Arterial Performance Measures Software

Stanley E. Young, Dennis So Ting Fong
Arterial Performance Measures Software

Stanley E. Young
Traffax, Inc.

Dennis So Ting Fong
Traffax, Inc.

SBIR Phase 3 Joint Transportation Research Project
Traffax, Inc.
Purdue University

February 13, 2017

<table>
<thead>
<tr>
<th>Deliverable Reference:</th>
<th>D2.3 Report on Arterial Performance Measures Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Name:</td>
<td>Sensor Fusion and MOE Development for Off-Line Traffic Analysis of Real Time Data</td>
</tr>
<tr>
<td>Contractor:</td>
<td>Traffax, Inc.</td>
</tr>
<tr>
<td>Contract Number:</td>
<td>DTFH61-14-C-00035</td>
</tr>
<tr>
<td>Contract Term Start</td>
<td>9/4/2014</td>
</tr>
<tr>
<td>Contract Term End</td>
<td>9/4/2017</td>
</tr>
<tr>
<td>Key Personnel</td>
<td>Stan Young, Darcy Bullock, Dennis So Ting Fong</td>
</tr>
</tbody>
</table>
Recommended Citation

Acknowledgments
This work was supported by Traffax/USDOT SBIR DTFH6114C00035. The contents of this paper reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein, and do not necessarily reflect the official views or policies of the sponsoring organizations. These contents do not constitute a standard, specification, or regulation.
## Table of Contents

1. Introduction ........................................................................................................................................... 3
2. Elements of Software Development ..................................................................................................... 6
3. Sample Data Processing Using the Software ........................................................................................ 9
4. Software Elements for the CWS5200 Data Standard .......................................................................... 16
5. Software Elements for Performance Measure Visualizations ............................................................. 18

Appendix A ................................................................................................................................................ 22
1 Introduction

This Small Business Innovative Research (SBIR) project includes aspects of software development to encourage widespread adoption of the breakthroughs in arterial characterization using re-identification data. The software developed as part of this project provides reference routines (or pseudo-code) for implementation of the performance measure visualizations and data standards developed previously in this initiative. Standards and reference-able pseudo-code, when used in conjunction with prevailing procurement practices, encourage uniformity of application within the industry, create increased value for the customer (typically a transportation jurisdiction), and hasten adoption of best practices.

Improvements in technology and methods are often accompanied by a confused and fragmented market as vendors vie for market share. Vendors typically provide proprietary technology and software spanning from sensor placement in the field all the way to final information product in the back office. This approach has the byproduct of encouraging vendor lock-in. Also, technology products are typically marketed by vendors as distinctly different (and better) than those of their competitors in order to preserve brand recognition and limit competition. As technology proliferates and matures, the market demands price competition and it begins to enforce uniformity and interoperability through procurement specifications. Although vendors push against such market forces, product consistency and uniformity of experience is needed to enhance market growth. Early adopters may not be deterred by sole-source contracts or limited competition but the wider market (and associated public rules governing procurement) favors a more mature, standards-driven market place.

The re-identification market finds itself at such a transition point. Initial product offerings by vendors have been vertically integrated solutions based on vendor specific (typically proprietary) software and data structures integrated with hardware (sensor) offerings. Interoperability of re-identification equipment with down-stream processing for performance measures and visualization are NOT the norm in the market. Currently data analytics options available to customers are limited to the analytics offered by the equipment vendor unless the customer invests in a significant and costly data integration process. Market standardization overcomes this limitation by allowing different companies to optimize different aspects of the product chain – the data collection equipment, back-office processing, and downstream analytics. Open data structures are a key to enabling this process.

Note that a mature market does not prohibit or discourage vertically integrated products, in fact such products provide substantial cost benefits to users. Rather, standards exert market pressure so that product offerings (even in vertically integrated solutions) are best of breed, interoperable, and insure long data shelf life. In short, standards minimize the danger that concerns over legacy data, software, or equipment will become a primary procurement driver and initial and future procurements may compete on value and functionality rather the need to preserve past investments (a.k.a vendor lock-in).

Similarly, consistency in the development and use of performance measures and analytics provides value and efficiency to public agencies. As transportation agencies are increasingly asked to demonstrate accountability for the systems that they manage through outcome-based performance measures (as promoted in the most recent MAP-21 comprehensive highway funding bill), consistent and industry-standardized methods to create performance measures are in increasing demand. However, in the current limited funding environment transportation agencies must better target the use of funding resources – both
in terms of identifying specific corridors for improvement as well as allocating funds for tools to assist in locating those corridors. Customers desire quality assurance from analytics tools such that the same set of inputs will generate the same output, regardless of vendor. Vendor practices to differentiate products, though perhaps well intentioned, work against consistency of practice in the public domain. Apart from a formal standards initiative, making reference implementations available is one method that can be used in the procurement process to counteract the natural tendency of market forces to differentiate product offerings between vendors and thus introduce variance in customer experience.

New sources of traffic data, particularly travel time traffic data, are continually evolving. Currently the primary types of travel time traffic data fall into two broad categories. Commercial vehicle probe (CVP) data, from the market’s three dominant companies - INRIX, HERE and TomTom, provides traffic information based on a subset of cars and trucks traveling the roadway that self-report their location and speed. The size of that subset is frequently cited as 1% to 2% of all vehicles, but precise metrics are a guarded trade secret by vendors. CVP data was introduced a decade ago and now drives many operations and planning functions within many state departments of transportation and metropolitan planning organizations. Vehicle probe data is beginning to obtain widespread use, particularly for freeway operations. Although no formal standards have been created for formation of the CVP data sets, market forces have matured the products such that variation between vendors is minimal. The sharing of lessons learned among states and, more importantly, re-use of procurement specifications have proven effective in creating uniformity of practice and customer expectations. Although no official standards have been promulgated from professional standards organizations (such as ITE, SAE, or ISO), the delivery and use of CVP data has matured such that public agencies expect and obtain a consistent product and experience in the market.

The second major source of traffic travel time data is re-identification data from Bluetooth and WiFi. Although it was introduced about the same time as CVP data, re-identification data has not matured in a similar fashion. Different vendors continue to provide data in varying formats in their vertically integrated products. Unlike vehicle probe data, which benefitted from large public procurement processes from which the entire industry gained knowledge, procurements of re-identification data have typically been smaller and isolated, many times being included as a sub-element of a larger construction project. As a result, the market for re-identification data is segmented by vendor, with customer expectations and experience highly varied. Unlike CVP data, market forces alone have proven insufficient to rapidly mature the market. As a result of these conditions, re-identification technology market offerings remain vendor specific with wide variance of customer experience. Adoption of best practices for re-identification technology including processing, archiving, data formats, analytics, and visualizations, are lagging in the industry. Each re-identification vendor typically provides a vertically integrated product that includes the re-identification sensor, communications, back-office processing, analytics tools, and performance visualization. No common set of user expectations has manifested itself in uniform procurement specifications to date.

A goal of this project is to provide additional tools to encourage best practice and uniformity in the market by providing reference implementations for re-identification data formats and performance measure visualizations. If the segment data standard, CWS5200 as submitted in an earlier deliverable to the Sensor Fusion and MOE Development for Off-Line Traffic Analysis of Real Time Data project, were adopted within procurement specifications, it would result in the ability to easily port data from one
equipment vendor to a different analytics tool vendor, empowering the customer to take advantage of best of breed products. Similarly, reference software routines, enforced by procurement practices, would provide a common experience for customers, enabling transferrable lessons learned and quicker adoption of best practices. Combined, these (data standard with reference software routines for implementation of performance measure visualizations) will encourage market maturation – creating uniform expectations and experience for the customer, increasing customer value, extending the shelf-life of data, and ultimately faster market growth.
2 Elements of Software Development

As a part of the project, reference software routines were developed for implementing a re-identification data format (CWS5200) as well as travel time overlay charts and cumulative frequency distributions. These software routines were integrated into two software packages.

The first software package is BluSTATs. BluSTATs is the base Bluetooth data processing software developed and maintained by Traffax Inc. Modifications have been made to BluSTATs to incorporate the use of the CWS5200 segment re-identification data standard. BluSTATs, although proprietary to Traffax Inc., is representative of any vendor’s back-office software which accepts raw data from field sensors and processes it into matched pair data reflecting observed travel times for a single vehicle. The term ‘segment data’ refers to the data set that contains the individual vehicle travel times for vehicles that traverse a specific link or corridor (referred to generically as a segment). The CWS5200 standard encapsulates segment data in a consistent, vendor-agnostic format. Re-identification data collected from Bluetooth sensors is processed within BluSTATs and then exported in a CWS5200 compliant data file utilizing the reference software routine for the CWS5200 segment data standard.

A second software package named VPXplore accepts the CWS5200 compliant data file for subsequent analytics and visualizations, again utilizing the reference software routines developed for CWS5200. VPXplore is representative of any independent analytics package capable of accepting CWS5200 re-identification data and creating value-added metrics and visualizations. VPXplore was originally created at the University of Maryland Center for Advanced Transportation Technology to assist in traffic data validation efforts as part of the I-95 Corridor Coalition’s Vehicle Probe Project. Based in Matlab™, VPXplore is a set of software scripts that can be shared as pseudo-code implementations. The program was augmented with libraries that allow it to read CWS5200 data files, calculate arterial performance measures, and create visualizations of those measures (overlay plots and cumulative frequency diagrams) consistent with the guidelines produced from the original research. Whereas BluSTATs is proprietary commercial software, the VPXplore code is freely shareable such that the routines can be used as pseudo-code and/or referenced in procurement specifications.

Figure 2.1 is a diagram of the software framework. BluSTATs (module A1) is proprietary to Traffax Inc. It is a closed base of source code distributed only in compiled format and it requires a license from Traffax Inc. for use. It is representative of back-office processing software distributed by any vendor of re-identification technology. The same CWS5200 reference software code integrated into VPXplore was also integrated into the BluSTATs source code to provide CWS5200 compliant segment data sets.
VPXplore is representative of an independent analytics package capable of ingesting the CWS5200 data standard and creating a variety of visualizations. The primary difference between VPXplore and third-party analytics packages, is that VPXplore is also freely sharable as a set of open source routines in Matlab™.

The data bridge, based on the CWS5200 data standard between BluSTATs and VPXplore, enables VPXplore to ingest a standardized version of re-identification data and create information and analytics. The software library used to read and write CWS5200 files in VPXplore is the same library compiled into BluSTATs, as denoted by models A2 and B2 in Figure 2.2, insuring uniformity of implementation.
In the second phase of the software development, standard visualization routines were embedded into VPXplore for travel time overlay charts and CFDs as denoted by module B3 in Figure 2.2.

VPXplore was originally developed to compare data from multiple vendors, as was required within the validation effort of I-95 Corridor Coalition’s Vehicle Probe Project (http://i95coalition.org/projects/vehicle-probe-project/). VPXplore was created to analyze data quality on signal controlled arterials because traditional validation measures used on freeways were ineffective on signal controlled arterials. VPXplore evolved from single purpose scripts into an integrated set of Matlab™ routines to visually compare various CVP data sets with reference re-identification data sets. Initial versions of VPXplore used non-standard data structures ported from other processes or software to enable the visual comparisons. The need to directly compare CVP data with re-identification data (each with different data formats) initiated the process to create standard visualization routines. As the number of vendors increased and the complexity of analysis increased, VPXplore began to incorporate more robust data structures that could accommodate multiple sources of traffic data. The motivation for this development was not to create reference implementations as earlier discussed, but rather to minimize programming and facilitate efficient validation. This constituted the first step toward standard data formats and reference software implementations that could be used with multiple types of data.

When the need to implement the CWS5200 appeared, VPXplore, with its evolving trend toward consistent traffic data formats and methods, was a natural basis for initial implementation. Furthermore, overlay charts and cumulative frequency distributions were already partially standardized within the software routines. This allowed the ability to provide a reference implementation of travel time visualizations suitable to both re-identification data and CVP data.
3 Sample Data Processing Using the Software

Prior to diving into the technical details of the reference software routines that implement the CWS5200 data standard, travel time overlay charts, and CFD charts, a walk-through of the software functionality is provided in this chapter. This walk-through familiarizes the reader with the steps involved in using BluSTATs and VPXplore. The ability to port data from BluSTATs to VPXplore already existed as one of the ad-hoc procedures to support validation exercises. This older method to port data between BluSTATs and VPXplore used a non-standard data structure and non-reusable software code. However, since this older method for porting data already existed, it was used for validating the new software. The standardized CWS5200 data bridge and reference implementations of overlay charts and CFDs were tested against these previous methods to insure consistency and accuracy of output. Close comparison of output before (when proprietary, non-standardized practices were implemented) with after (when CWS5200 and reference visualization routines were utilized) provides confirmation of the accuracy of the referenced software routines.

As an example, a sample re-identification data set from New Jersey Route 13 will be used. This data was collected in July of 2015 as part of the validation exercises of the Vehicle Probe Project. The location of data collection is shown in Figure 3.1 below. The data sample comes from the segment from sensor E (labeled ECC E in Figure 3.1) to sensor F (labeled EE5 F), which is approximately 0.5 miles in length. A travel time plot of the matched pair segment data for the test segment is shown in Figure 3.2. BluSTATs is the software used for processing and displaying the segment re-identification data in Figure 3.2. As previously noted BluSTATs is proprietary and is representative of any back-office re-identification processing software. Although BluSTATs has output functions to allow segment data to be written to spreadsheets or data files, these formats were created specific to the Traffax technology and are not suitable for a vendor-agnostic standard data format. Any follow-on software that used the original data outputs needed to be customized to the available BluSTATs output format.
Figure 3.2 Travel Time Plot from BluSTATs

As a part of this research project, BluSTATs has been extended with software modules that allow CWS5200 data output, resulting in a MATLAB™ data file not customized to Traffax’s BluSTATs product, but which can be ported to any CWS5200 compliant analytics package.

VPXplore has also been modified to accept the CWS5200 data file. A screen capture from VPXplore, depicting the input process is shown in Figure 3.3. Note that VPXplore had previously been enabled to accept BluSTATs data output using BluSTATs customized data formats. As a result VPXplore is capable of accepting the E-F segment data from BluSTATs both in the CWS5200 format and in the older proprietary format.
Figure 3.3 VPXplore Input Screen

Figure 3.4 is a scatter plot of the E-F test segment in VPXplore. The data displayed in Figure 3.4 is a subset of the data from the scatter plot created in BluSTATs as depicted in Figure 3.2. The difference between BluSTATs output (Figure 3.2) and that of VPXplore (Figure 3.4) in the inclusion of outliers. BluSTATs plots both valid travel time data points as well as data points BluSTATs flagged as outliers. Only valid travel time matched pairs were written to the CWS5200 format from BluSTATs. As a result, Figure 3.4 contains only the valid travel time data points.

Figure 3.4 Travel Time Plot from VPXplore
To verify accuracy of the CWS5200 file format, E-F data was ported to VPXplore in the older proprietary BluSTATs data format. Within VPXplore the travel time scatter plot was created using both data sets, as shown in Figure 3.5. The older custom data format (labeled *BTM Reference Data*) is plotted using ‘X’ data markers, while the CWS5200 data is plotted using ‘O’ data markers. An exact overlay of data indicates that CWS5200 ports the data with the same level of precision as former custom data formats.

![Figure 3.5 Scatter Plot Comparison of Custom Data Format with CWS5200 Standard Data Format](image)

Note that VPXplore also accepts commercial vehicle probe (CVP) data. No data standard equivalent to CWS5200 exists for CVP data. VPXplore uses a data format created at UMD to which CVP data from different vendors could be ported. (This custom data format created at UMD may serve as the basis for future CVP data standardization.) A comparison of re-identification data from the CWS5200 data file with commercial vehicle probe data for the same segment is shown in Figure 3.6. Note that the CVP data set in Figure 3.6 contains two additional days of data (July 12th and 13th) not available from the re-identification data set.
Figure 3.6 Comparison of Re-Identification Data Set with Vehicle Probe Data Set
Both re-identification and CVP travel time data were converted to overlay charts and plotted for comparison in Figure 3.7. CVP data is depicted in red in the upper chart, and re-identification data from a CWS5200 compliant file is depicted in blue on the bottom chart. The same plotting engine was used for both data sets. This plotting engine provides a reference implementation for travel time overlay charts and demonstrates that these plotting routines are applicable to any travel time data regardless of source.

Figure 3.7 Comparative Weekday Overlay Charts
Figure 3.8 extends the visualization to cumulative frequency distribution (CFD) curves, providing a comparison not only between two different sources of travel time data, but also a comparison of travel time patterns throughout the day. The travel time from 11AM to noon is highlighted in the CFD as a bold black line in contrast to the travel times from other hours of the day.

Figure 3.8 Comparative CFD
4 Software Elements for the CWS5200 Data Standard

This chapter presents the software library routines used to implement the CWS5200 functionality in both BluSTATs and VPXplore. The software library for CWS5200 is comprised of three callable routines (collectively referred to as a CWS5200 function library) as explained below.

- **Creator Function:** This function creates an empty (or unpopulated) data structure to hold re-identification data compliant to the CWS5200 definition. This provides a data structure in Matlab™ insured to conform to the CWS5200 definition.

- **Verification Function:** This function verifies that the re-identification data inserted into the CWS5200 data structure conforms to the CWS5200 definition. The verification function can also be used to check the integrity of a data structure received from a third party. This function confirms that both the data structure format and the integrity of data comply with the CWS5200 standard.

- **Browser Function:** This function provides a convenient way to browse any CWS5200 data set and make modifications to comment and descriptor fields. A screenshot of the CWS5200 browser is shown in Figure 4.1 on the following page with data from the sample NJ13-0005 segment.

Collectively, this CWS5200 data library is incorporated into the BluSTATs software as well as VPXplore. A code listing for the Creator function is included in the appendix for reference. The source code for all three functions is made available at:

https://www.dropbox.com/sh/ywg2s8alor40ixb/AACgQJmgswElFfueqt-StFTpa?dl=0

These files are located under the folder named *CSW5200_function_library*. 
Figure 4.1 CWS Browser screenshot with data from NJ13-0005
5 Software Elements for Performance Measure Visualizations

The performance measure visualizations available through VPXplore employ a two-step process. The first step, labeled ‘Analytics Processing’ in Figure 5.1 below, extracts a common set of travel time parameters as well as a percentile based statistical summary from the CWS5200 data set. Due to the way the software evolved, VPXplore can also perform analytics processing on a variety of non-standard, proprietary data formats. This capability allows VPXplore to perform side by side comparison of CWS5200 re-identification data with CVP data from different vendors (in non-standard data formats). With some additional standards work, CVP datasets may one day be able to be ported using standard data structures, enabling VPXplore to be used with any standards compliant data set without additional programming.

After the base data is processed, the second step is to visualize the data by using a series of plotting routines. These routines provide scatter plots, overlay plots, and cumulative frequency diagrams (CFDs) based on the data summary extracted in step one. Critical portions of each step are explained below in greater detail for technical personnel. Key programming sub-routines from VPXplore are cited for reference implementation.

Prior to performance analytics processing, date and time filters are applied. VPXplore can define a date range different from the CWS5200 dataset being used. Similarly, data can be filtered by day of week such that visualizations are limited to only weekdays, weekends, a specific day of the week, or any combination thereof.

ANALYTICS PROCESSING

Four parameters are defined and calculated in the analytics processing. The explanations provided below assume the reader has a basic knowledge of technical computing and is familiar with the Matlab programming environment.
**x_dateTimelocal**  This is the date and time expressed as a Matlab datenum for each travel time data point in the dataset. It is an nx1 vector, where n is the number of matched pairs travel time data points in the segment dataset. In the current version of VPXplore, this date and time is taken from the first observation at the downstream sensor location. Future versions may allow for users to designate either upstream or downstream and either first, mid, or last observation.

**x_timelocal**  This is the time of day expressed as a number between 0 to 24 for each data point in the dataset. For example, 3:45 PM would be 15.75. It is also an nx1 vector. As with the x_dateTimelocal, the current version of VPXplore assigns the x-timelocal based on the first observation at the downstream station.

**y_traveltimeminutes**  This is the travel time in minutes for each data point in the segment data set. It is also an nx1 vector. Within the current version of VPXplore, travel time is calculated between the first observation at the upstream and downstream station. Future versions may allow for other permutations such as first to last, last to first, or mid to mid.

**cfd**  This is a 25 row by 19 column matrix of travel time percentiles. Each row number represents an hour of the day. Row 1 is from midnight to 1 AM. Row 2 is from 1 AM to 2 AM ... Row 24 is from 11 PM to midnight. The last row, row 25, represents data from all 24 hours. Each column is a percentile. The first column is the 5th percentile travel time, the second column the 10th percentile, and so on up to the last column (the 19th column) being the 95th percentile travel time. The 19 entries in row one contains the 5th, 10th, 15th, ... 90th, 95th percentile travel time for all trips between midnight and 1 AM for the segment.

Once these four parameters are calculated for a CWS5200 dataset (or other non-standard datasets), these parameters are stored for easy access by the visualization engine.

The software routines that perform the analytics processing can be viewed in the VPXplore code archive. The specific routine is available at the following location within the code base:

`VPPXPLOR_V_122\ANALYSIS_OPTIONS\PLOT_ONE_MINUTE_DATA_V2\private\SD_CWS5200.m`

The archive of the VPXplore source code can be accessed from the same site as the CWS5200 code library at:

[https://www.dropbox.com/sh/ywg2s8alar40jxb/AACgQJmgswElFfueqt-StFTpa?dl=0](https://www.dropbox.com/sh/ywg2s8alar40jxb/AACgQJmgswElFfueqt-StFTpa?dl=0)

These VPXplore source is located under the folder named `VPXplore_source`. Version 1.22 was uploaded to the site at the time of this publication. As upgrades are made, future versions will be uploaded.

As previously explained, VPXplore also creates standard visualizations from non-standard datasets. Additional software code within the archive exists that performs analytic processing on such data from CVP and re-identification data. The end result is that common travel time parameters and statistical summaries are derived that can be plotted with a common visualization engine.
VISUALIZATION ENGINE

Once the travel time data has been analyzed to create the above parameters, the data is passed to plotting routines that can create a variety of visualizations including scatter plot, overlay charts and cumulative frequency diagrams (CFDs). Several of these visualizations were demonstrated in section three of the report in Figures 3.4 (single source scatter plot), 3.5 & 3.6 (multi-source scatter plot), 3.7 (overlay chart), and 3.8 (combined overlay chart with CFD).

VPXplore currently provides three basic visualizations of the data. These include:

(1) **Scatter Plots**: Plots the travel time \(y_{\text{traveltimeminutes}}\) against the date&time \(x_{\text{datetimelocal}}\). A single data set can be plotted (as in Figure 3.4), or two data sets can be plotted (as in Figure 3.5 and 3.6). Any date filters or day of week filters will result in missing data along the x axis. A single data set scatter plot showing only weekdays is depicted in Figure 5.2 below for a segment NJ13-0005, another segment from the validation data set.

![Figure 5.2 Single source scatter plot showing only weekday data](image)

(2) **Overlay Charts**: Plots the travel time \(y_{\text{traveltimeminutes}}\) against the time of day \(x_{\text{timelocal}}\). This is always a comparison chart (as in Figure 3.7) between two data sets.

(3) **Combined Overlay and CFD Plot**: Plots overlay charts in the left two axes (as described above), and then plots the corresponding CFD in the right axes for each hour of the day, and for the entire day. Each line in the CFD visualization is created from a row of percentile data in the cfd data matrix. The percentile from the cfd matrix (0.05 through 0.95 in 0.05 increments) is plotted.
on the independent y axis against the travel time value of the corresponding percentile from the entry in the cfd matrix on the x axis. An example is shown in Figure 3.8 for the test segment. Another example from segment NJ13-0005 is shown below in Figure 5.3.

![Figure 5.3 Sample Overlay & CFD chart from VPXplore](image)

The highlighted curve in the CFD plots correspond to the hour of the day demarked by the two dashed lines in the Overlay chart. In Figure 5.3, the highlighted hour is noon to 1 PM, corresponding to the longest travel time and most unreliable travel time period of the day (based on the CFD chart). The highlight hour can be selected through user controls.

Also note the table of values in the lower left right CFD chart. The table depicts common performance measures derived from both data sets including the travel time index (TTI), planning time index (PTI), buffer time index (BTI), 25th, 50th, 75th and 95th percentile travel times, and the inter-quartile range (IQR).

**FINAL NOTES ON SOFTWARE**

The source code for VPXplore is provided for technical personnel to access critical sub-routines that provide either the implementation of the CWS5200 data standard, or reference implementation for the scatter plot, overlay charts, or cumulative frequency diagrams (CFDs). The current version of VPXplore was developed on Matlab 2013b. Sample output included in this document is from the current version (1.22) at the time of writing. This is a constantly evolving open source code set used primarily for validation activities. Along with the code, a data set is also distributed to exercise the software.
% Appendix A

function [varargout] = CWS5200_structure(varargin)

%% Creates / Populates the CWS5200 data structure
%%
%% CWS5200 Structure Definition
%% BASE LEVEL CWS5200 STRUCTURE FIELDS - (M) is Mandatory, (O) is optional
%%
%% ds.dataformat = 'CWS5200'; % (M) Either 'CWS5200' or 'CATTWORKS STANDARD 5200 REIDENTIFICATION DATASET'
%% ds.datasetname = ''; % (O) Text field with a descriptive name of the data set
%% ds.local_datetime.begin = [ ]; % (M) The beginning date and time, in the local time reference of the entire data set. % The begin date-time is preferred to the nearest minute. The field is in a text format of 'yyyy-mm-dd HH:MM:SS'. Ex., if the data set spans a two week period % from January 14 to January 28 of 2015, the local_datetime.begin would be '2015-01-14 00:00:00', indicating the beginning date-time of Jan 14, 2015. % Local time implies that any adjustment for Daylight Savings Time has been applied.
%% ds.local_datetime.end = [ ]; % (M) The end date and time, in the local time reference of the data set. % The end date-time is preferred to the nearest minute. The field is in a text format of 'yyyy-mm-dd HH:MM:SS'. Ex., if the data set spans a two week period % from January 14 to January 28 of 2015, the local_datetime.end would be '2015-01-28 23:59:59' reflecting the end of the day of Jan 28, 2015, or % alternatively it could be '2015-01-29 00:00:00' The intent is to bracket the timeframe of the dataset. Local time implies that any adjustment % for Daylight Savings Time has been applied.
%% ds.local_datetime.timezone = ''; % (O) A text field indicating the local time zone.
%% % Useful if data may be combined with other data sets that span time zones, or are specified in UTC or GMT date-time formats
%% ds.lengthunits = ''; % (M) Text field containing one of the following 'miles' or 'km'
%% ds.middefinition = ''; % (O) Type of Mid-point calculation
%% % If a mid-point, (or intermediate point) time offset is provided in the matched pair data, this text field describes how the mid-point is defined % or derived. Examples include such things as 'highest RSSI reading' or 'median observation'. The method for a mid-point or intermediate point % is often times technology dependent.
%% ds.datecreated = datestr(floor(now),29); % (O) The date the data set was created in 'yyyy-mm-dd' format.
%% ds.contact.name = ''; % (O) Text field with name of contact person in case of questions.
%% ds.contact.number = ''; % (O) Text field with phone number of contact in case of questions.
%% ds.contact.email = ''; % (O) Text field with email address of contact in case of questions.
%% ds.filename = ''; % (O) Text field with original filename of dataset

%% STATION SUB_STRUCTURE - for each station 'i' the following fields are defined
%%
%% ds.station(i).name = ''; % (M) Name of the station.
%% % This is a mandatory text field, and is intended to contain the identifying name of the station as determined or needed by the application, % such as a sequential naming scheme. It is a text field. Any combinations of characters are allowed except quotes, brackets, braces, ampersand, % or parentheses. Station name must be unique.
%% ds.station(i).uid = ''; % (M) Unique identifier of the station. This is a mandatory field.
%% % It is intended to be populated with a machine code or other automatically assigned identifier. Station UID must be unique.
%% ds.station(i).lat = [ ]; % (M) Latitude of the station location in decimal degrees
%% ds.station(i).lon = [ ]; % (M) Longitude of the station location in decimal degrees

Arterial Performance Measures Software
Contract #: DTFH61-14-C-00035
%% ds.station(i).roadway = ''; % (O) Text field indicating roadway on which station is located. Ex. ‘US-40’.
%% ds.station(i).crossroad = ''; % (O) Text field indicating nearest crossroad to the station. Ex. ‘Bell Rd.’
%% ds.station(i).notes = ''; % (O) Freeform text field for additional information about the station

%% SEGMENT SUB_STRUCTURE - for each segment i
%% ds.segment(i).name = ''; % (M) Name of the segment.
%% % Intended to contain the identifying name of the segment as determined or needed by the application. It is a text field.
%% % Any combinations of characters are allowed except quotes, brackets, braces, ampersand, or parentheses. Segment name must be unique.
%% ds.segment(i).name2 = ''; % (O) Alternate name of the segment.
%% % This is an optional field to facilitate a secondary naming scheme. It is a text field. Any combinations of characters are allowed
%% % except quotes, brackets, braces or parentheses. Segment secondary name should be unique.
%% ds.segment(i).upstreamstation = ''; % (M) Unique name of the upstream station.
%% ds.segment(i).downstreamstation = ''; % (M) Unique name of the downstream station.
%% ds.segment(i).length = ''; % (M) Length of segment, used to convert travel time to speed. Units are in ds.lengthunits
%% ds.segment(i).roadname1 = ''; % (O) Primary road designation (such as I-70)
%% ds.segment(i).roadname2 = ''; % (O) Secondary road designation (such as PA Tollway)
%% ds.segment(i).direction = ''; % (O) Direction given as a text field. Examples include ‘northbound’, ‘NB’, or ‘ccw’
%% % (For counter-clockwise, as in a beltway).
%% ds.segment(i).description = ''; % (O) Freeform text field for additional information about the segment

%% MATHED PAIR (mp) SUB_STRUCTURE - for matched pair data set i
%% ds.mp(i).segment = ''; % (M) The unique segment name for which matched pair data is provided
%% ds.mp(i).reidentificationtype = ''; % (M) Text field containing one of the following ‘BTM’, ‘WIFI’, ‘BTMWIFI’, ‘ALPR’, ‘TOLLTAG’
%% ds.mp(i).notes = ''; % (O) Freeform text field for additional information about the matched pair data
%% ds.mp(i).data = see subfield below
%% data.uid = ''; % (O) Alphanumeric string unique to the matched pair.
%% % An N by 1 vector (or 1 x N)
%% data.upstream_initial_datetimeoffset = []; % (M) The offset in decimal days from ds.local_datetime.begin
%% % Minimum precision is to the nearest second. This may be stored as single precision floating point (see implementation notes).
%% % An N by 1 vector (or 1 x N), where N is the number of data points in matched pair data set i
%% data.upstream_final_timeoffset = []; % (M) The offset in seconds from the initial observation at the upstream station to the last
%% % observation of the UID at the upstream station.
%% % An N by 1 vector (or 1 x N)
%% data.downstream_initial_timeoffset = []; % (M) The offset in seconds from the initial observation at the upstream station to the first
%% % observation of the UID at the downstream station.
%% % An N by 1 vector (or 1 x N)
%% data.downstream_final_timeoffset = []; % (M) The offset in seconds from the initial observation at the upstream station to the last
%% % observation of the UID at the downstream station.
%% % An N by 1 vector (or 1 x N)
%% data.upstream_mid_timeoffset = []; % (O) The offset in seconds from the initial observation at the upstream station to the midpoint
%% % of the upstream station.
%% % An N by 1 vector (or 1 x N)
%% data.downstream_mid_timeoffset = []; % (O) The offset in seconds from the initial observation at the upstream station to the midpoint
%% % of the downstream station.
%% Input variables to the function

    callmode = varargin{1}; % Only one call mode is defined 'V1'

%% Initialize output

    varargout{1} = []; % Default output is a empty

switch callmode

    case {'V1'}

%=================================================================
% Creates a blank CWS5200 structure and initialize it. Note: (M) is mandatory (O) is optional
%=================================================================

% Create base field definitions

    ds.dataformat = 'CWS5200'; % (M) Either 'CWS5200' or 'CATTWORKS STANDARD 5200 REIDENTIFICATION DATASET'
    ds.datasetname = ''; % (O) Text field with a descriptive name of the data set
    ds.local_datetime.begin = []; % (M) The beginning date and time, in the local time reference of the entire data set.
        % The begin date-time is preferred to the nearest minute. In the format of yyyy-mm-dd HH:MM:SS For example, if the data set spans a two week
        % period from January 14 to January 28 of 2015, the local_datetime.begin would be '2015-01-14 00:00:00' reflecting the beginning of Jan 14, 2015.
        % Local time implies that any adjustment for Daylight Savings Time has been applied.
    ds.local_datetime.end = []; % (M) The end date and time, in the local time reference of the data set.
        % The end date-time is preferred to the nearest minute. In the format of yyyy-mm-dd HH:MM:SS For example, if the data set spans a two week period
        % from January 14 to January 28 of 2015, the local_datetime.end would be '2015-01-28 23:59:59' reflecting the end of the day of Jan 28, 2015, or
        % alternatively it could be '2015-01-29 00:00:00' The intent is to bracket the timeframe of the dataset. Local time implies that any adjustment
        % for Daylight Savings Time has been applied.
    ds.local_datetime.timezone = ''; % (O) A text field indicating the local time zone.
        % Useful if data may be combined with other data sets that span time zones, or are specified in UTC or GMT date-time formats
    ds.lengthunits = ''; % (M) Text field containing one of the following 'miles' or ‘km’
    ds.middefinition = ''; % (O) Type of Mid-point calculation
        % If a mid-point, (or intermediate point) time offset is provided in the matched pair data, this text field describes how the mid-point is defined
        % or derived. Examples include such things as 'highest RSSI reading' or 'median observation'. The method for a mid-point or intermediate point
        % is often times technology or vendor dependent.
    ds.datecreated = datestr(floor(now),29); % (O) The date the data set was created in 'yyyy-mm-dd' format.
    ds.contact.name = ''; % (O) Text field with name of contact person in case of questions.
    ds.contact.number = ''; % (O) Text field with phone number of contact in case of questions.
    ds.contact.email = ''; % (O) Text field with email address of contact in case of questions.
    ds.filename = ''; % (O) Text field with original filename of dataset.
    ds.station = CWS5200_structure('station_blank'); % (M) A separate call to populate the station sub-structure. See below.
    ds.segment = CWS5200_structure('segment_blank'); % (M) A separate call to populate the segment sub-structure. See below.
    ds.mp = CWS5200_structure('mp_blank'); % (M) A separate call to populate the matched-pair (mp) sub-structure. See below.

    varargout{1} = ds;

    case {'station','station_blank'}

%=================================================================
% Create the station sub-structure. Note: (M) is mandatory (O) is optional.
%=================================================================

    % ‘station’ Returns a station structure with fields that are empty.
    % ‘station_blank’ Returns an empty station structure.
%=================================================================

Arterial Performance Measures Software

Contract #: DTFH61-14-C-00035
station.name = '';                      % (M) Name of the station.
% This is a mandatory text field, and is intended to contain the identifying name of the station as determined or needed by the application,
% such as a sequential naming scheme. It is a text field. Any combinations of characters are allowed except quotes, brackets, braces, ampersand,
% or parentheses. Station name must be unique.
station.uid = '';                       % (M) Unique identifier of the station. This is a mandatory field.
% It is intended to be populated with a machine code or other automatically assigned identifier. Station UID must be unique.
% The station uid and name may be the same.
station.lat = [];                       % (M) Latitude of the station location in decimal degrees
station.lon = [];                       % (M) Longitude of the station location in decimal degrees
station.roadway = '';                  % (O) Text field indicating roadway on which station is located. Ex. ‘US-40’.
station.crossroad = '';                % (O) Text field indicating nearest crossroad to the station. Ex. ‘Bell Rd.’
station.notes = '';                    % (O) Freeform text field for additional information about the station

if strcmp(callmode,'station')
    varargout{1} = station;          % Returns a station structure with the fields empty.
    return;
end
if strcmp(callmode,'station_blank')
    station(1) = [];
    varargout{1} = station;          % Returns an empty station structure.
    return
end

case {'segment','segment_blank'}
%===================================================================
% Create the segment sub-structure. Note: (M) is mandatory (O) is optional.
% ‘segment’ Returns a segment structure with fields that are empty.
% ‘segment_blank’ Returns an empty segment structure.
%===================================================================
segment.name               = '';            % (M) Name of the segment.
% Intended to contain the identifying name of the segment as determined or needed by the application. It is a text field.
% Any combinations of characters are allowed except quotes, brackets, braces, ampersand, or parentheses. Segment name must be unique.
segment.name2              = '';            % (O) Alternate name of the segment.
% This is an optional field to facilitate a secondary naming scheme. It is a text field. Any combinations of characters are allowed
% except quotes, brackets, braces or parentheses. Segment secondary name must be unique.
segment.upstreamstation    = '';            % (M) Unique name of the upstream station.
segment.downstreamstation  = '';            % (M) Unique name of the downstream station.
segment.length             = '';            % (M) Length of segment, used to convert travel time to speed. Units are in ds.lengthunits
segment.roadname1          = '';            % (O) Primary road designation (such as I-70)
segment.roadname2          = '';            % (O) Secondary road designation (such as PA Tollway)
segment.direction          = '';            % (O) Direction given as a text field. Examples include ‘northbound’, ‘NB’, or ‘ccw’
% (for counter-clockwise, as in a beltway).
segment.description        = '';            % (O) Freeform text field for additional information about the segment
if strcmp(callmode,'segment')
    varargout{1} = segment;          % Returns a segment structure with fields that are empty.
    return;
end
if strcmp(callmode,'segment_blank')
segment(1) = []; 
varargout(1) = segment; 
return; % Returns an empty segment structure.
end

case {’mp’,’matched pairs’,’mp_blank’}

%%=======================================================
%% Create the matched_pair (mp) sub-structure
%%=======================================================

mp.segment              = '';                % (M) The unique segment name for which matched pair data is provided
mp.reidentificationtype = '';                % (M) Text field containing one of the following ‘BTM’, ‘WIFI’, ‘BTMWIFI’, ‘ALPR’, ‘TOLLTAG’
mp.notes                = '';                % (O) Freeform text field for additional information about the matched pair data
mp.data                 = CWS5200_structure(’mpdata’);

switch callmode
  case {’mp’,’matched_pairs’}
    varargout{1} = mp;
  case ‘mp_blank’
    mp(1)        = [];
    varargout{1} = mp;
  otherwise
end
varargout{1}           = mp;

%%=======================================================
%% Create the matched_pair data (mpdata) sub-structure
%%=======================================================

case {’mpdata’,’matched pair data’}

%%==================================================================
%% Create the matched_pair data (mpdata) sub-structure
%%==================================================================

d.uid                               = '';   % (O) Alphanumeric string unique to the matched pair.
d.upstream_initial_datetimeoffset   = [];   % (M) The offset in decimal days from ds.local_datetime.begin
  % Minimum precision is to the nearest second. This may be stored as single precision floating point (see implementation notes).
d.upstream_final_timeoffset     = [];   % (M) The offset in seconds from the initial observation at the upstream station to the last
  % observation of the UID at the upstream station.
d.downstream_initial_timeoffset = [];   % (M) The offset in seconds from the initial observation at the upstream station to the first
  % observation of the UID at the downstream station.
d.downstream_final_timeoffset   = [];   % (M) The offset in seconds from the initial observation at the upstream station to the last
  % observation of the UID at the downstream station.
d.upstream_mid_timeoffset               = [];   % (O) The offset in seconds from the initial observation at the upstream station to the midpoint
  % of the upstream station.
d.downstream_mid_timeoffset             = [];   % (O) The offset in seconds from the initial observation at the upstream station to the midpoint
  % of the downstream station.
varargout{1}                        = d;

otherwise
end
Report Sponsor

The “Small Business Innovation Development Act of 1982” (Pub. L. No. 97-219), along with reauthorizing legislation (Pub. L. No. 99-443 and Pub. L. No. 102-564, the “Small Business Research and Development Enhancement Act of 1992”), seeks to encourage the initiative of the private sector and to use small business effectively to meet federal research and development objectives. To comply with statutory obligations of the Act, the U.S. Department of Transportation established the Small Business Innovation Research (SBIR) Program, which conforms to the guidelines and regulations provided by the Small Business Administration. Annually, small businesses are solicited to submit innovative research proposals that address the high-priority requirements of the U.S. Department of Transportation and that have potential for commercialization.

This report was developed through a partnership between Traffax, Inc., and Purdue University with funding from a Phase III SBIR contract (DTFH6114C00035) with the Federal Highway Administration. The project, entitled “Sensor Fusion and MOE Development for Off-Line Traffic Analysis of Real Time Data,” created and refined methods and tools for the characterization of performance along arterial corridors.

Publication

This report is part of a series of reports published in collaboration with USDOT, Traffax, Inc., and Purdue University. The full report series is available for download at http://docs.lib.purdue.edu/apmtp/.

Open Access and Collaboration with Purdue University

The Indiana legislature established the Joint Highway Research Project in 1937. In 1997, this collaborative venture between the Indiana Department of Transportation and Purdue University was renamed as the Joint Transportation Research Program (JTRP) to reflect state and national efforts to integrate the management and operation of various transportation modes. Since 1937, the JTRP program has published over 1,600 technical reports. In 2010, the JTRP partnered with the Purdue University Libraries to incorporate these technical reports in the University’s open access digital repository and to develop production processes for rapidly disseminating new research reports via this repository. Affiliated publications have also recently been added to the collection. As of 2017, the JTRP collection had over 1.5 million downloads, with some particularly popular reports having over 20,000 downloads.