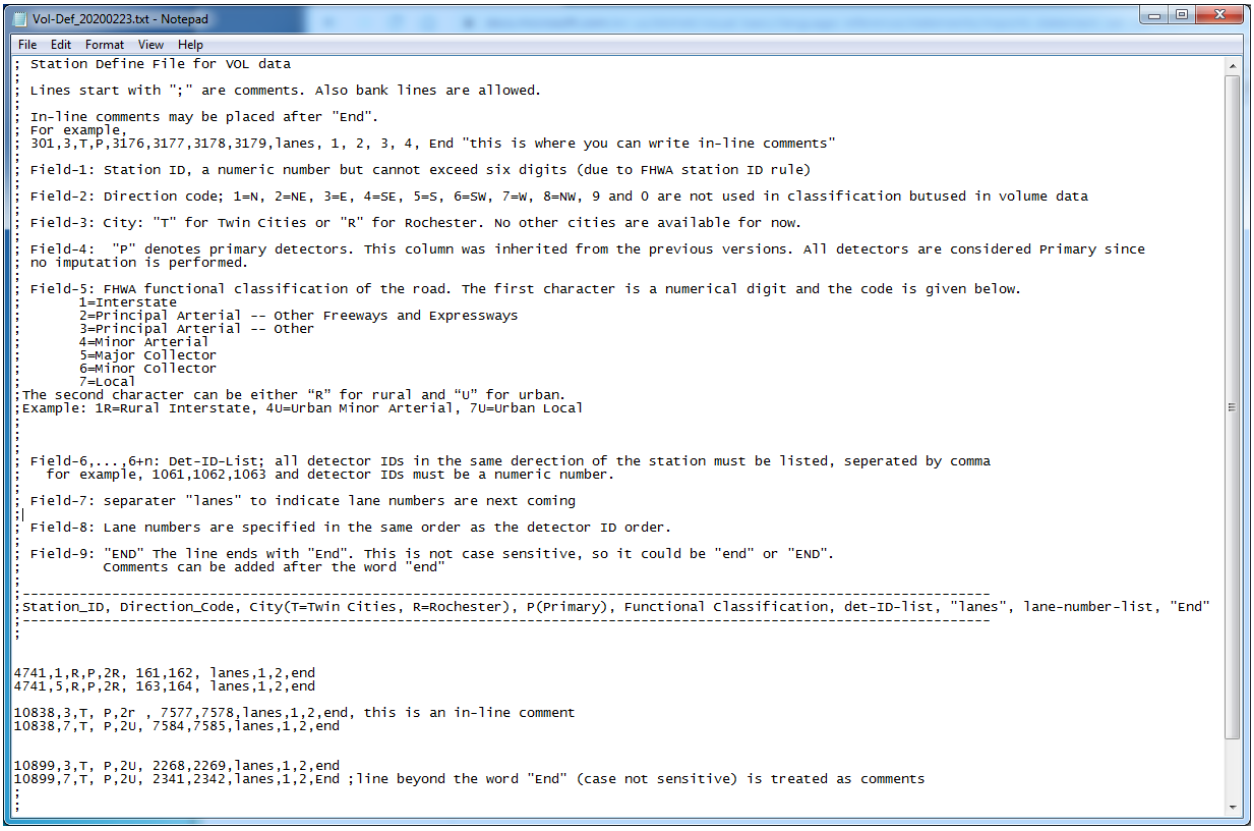


Figure 18: Screen capture of a sample VOL define file

The GUI and how to retrieve bulk VOL data is exactly same as the LEN and SPD data. After selecting the begin and end dates of the desired retrieval period using the provide calendar tools, pressing the "**Run Begin-to-End Dates**" button produces the TMG-2016 data for the given period and the stations defined in the defines file.

With bulk retrieval functions available for LEN, SPD and VOL, this tab should be used for data retrievals that require a large number of stations.

CHAPTER 4: CONLUSIONS AND RECOMMENDATIONS

4.1 CONCLUSIONS

This project was originally proposed to integrate RTMC length-based classification data into the existing volume and detector-health database created in the previous project. Unfortunately, maintaining a database server was considered too costly for a small office like TFA, and thus the TAP members for this project and the University research team collectively decided to remove the existing database server and create a stand-alone (server-less), desktop application program. Consequently, the initial phase of this project focused on migrating data and the management software from a client/server-based system to a stand-alone data system. Migration was mainly achieved through converting database tables to CSV files and then developing new file-based data search/retrieval functions along with the use of a hierarchical directory tree.

After successfully completing the migration, application programs were added as tabs in the detHealth_app program. The front page of the original detHealth_app program provides a pie chart and a list of detector-health class, from which more details can be retrieved. This function was successfully restored by implementing retrieval functions based on CSV file searches. The "r_node Map" tab had some database dependencies, and those were also removed and converted to use the new data structure. Three new tabs were added: the "Volume/Speed" tab in which speed data was integrated; the "LEN Class" tab in which length-based RTMC classification was integrated; and the "Bulk FHWA" tab in which bulk data in TMG-2016 formats for volume, speed, and classification were provided.

In conclusion, the objective of integrating length-based classification data into the existing traffic volume data was successfully completed after conversion from the existing client/server model to a stand-alone data model. In addition, the research team implemented three batch-processing tools to allow retrieval of a large amount of FHWA-formatted data for volume, speed and length-based classification.

4.2 RECOMMENDATIONS

The software detHealth_app created in this project has been thoroughly tested and successfully used by MnDOT TFA. The recommendations written here are purely based on the principal investigator's perspective and opinion.

Currently, MnDOT is using detHealth_app to produce volume, speed, and length-based classification in TMG-2016 format. The produced FHWA-formatted data are then loaded to the Jackalope data management system provided by the High Desert Traffic (HDT) company. Jackalope provides quality control and submission services to FHWA TMAS. Setting up stations and loading data to Jackalope are currently done manually. Since the detHealth_app is capable of producing station description files (.STA), it would be less laborious if the manual processes were all automated with the available station description files and some sort of scripts or an interfacing program.

MnDOT currently has separate computer programs for producing continuous count (CC) and short-duration count (SC) traffic data. Having separate programs has the benefit that each program can be more clearly run and assigned to different personnel. However, since the detHealth_app is capable of checking quality of traffic volume data, there might be a benefit from integrating CC and SC programs to the detHealth_app software. Thus, analyzing benefits and drawbacks of this integration is recommended.

REFERENCES

[1] Federal Highway Administration. (2016). *Traffic Monitoring Guide*. FHWA, Washington, DC.

[2] Federal Highway Administration. (2023). *Traffic Monitoring Guide*. FHWA, Washington, DC.

[3] Federal Highway Administration. (1995). *Traffic Monitoring Guide*, FHWA, Washington, DC.

[4] American Association of State Highway and Transportation Officials. (1992). *AASHTO Guidelines for Traffic Data Programs*. American Association of State Highway and Transportation Officials, Washington, DC.

[5] Kwon, T. M. (2020). *Improve Traffic Volume Estimates from MnDOT's Regional Traffic Management Center* (Report No. MN 2020-02). Minnesota Department of Transportation, St. Paul, MN.

APPENDIX A
FHWA VOL FORMAT (TMG 2016)

FHWA collects data from State DOT sources and feeds them to their Travel Monitoring Analysis System (TMAS). This data collection includes traffic volume, vehicle classification, and truck weight. The following describes the volume (VOL) format that detHealth_app produces and conforms to TMG 2013-2016. Volume records are all specified in hourly traffic volume.

1. Record Type (Column 1)
   3 = Traffic volume record (Code the value "3" in the first column.)

2. FIPS State Code (Columns 2-3) – "27" for Minnesota

3. Functional Classification Code (Columns 4-5) –

   Column 4 contains one of the Functional Classification Codes listed in below Table. Column 15 contains either an "R" for rural or "U" for urban. For example, a code of 2R indicates a Rural Principal Arterial – Other Freeways and Expressways.

Code	Functional Classification
1	Interstate
2	Principal Arterial – Other Freeways and Expressways
3	Principal Arterial – Other
4	Minor Arterial
5	Major Collector
6	Minor Collector
7	Local

4. Station Identification (Columns 6-11) –
   This should be right-justified with unused columns zero-filled.

5. Direction of Travel Code (Column 12) -

Code	Direction
1	North
2	Northeast
3	East
4	Southeast
5	South
6	Southwest
7	West
8	Northwest
9	North-South or Northeast-Southwest combined (volume stations only)
0	East-West or Southeast-Northwest combined (volume stations only)

6. Lane of Travel (Column 13) – 1 digit

Code	Lane
0	Data with lanes combined
1	Outside (rightmost) lane
2-9	Other lanes

7. Year of Data (Columns 14-17) – 4 digits

8. Month of Data (Columns 18-19) – 2 digits
   01 = January
   02 = February
   03 = March
   04 = April
   05 = May
   06 = June
   07 = July
   08 = August
   09 = September
   10 = October
   11 = November
   12 = December

9. Day of Data (Columns 20-21) – 2 digits
   Code the day of the month of data, 01-31. Must correspond to the month of data.

10. Day of Week (Column 22) – 1 digit
   1 = Sunday
   2 = Monday
   3 = Tuesday
   4 = Wednesday
   5 = Thursday
   6 = Friday
   7 = Saturday

11 through 34. Traffic Volume Counted Fields (Columns 23-27, ..., 138-142) – 5 digits per hour
   Enter the traffic volume counted during the hour covered in the columns indicated in below Table.
   If the data is missing, blank fill the appropriate columns.

Field	Hour Covered
11	after 00:00 to 01:00
12	after 01:00 to 02:00
13	after 02:00 to 03:00
14	after 03:00 to 04:00
15	after 04:00 to 05:00
16	after 05:00 to 06:00
17	after 06:00 to 07:00
18	after 07:00 to 08:00
19	after 08:00 to 09:00
20	after 09:00 to 10:00
21	after 10:00 to 11:00
22	after 11:00 to 12:00
23	after 12:00 to 13:00
24	after 13:00 to 14:00
25	after 14:00 to 15:00
26	after 15:00 to 16:00
27	after 16:00 to 17:00
28	after 17:00 to 18:00
29	after 18:00 to 19:00
30	after 19:00 to 20:00
31	after 20:00 to 21:00
32	after 21:00 to 22:00
33	after 22:00 to 23:00
34	after 23:00 to 24:00

35. Restrictions (Column 143) – 1 digit
    0 = no restrictions
    1 = construction or other activity affected traffic flow, traffic pattern not impacted
    2 = traffic counting device problem (e.g., malfunction or overflow)
    3 = weather affected traffic flow, traffic pattern not impacted
    4 = construction or other activity affected traffic flow, traffic pattern impacted
    5 = weather affected traffic flow, traffic pattern impacted

APPENDIX B
FHWA LEN FORMAT (TMG 2016)

FHWA collects data from State DOT sources and feeds them to their Travel Monitoring Analysis System (TMAS). This data collection includes traffic volume, vehicle classification, and truck weight. The following describes the length-based classification (LEN) format that detHealth_app produces and conforms to TMG 2013-2016.

1. Record Type (Column 1) – "C" for classification

2. FIPS State Code (Columns 2-3) – "27" for Minnesota

3. Station Identification (Columns 4-9) – *6 digits*
   This field should be right-justified with unused columns zero-filled.

4. Direction of Travel Code (Column 10) – 1 digit

Code	Direction
1	North
2	Northeast
3	East
4	Southeast
5	South
6	Southwest
7	West
8	Northwest
9	North-South or Northeast-Southwest combined (volume stations only)
0	East-West or Southeast-Northwest combined (volume stations only)

5. Lane of Travel (Column 11) – 1 digit

Code	Lane
0	Data with lanes combined
1	Outside (rightmost) lane
2-9	Other lanes

Note: The Station ID, Direction of Travel, and Lane of Travel make up the Station Code. *There should be one Station Description record per Station Code.*

6. Year of Data (Columns 12-15) – *4 digits*

7. Month of Data (Columns 16-17) – *2 digits*

8. Day of Data (Columns 18-19) – *2 digits*
   Code the day of the month of data, 01-31. Must correspond to the month of data.

9. Hour of Data (Columns 20-21) – *2 digits*
   Code the beginning of the hour in which the count was taken:
   00 = after 00:00 to 01:00
   01 = after 01:00 to 02:00
   ...
   ...
   …
   22 = after 22:00 to 23:00
   23 = after 23:00 to 24:00

10. Classification Data Time Interval – (Column 22) – *blank to indicate 60-minute interval*

11. Total Interval Volume (Columns 23-27) – *5 digits*
Total traffic volume of the hour specified in field #9.

12. Restrictions (Column 28) – *1 digit*
0 = no restrictions
1 = construction or other activity affected traffic flow, traffic pattern not impacted
2 = traffic counting device problem (e.g., malfunction or overflow)
3 = weather affected traffic flow, traffic pattern not impacted
4 = construction or other activity affected traffic flow, traffic pattern impacted
5 = weather affected traffic flow, traffic pattern impacted

13. Class 1 Count (Columns 29-33) – *5 digits*
RTMC motorcycle count of the hour

14. Class 2 Count (Columns 34-38) – *5 digits*
RTMC short-length vehicles count of the hour

15. Class 3 Count (Columns 39-43) – *5 digits*
RTMC medium-length vehicles count of the hour

16. Class 4 Count (Columns 44-48) – *5 digits*
RTMC long-length vehicles count of the hour

17. Class 5 Count (Columns 49-53) – *5 digits,* leave blank to indicate no data

18. Class 6 Count (Columns 54-58) – *5 digits,* leave blank to indicate no data

19. Class 7 Count (Columns 59-63) – *5 digits,* leave blank to indicate no data

20. Class 8 Count (Columns 64-68) – *5 digits,* leave blank to indicate no data

21. Class 9 Count (Columns 69-73) – *5 digits,* leave blank to indicate no data

22. Class 10 Count (Columns 74-78) – *5 digits,* leave blank to indicate no data

23. Class 11 Count (Columns 79-83) – *5 digits,* leave blank to indicate no data

24. Class 12 Count (Columns 84-88) – *5 digits,* leave blank to indicate no data

25. Class 13 Count (Columns 89-93) – *5 digits,* leave blank to indicate no data

APPENDIX C
FHWA SPD FORMAT (TMG 2016)

FHWA collects data from State DOT sources and feeds them to their Travel Monitoring Analysis System (TMAS). This data collection includes traffic volume, vehicle classification, and truck weight. The following describes the speed data (SPD) format that detHealth_app produces and conforms to the TMG 2013-2016 SPD format.

1. Record Type (Column 1) – "T" for vehicle speed record

2. FIPS State Code (Columns 2-3) – "27" for Minnesota

3. Station Identification (Columns 4-9) – *6 digits*
   This field should be right-justified with unused columns zero-filled.

4. Direction of Travel Code (Column 10) – 1 digit

Code	**Direction**
1	North
2	Northeast
3	East
4	Southeast
5	South
6	Southwest
7	West
8	Northwest
9	North-South or Northeast-Southwest combined (volume stations only)
0	East-West or Southeast-Northwest combined (volume stations only)

5. Lane of Travel (Column 11) – 1 digit

Code	**Lane**
1	Rightmost lane
2-9	Other lanes

Note: The Station ID, Direction of Travel, and Lane of Travel make up the Station Code. **No combined lanes are allowed.** *There should be one Station Description record per Station Code.*

6. Year of Data (Columns 12-15) – *4 digits*

7. Month of Data (Columns 16-17) – *2 digits*

8. Day of Data (Columns 18-19) – *2 digits*
   Code the day of the month of data, 01-31. Must correspond to the month of data.

9. Hour of Data (Columns 20-21) – *2 digits*
   Code the beginning of the hour in which the count was taken:
   00 = after 00:00 to 01:00
   01 = after 01:00 to 02:00
   ...
   ...
   …
   22 = after 22:00 to 23:00
   23 = after 23:00 to 24:00

10. Speed Data Time Interval – (Column 22) – *blank to indicate 60-minute interval*

11. Definition of First Speed Bin (Column 23) – *Optional*
This column was left blank, which indicates the first bin's range is [0, 20) mph.

12. Total Number of Speed Bins Reported (Columns 24-25) – *2 digit number*
Since the total number of bins recorded is 22, this field is always "22".

13. Total Interval Volume (Columns 26-30) – 5 digit number
Total volume of the hour obtained from the detector volume data. This is not the sum of speed bin data but directly obtained from the director volume such that the total of speed bins may not match with this total volume, but discrepancy indicates missing data in speed bins.

14. Bin 1 Count (Columns 31- 35) – *5 digits*
Count of vehicles in the speed range, 0≤Speed<20 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

15. Bin 2 Count (Columns 36- 40) – *5 digits*
Count of vehicles in the speed range, 20≤Speed<25 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

16. Bin 3 Count (Columns 41- 45) – *5 digits*
Count of vehicles in the speed range, 25≤Speed<30 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

17. Bin 4 Count (Columns 46-50) – *5 digits*
Count of vehicles in the speed range, 30≤Speed<35 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

18. Bin 5 Count (Columns 51-55) – *5 digits*
Count of vehicles in the speed range, 35≤Speed<40 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

19. Bin 6 Count (Columns 56-60) – *5 digits*
Count of vehicles in the speed range, 40≤Speed<45 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

20. Bin 7 Count (Columns 61- 65) – *5 digits*
Count of vehicles in the speed range, 45≤Speed<50 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

21. Bin 8 Count (Columns 66-70) – *5 digits*
Count of vehicles in the speed range, 50≤Speed<55 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

22. Bin 9 Count (Columns 71- 75) – *5 digits*
Count of vehicles in the speed range, 55≤Speed<60 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

23. Bin 10 Count (Columns 76-80) – *5 digits*
Count of vehicles in the speed range, 60≤Speed<65 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

24. Bin 11 Count (Columns 81- 85) – *5 digits*
Count of vehicles in the speed range, 65≤Speed<70 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

25. Bin 12 Count (Columns 86-90) – *5 digits*
Count of vehicles in the speed range, 70≤Speed<75 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

26. Bin 13 Count (Columns 91-95) – *5 digits*
Count of vehicles in the speed range, 75≤Speed<80 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

27. Bin 14 Count (Columns 96-100) – *5 digits*
Count of vehicles in the speed range, 80≤Speed<85 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

28. Bin 15 Count (Columns 101-105) – *5 digits*
Count of vehicles in the speed range, 85≤Speed<90 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

29. Bin 16 Count (Columns 106-110) – *5 digits*
Count of vehicles in the speed range, 90≤Speed<95 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

30. Bin 17 Count (Columns 111-115) – *5 digits*
Count of vehicles in the speed range, 95≤Speed<100 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

31. Bin 18 Count (Columns 116-120) – *5 digits*
Count of vehicles in the speed range, 100≤Speed<105 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

32. Bin 19 Count (Columns 121-125) – *5 digits*
Count of vehicles in the speed range, 105≤Speed<110 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

33. Bin 20 Count (Columns 126-130) – *5 digits*
Count of vehicles in the speed range, 110≤Speed<115 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

34. Bin 21 Count (Columns 131-135) – *5 digits*
Count of vehicles in the speed range, 115≤Speed<120 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.

35. Bin 22 Count (Columns 136-140) – *5 digits*
Count of vehicles in the speed range, speed≥120 mph, for the corresponding hour interval.
If no vehicles are observed in this speed range, "00000" is entered.