ROADWAY SAFETY INSTITUTE

Human-centered solutions to advanced roadway safety

I-94 Connected Vehicles Testbed Operations and Maintenance

Melissa Duhn
Gordon Parikh
John Hourdos

Minnesota Traffic Observatory
Department of Civil, Environmental, and Geo-Engineering
University of Minnesota

Final Report

CTS 19-16

Technical Report Documentation Page

1. Report No. CTS 19-16	2.	3. Recipients Accession No.
4. Title and Subtitle I-94 Connected Vehicles Testbed Operations and Maintenance		5. Report Date June 2019 6.
7. Author(s) Melissa Duhn, Gordon Parikh, John Hourdos		8. Performing Organization Report No.
9. Performing Organization Name and Address Minnesota Traffic Observatory University of Minnesota Department of Civil, Environmental, and Geo- Engineering CE790 Pillsbury DR SE Minneapolis, MN 55455		10. Project/Task/Work Unit No. CTS#2018035 11. Contract (C) or Grant (G) No. DTRT13-G-UTC35
12. Sponsoring Organization Name and Address Roadway Safety Institute Center for Transportation Studies University of Minnesota University Office Plaza, Suite 440 2221 University Ave SE Minneapolis, MN 55414		13. Type of Report and Period Covered Final Report 14. Sponsoring Agency Code
15. Supplementary Notes http://www.roadwaysafety.umn.edu/publications/		

16. Abstract (Limit: 250 words)

In March 2017, the Connected Vehicle Testbed along I-94 went live. The original project was sponsored by the Roadway Safety Institute and built on the Minnesota Traffic Observatory's (MTO) existing field lab, also utilizing certain Minnesota Department of Transportation (MnDOT) infrastructure. The testbed originally consisted of seven stations, rooftop and roadside, capable of transmitting radar and video data collected from the roadway back to a database at the MTO for analysis, emulating what a future connected vehicle (CV) roadway will look like. This project funded maintenance and upgrades to the system, as well as movement of some stations due to construction on I-94. In addition, better visualization tools for reading the database were developed. The CV testbed is state-of-the-art, fully functional, and uniquely situated to attract freeway safety-oriented vehicle to infrastructure (V2I) and vehicle to vehicle (V2V) safety application development, implementation, and evaluation projects going forward.

17. Document Analysis/Descriptors Connected vehicles, Test beds, Trajectory, Radar, Visualization, Data collection, Vehicle to infrastructure communications, Vehicle to vehicle communications, Data analysis		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 71	22. Price

I-94 Connected Vehicles Testbed Operations and Maintenance

FINAL REPORT

Prepared by:

Melissa Duhn
Gordon Parikh
John Hourdos

Minnesota Traffic Observatory
Department of Civil, Environmental, and Geo- Engineering
University of Minnesota

June 2019

Published by:

Roadway Safety Institute
Center for Transportation Studies
University of Minnesota
University Office Plaza, Suite 440
2221 University Ave SE Minneapolis, MN 55414

ACKNOWLEDGMENTS

The funding for this project was provided by the United States Department of Transportation's Office of the Assistant Secretary for Research and Technology for the Roadway Safety Institute, the University Transportation Center for USDOT Region 5 under the Moving Ahead for Progress in the 21st Century Act (MAP-21) federal transportation bill passed in 2012.

The research team would like to thank the Roadway Safety Institute for its generous funding of this project and MnDOT for the use of equipment for the testbed. In addition, we would like to thank undergraduate students Kah Hin Lai and Eugene Hao Lee Yu for their efforts in developing software to visualize the testbed database.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

The I-94 Connected Vehicles (CV) Testbed was funded by the Roadway Safety Institute (RSI) to expand and upgrade the Minnesota Traffic Observatory's (MTO) previously existing I-94 Field Laboratory. RSI and Minnesota Department of Transportation (MnDOT) projects have covered most of the expenses of monitoring and maintenance, but due to the permanent placement of the lab, there are periods of time when no project utilizes it, and ongoing costs must be taken care of. This project allowed for the continuous operation and maintenance of the CV Testbed and I-94 Field Lab, both of which have many pieces of technology requiring ongoing monitoring and maintenance. The project also allowed for changes in the placement of the field lab due to the I-94/I-35W construction project that began in April 2018, as well as technology upgrades, like adding Wavetronix sensors to MTO stations.

The CV testbed is state-of-the-art, fully functional, and uniquely situated to attract freeway safety-oriented vehicle to infrastructure (V2I) and vehicle to vehicle (V2V) safety application development, implementation, and evaluation projects going forward. Specific projects like Queue Warning and Speed Harmonization will be implemented and tested on the corridor. The previous Field Lab was enhanced to support fully developed CV safety systems as well as the research and evaluation of the underlying human factors of such systems and the enhanced system has been in operation since March 2017.

In addition to the positioning of the Field Lab stations, maintenance, and upgrades, more sophisticated database visualization tools were developed during this project. These tools will be used to visualize and extract real-world vehicle trajectories from the sensor data.

Overall, the CV Testbed is unique from other CV pilot sites in that it operates on a fully functioning, high-volume freeway. It is not isolated or under fully controlled conditions. It is a step closer to what CVs entering the marketplace will be like. Given the unique nature of the system deployed in a real-world setting, any CV applications requiring interaction with a large number of unconnected vehicles could be tested in this facility, which is an important stepping stone for research as CVs start to enter the marketplace.

CHAPTER 1: INTRODUCTION

Safety and traffic operations concepts based on vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication have been in development for some time. Since 2003, with the assistance of the Minnesota Department of Transportation (MnDOT), the Minnesota Traffic Observatory (MTO) at the University of Minnesota has studied and experimented with infrastructure-based Queue Warning (Q-WARN) systems. A permanent field lab has been established at the high-crash area of westbound I-94 in Minneapolis, Minnesota, capturing detailed data on hundreds of crashes. This area experiences upward of 100 crashes annually, most of them rear-end crashes due to failure to stop or too little headway.

The Roadway Safety Institute (RSI) funded the creation of the I-94 Connected Vehicles Testbed as an addition to the existing I-94 Field Laboratory of the MTO. This project aims to expand and maintain both the already extensive instrumentation available at the I-94 Field Lab, as well as the Connected Vehicles (CV) Testbed specifically for the implementation and testing of Speed Harmonization (SPD-HARM) and Q-WARN systems. The current site was enhanced to support fully developed CV safety systems as well as the research and evaluation of the underlying human factors of such systems and began operation in March 2017.

Both the continuous operation of the I-94 Field Lab and now the CV Testbed involves many layers of technology that require constant monitoring and frequent maintenance. While most of such expenses have been covered by Roadway Safety Institute (RSI) and MnDOT projects, there are costs that are ongoing and periods when no projects are utilizing the Testbed infrastructure. The funding provided for this project allowed the uninterrupted operation of the Testbed, as well as changes and upgrades in view of the large reconstruction project MnDOT began in April 2018.

1.1 ORIGINAL I-94 CV TESTBED PROJECT AND ISSUES

The final goal of both the original I-94 CV Testbed project, along with this operations and maintenance period, is a fully functional CV testbed uniquely situated to attract freeway safety-oriented V2I and V2V safety application development, implementation, and evaluation projects.

The Testbed upgrade to MTO’s existing I-94 field laboratory’s surveillance and data collection infrastructure introduced the ability of collecting detailed vehicle trajectories from seven 24 GHz radar covering the I-94 WB high-crash frequency area in downtown Minneapolis. The radar infrastructure was complimented by a database system that collects the vehicle trajectories in real-time, stores them and makes them available to any external applications through a provided Application Program Interface (API). Developing the CV Testbed to provide such information from a busy and difficult to access freeway was challenging and required many customized hardware and software solutions. For example, due to the lack of power in two of the locations where radar was needed to provide seamless trajectories, MTO engineers designed and assembled a solar-powered solution mounted on temporary wooden poles.

Although the original plan was to deploy overlapping radar sensors on a half-mile stretch of I-94, it became clear following several meetings and discussions with MnDOT that it would not allow the MTO project team to install the sensors on a temporary basis and that it required a secure permanent deployment specifically in terms of the power line conduits. The original project, as budgeted, did not have the funds to complete such deployments. Instead, the project as much as possible used the existing available MnDOT camera poles, even if their location limited coverage and in some cases accuracy of the sensors. At this point in the project's lifespan, the priority was to establish the backbone of the sensor communication network and data collection system.

In addition to the aforementioned issues with the radar station deployment, an additional problem involving the radar was encountered. The project had proposed using the latest generation of radar sensors, which provide higher performance and accuracy in heavy traffic sites like I-94. The proposal was developed following several reassurances from the manufacturer that these latest models would be available in the United States shortly after the start of the project. The original project officially started in November 2014, and the agreement was that we would purchase the latest generation; until they could be sold in the United States, MTO would receive the older generation for testing and deployment in the field lab. In February 2015, after considerable pressure, the manufacturer revealed that it did not anticipate making the sensors available until February 2016 because the matter was being litigated in U.S. courts. Given this reality, the project team decided to proceed with the older generation sensors, develop and deploy the system, identify where the older generation sensors were insufficient and by how much, and plan for a later upgrade to the better radar sensors when they became available.

The full system has been in operation since March 2017 and has already provided millions of vehicle trajectories, along with video and conventional sensor information. Given the custom-made nature of the whole system, it requires constant monitoring and relatively frequent maintenance, which can be costly since two of the sensor locations are only accessible by bucket truck. The MTO is continuing the development of the data collection and storage system as well as the development of data processing applications to clean, verify, and stitch together the raw vehicle trajectories.

1.1.1 Planned Project Maintenance and Effort

It is essential to maintain the current CV Testbed as well as keep it at the cutting edge of technology to succeed in attracting research projects using the available data. This extension project will cover the following efforts and expenses:

1. Cost of renting the I-94 lab rooftop stations that form the communication backbone of the whole system.
2. Effort for the ongoing monitoring of the system operation and the ongoing collection of video and sensor data.
3. Updates to the presentations and CV Testbed status material delivered as part of the final report.
4. Hardware replacement costs for current components of the CV Testbed.

5. Maintenance of current software infrastructure to ensure system security.
6. Continue the development of the system API that allows external applications to utilize the Testbed sensors.
7. Continue the development of data filtering and data mining applications that improve the quality of the raw collected vehicle trajectories.
8. Relocate one of the solar stations to a new location probably upstream of the 11th Street overpass.
9. Expansion of sensing, surveillance, and communication abilities along the testbed.

This report will cover each of the efforts categorized by hardware and software upgrades and maintenance.

CHAPTER 2: HARDWARE UPGRADES AND MAINTENANCE

To ensure the CV Testbed remained fully operational, regular monitoring of the hardware's status occurred. In addition to maintenance performed, stations were relocated due to construction. Also, expanding on the original project, Wavetronix radars were added to several locations to improve data collection resolution.

2.1 HARDWARE REPLACEMENT COSTS

Given the year-round operation of the laboratory, most of the Testbed hardware is exposed to extreme temperatures, hot and cold, as well as pollution from the 150,000 vehicles traveling on I-94 daily. From September 2017 to August of 2018, the stations were inspected and refurbished as required due to wear and tear. Common things needing to be fixed were power wiring, communication issues, and structural and mounting elements. On the solar stations, batteries were replaced, and the solar panels were checked for damage and grime. The solar units presented no trouble except when covered in snow for extended periods of time or becoming too cold to function properly, as is sometimes the case with Minnesota winters.

2.2 STATION RELOCATION

The original Testbed project included provision for the inevitable removal and relocation of some of the stations due to the construction on I-94 that began in August of 2017. The goal of relocating stations was to simply change the area of coverage, without losing any capabilities. With cooperation from the project contractors, MTO was able to keep stations running right up until they would be in the way of the excavators. Figure 2.1shows the original Testbed and areas of camera and radar coverage. Figure 2.2 shows the Testbed after stations were retrieved and relocated.

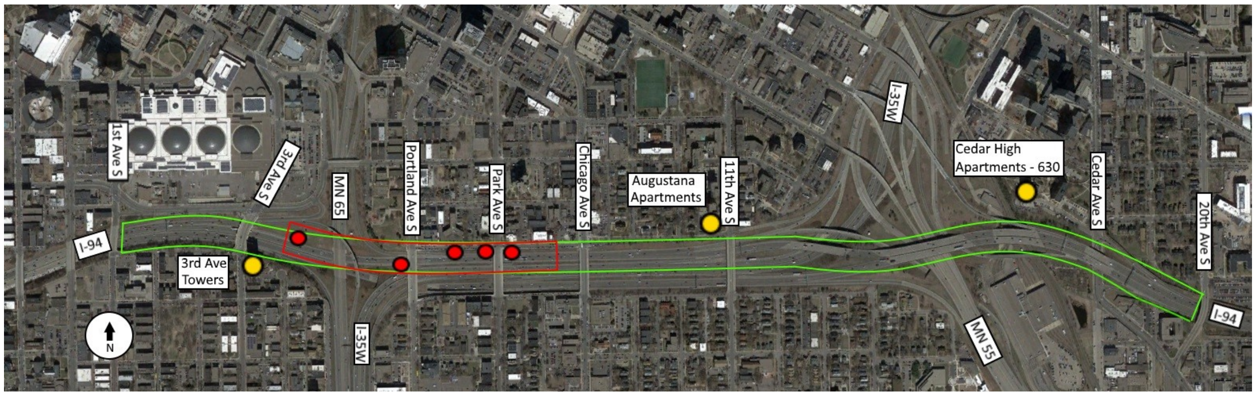


Figure 2.1 Original Testbed site showing radar sites (red dots), radar coverage (outlined in red), rooftop stations (yellow dots) and camera coverage (outlined in green).

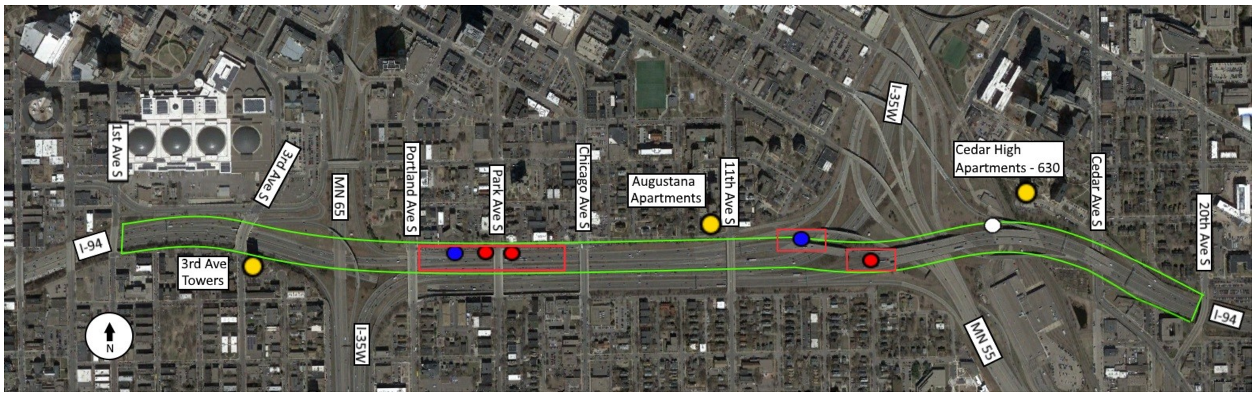


Figure 2.2 New testbed site showing radar sites (red dots), Wavetronix sites (blue dots), radar coverage (red outlines), rooftop stations (yellow dots), and a future MnDOT station (white dot).

As shown in Figure 2.2, both the 3rd Ave station and Portland Ave stations were retrieved because of the construction. Using parts from those stations with some refurbishment and rebuilding, new stations were deployed at Hiawatha and on the I-94 median just east of the I-35 junction. In addition to the layout at the time of writing, one additional future MnDOT station is planned for east of I-35 North.

The Hiawatha station consists of a solar pole moved due to construction on I-94. While a contracting company moved the pole itself, MTO staff rented a bucket truck and re-assembled the station on site. Figure 2.3 shows the pole after being assembled, with bucket truck still in view. Figure 2.4 shows a closer view of the pole's radar, camera, Wavetronix, and solar panels. (This shows the westward facing radar. There is an identical radar facing east on the opposite side of the pole, not shown.) Figure 2.5 shows the roadway views on either side of the Hiawatha station.

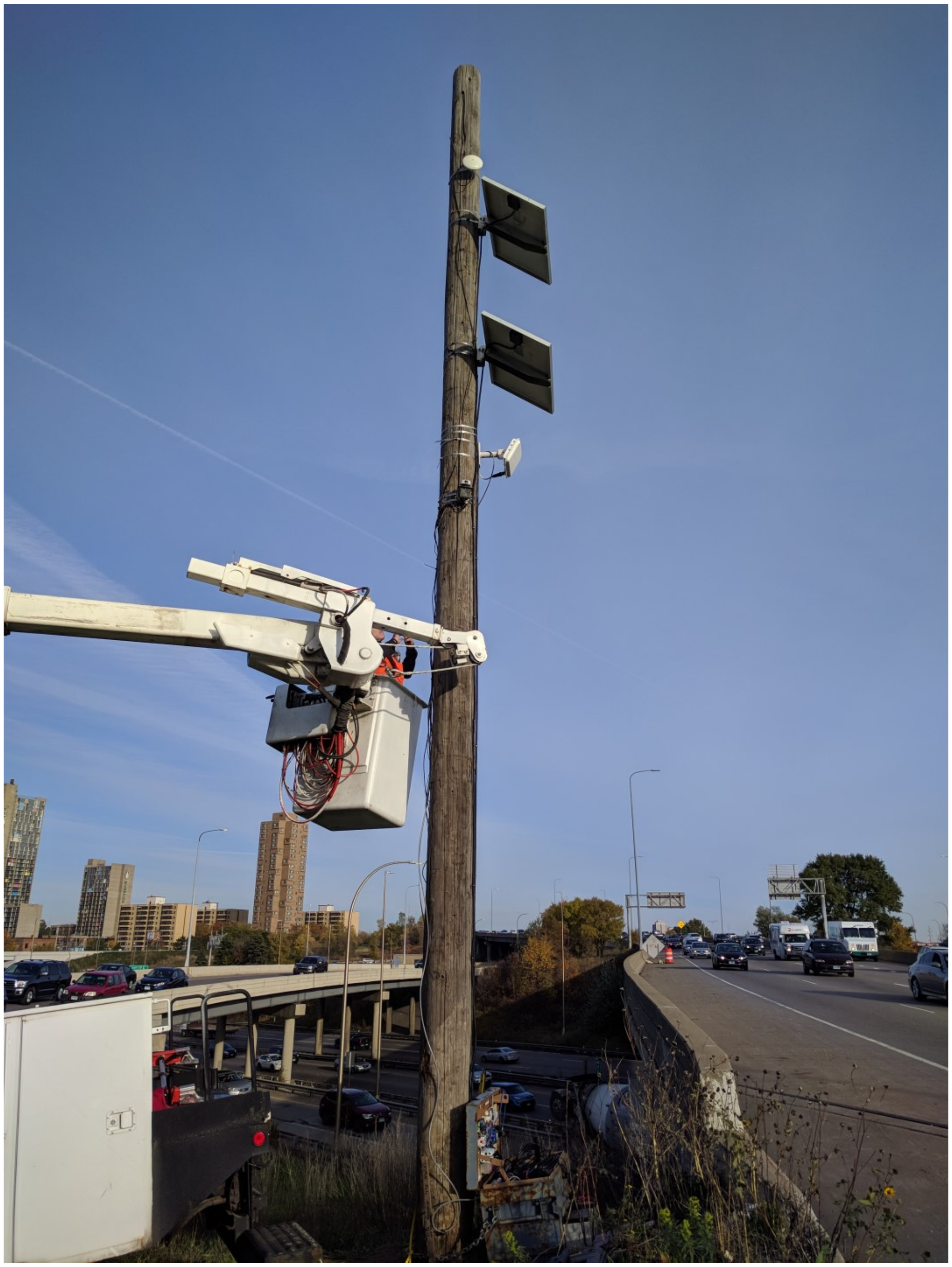


Figure 2.3 Hiawatha station being assembled.

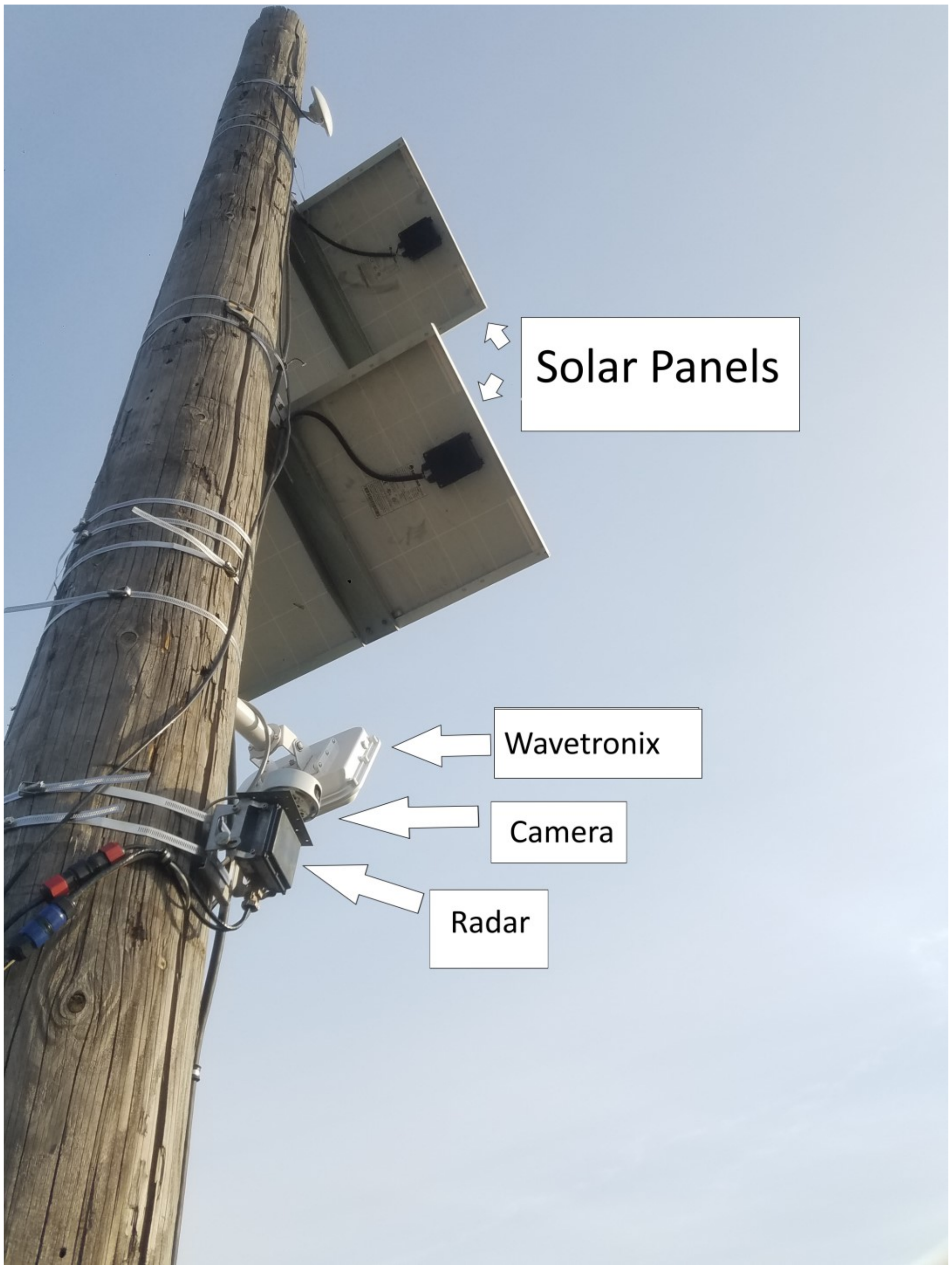
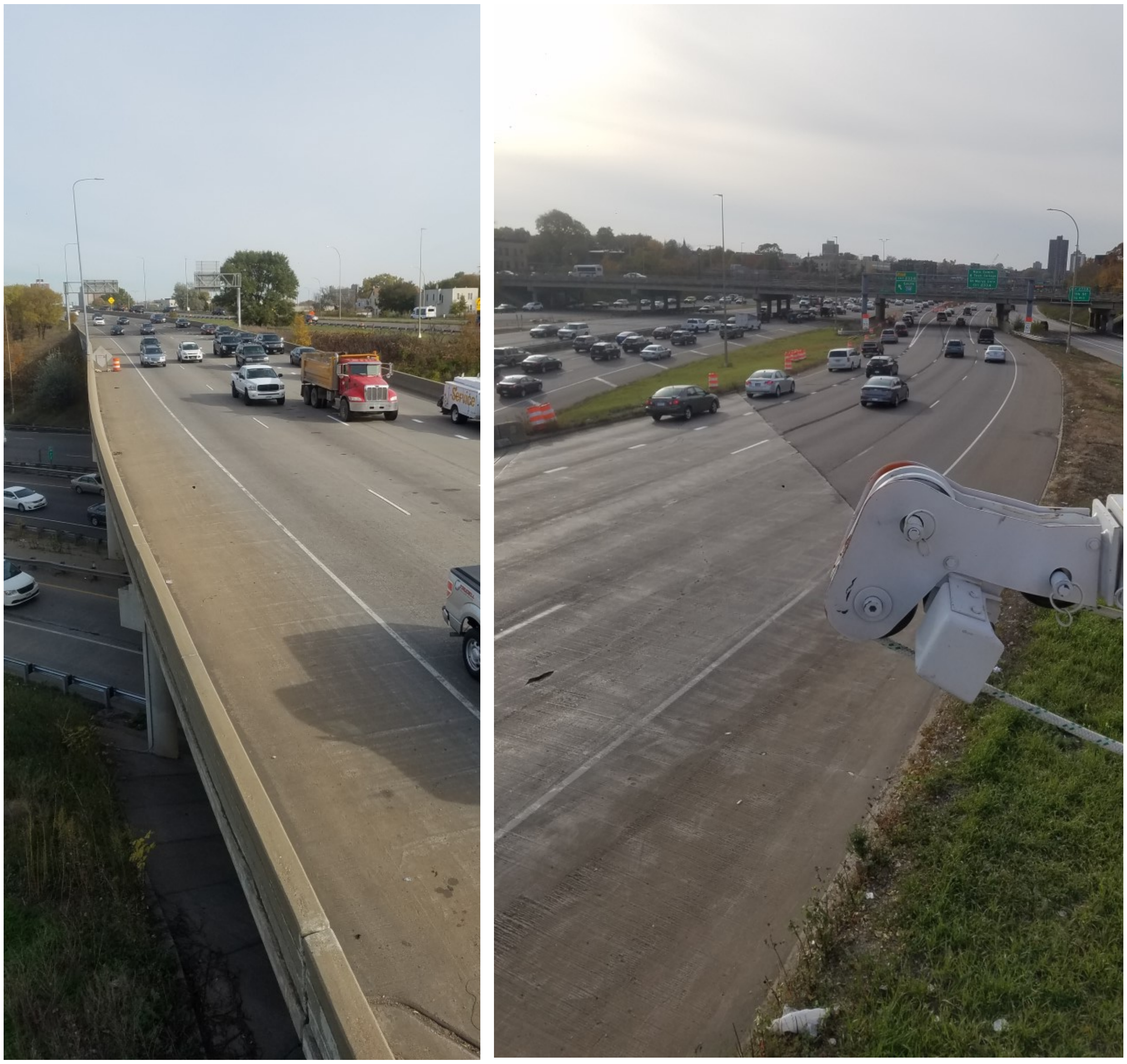


Figure 2.4 Hiawatha station closeup, with components labeled.



(a) (b)

Figure 2.5 View from Hiawatha station, looking east (a) and west (b) (bucket arm from assembly in view).

For illustration, Figure 2.6 shows the rooftop stations which are mounted and connected to power. Figure 2.7 shows MTO engineer Gordon Parikh working on the Augustana station.

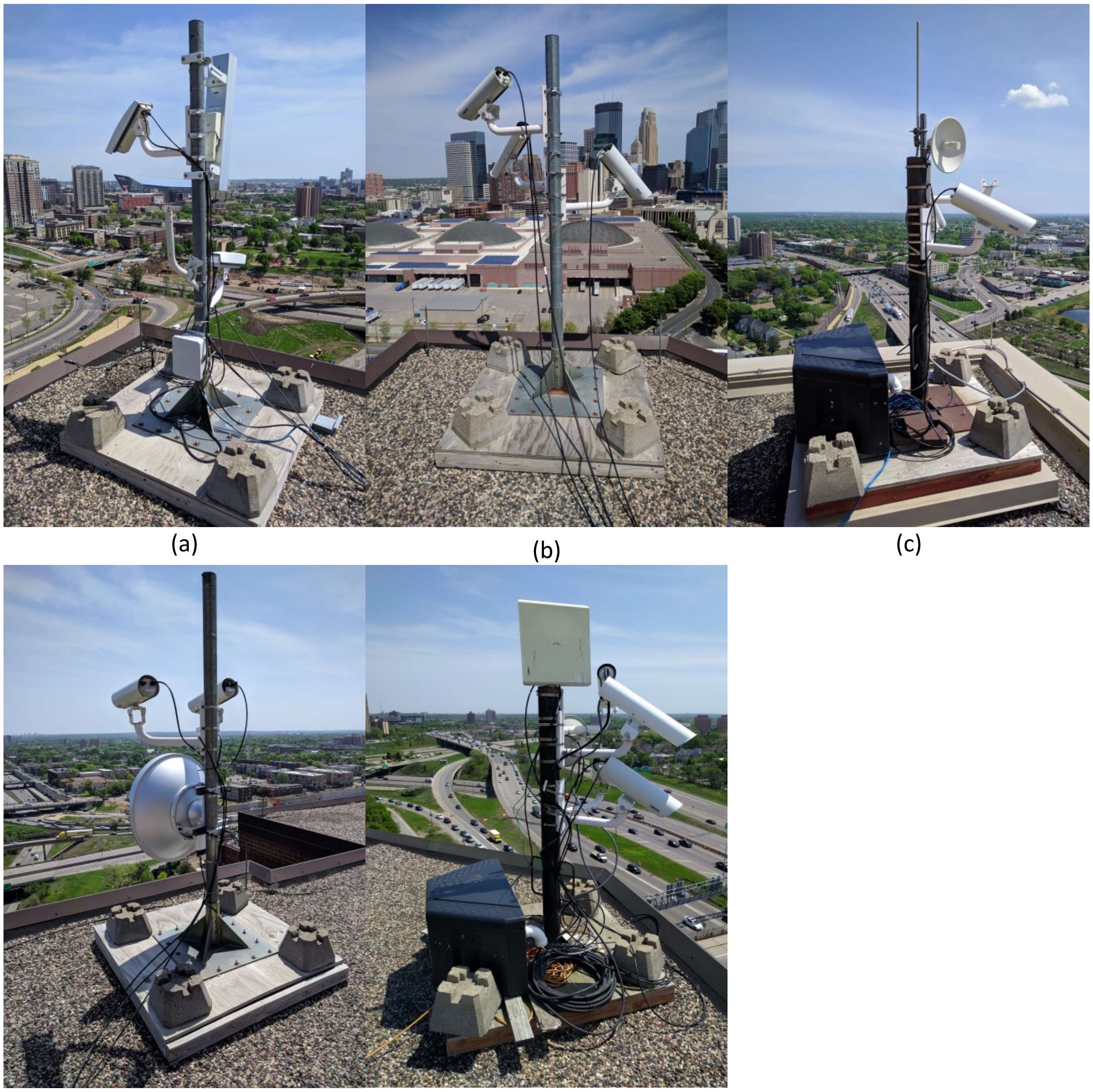


Figure 2.6 Rooftop stations (a)-(b) along 3rd , (c) Cedar Ave, (d) 3rd Ave and (e) Augustana

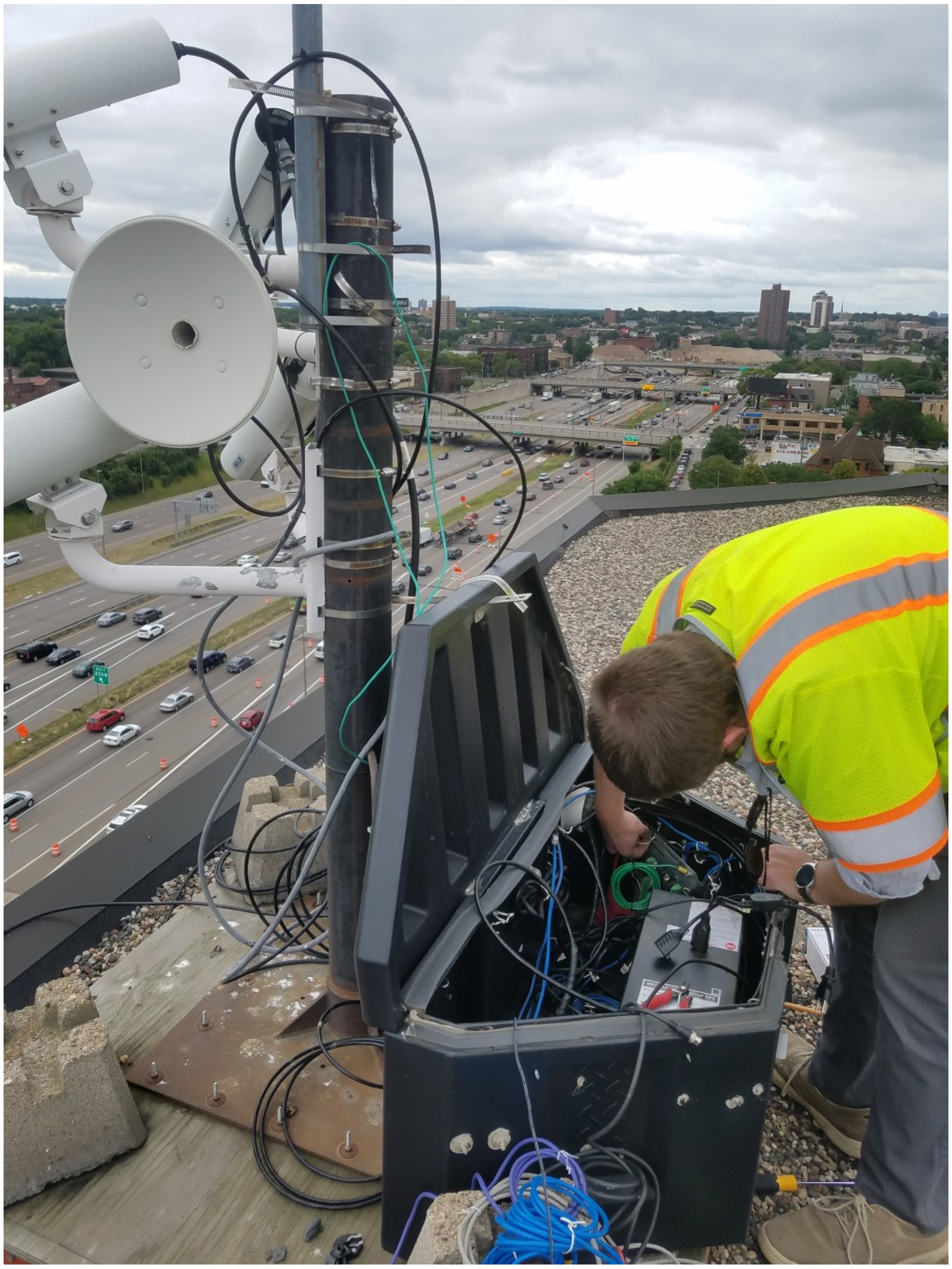


Figure 2.7 MTO engineer Gordon Parikh repairing the Augustana station

Figure 2.8 shows the retrieval of the Park Ave sensor. Examining the background shows the sensors really were in place right up until the construction reached them.

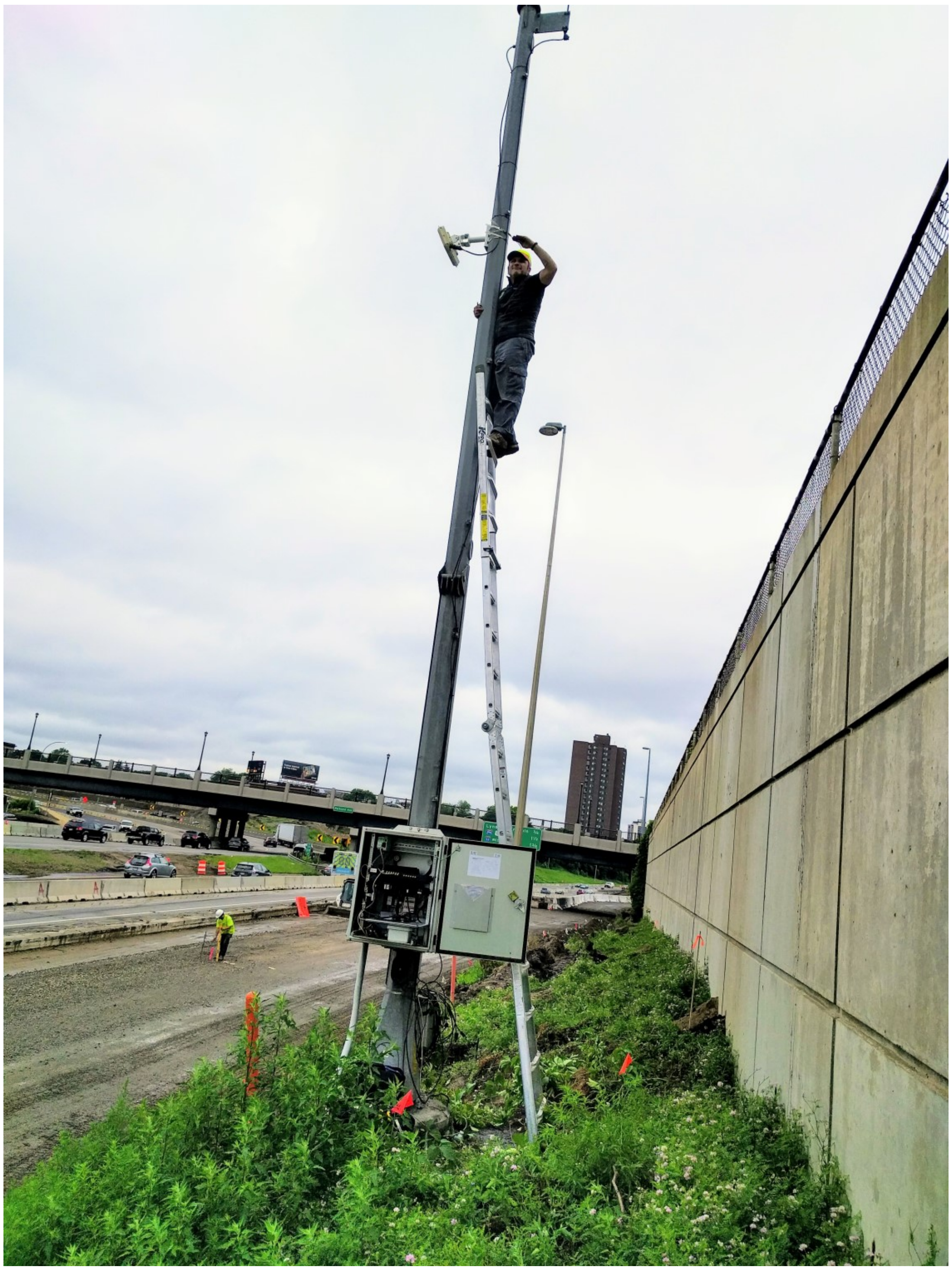


Figure 2.8 Retrieval of the Park Ave Sensor

Figure 2.9 shows the trailer station being set up.

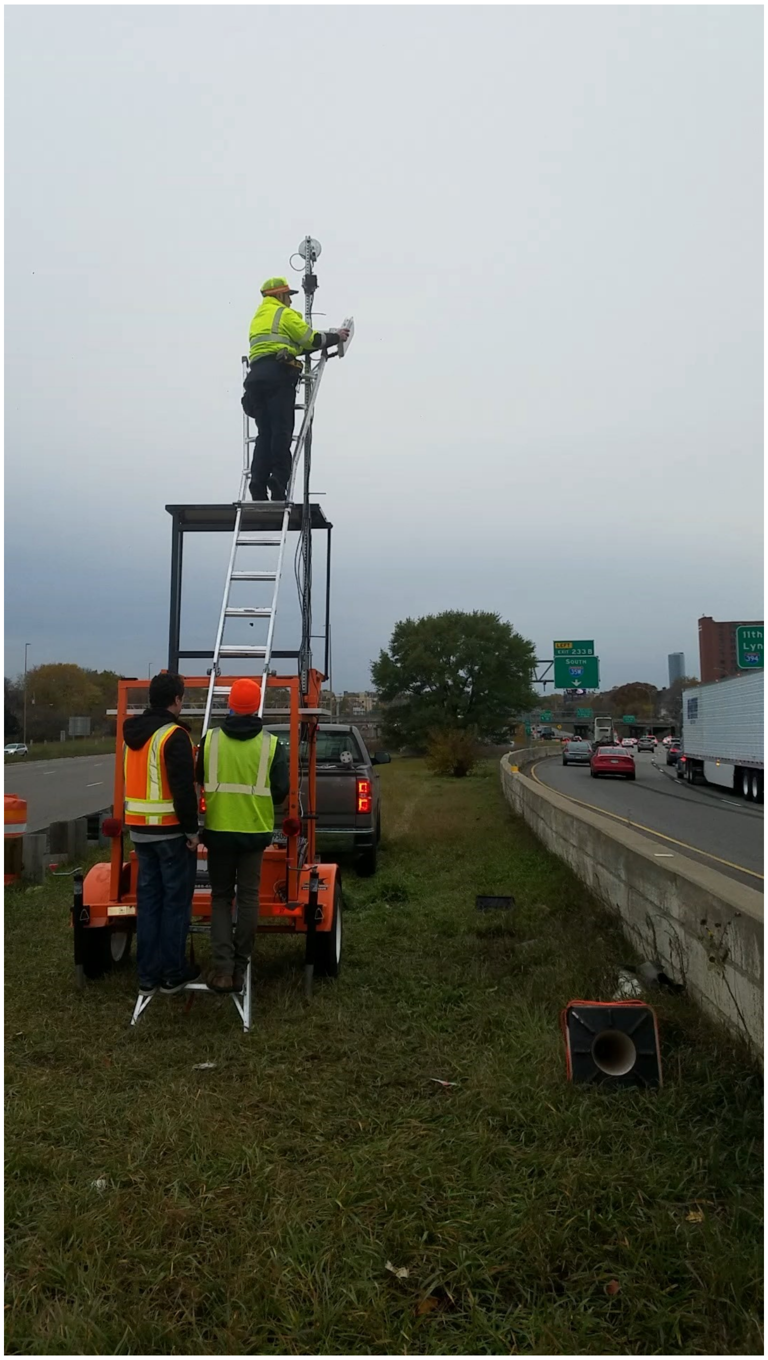


Figure 2.9 Deployment of the TH 52 trailer.

A day was spent examining a potential other new site on 7th Street South, as MnDOT was planning to put up a station of their own. Figure 2.10 shows the planned MTO site.

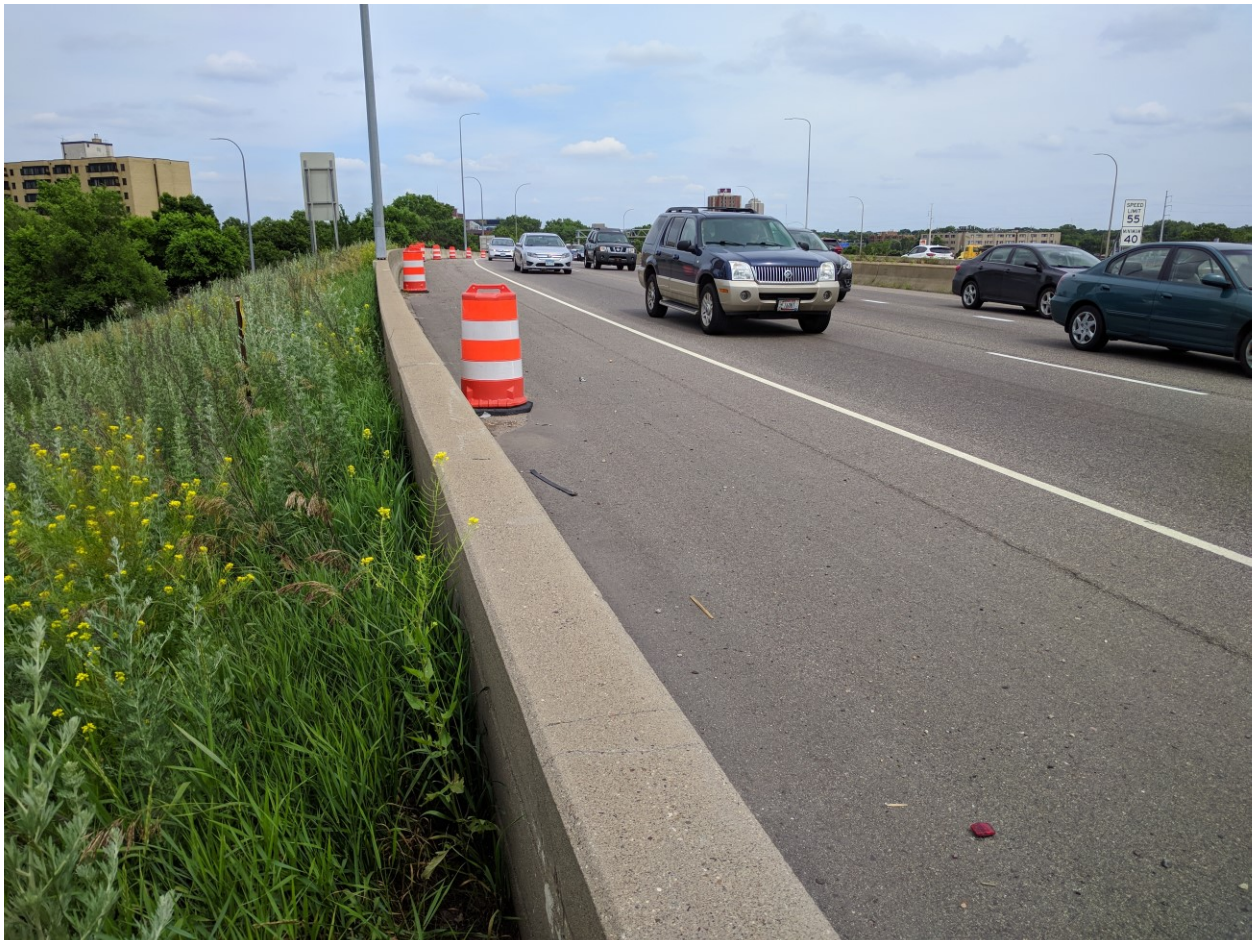


Figure 2.10 Potential site on 7th St South.

MnDOT ultimately decided against putting infrastructure there, so MTO did not add a station.

2.3 SENSING, SURVEILLANCE AND COMMUNICATION EXPANSION

As mentioned in Section 2.2, the Hiawatha Ave station was deployed with a new, upgraded radar station called Wavetronix. Wavetronix can capture more than 64 unique simultaneous objects, which the Type 29 and Type 30 radars along the original Testbed maxed out at. Wavetronix also has a longer range; depending on the style of Wavetronix, anywhere from 6 ft to 600 ft. These stations are using Wavetronix SmartSENSOR HD, which has an average range of 250 ft. Currently, the Hiawatha station is not close enough to provide continuous coverage with the corridor radars. Wavetronix also detects the entire roadway approach, rather than lane differentiated data. Switching to Wavetronix captures higher resolution data and will make future applications like Queue Warning easier to implement than Type 29 or Type 30 radar. Figure 2.11 shows a close-up of the Wavetronix sensor at Hiawatha Ave.

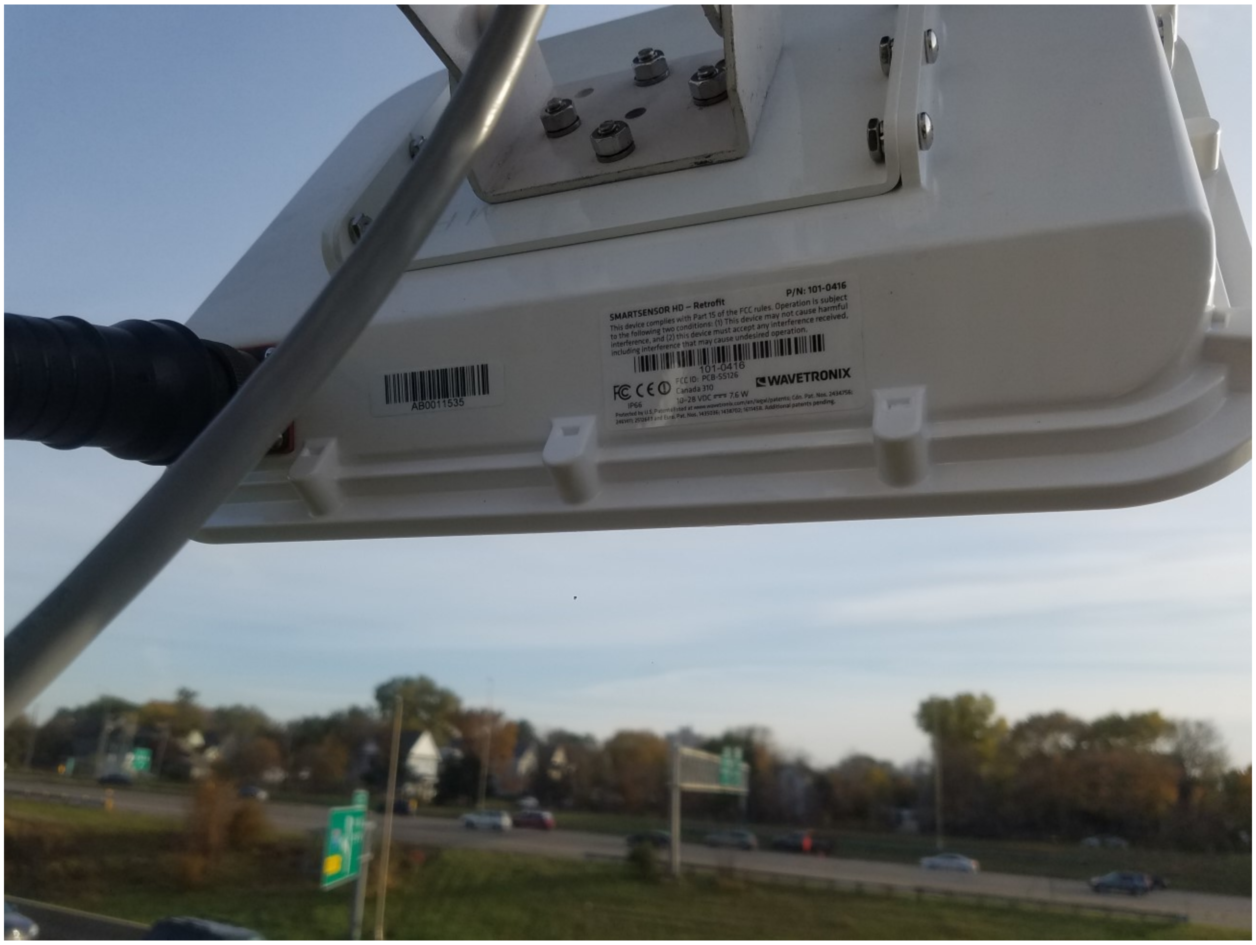


Figure 2.11 Close-up of Wavetronix sensor mounted at Hiawatha Ave.

In addition to the Hiawatha Ave station, MTO is now able to access a Wavetronix radar owned by MnDOT located on I-94 East between Portland and Park Avenues. The additional planned station east of I-35 North will have Wavetronix capacity as well.

Cellular communication has been implemented as a backup communication to the stations, and a secondary way to power cycle and reset the station radio in case of malfunction.

CHAPTER 3: SOFTWARE UPGRADES AND MAINTENANCE

In the Testbed layout, both video footage from camera stations and radar sensors collected data. The data was transmitted along a wireless communication network, utilizing existing rooftop nodes in the MTO's I-94 Field Lab. Figure 3.1below shows how the Roadside Stations (in the original configuration, but the system remains true for the new geometry of the Testbed) are able to transmit data back to the MTO.

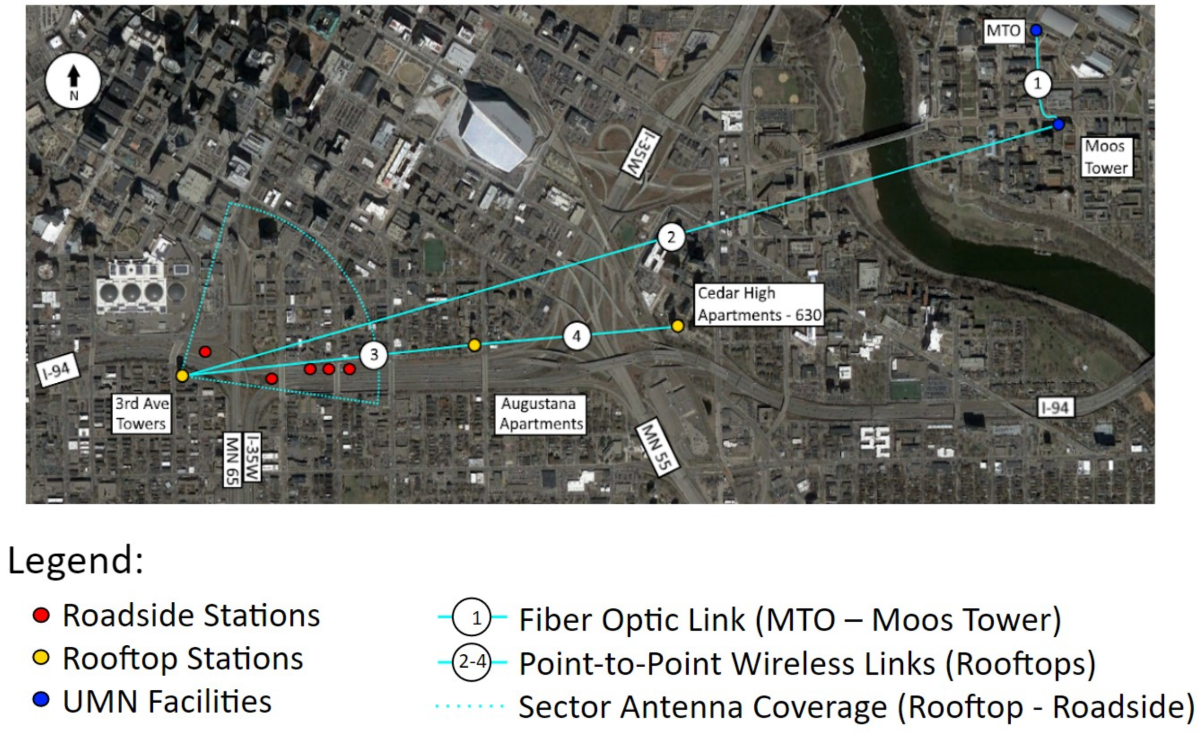

N
MTO
1
Moos Tower
I-35W
2
Cedar High Apartments - 630
3
4
I-94
3rd Ave Towers
I-35W MN 65
Augustana Apartments
MN 55
I-94
Legend:
● Roadside Stations
● Rooftop Stations
● UMN Facilities
1 Fiber Optic Link (MTO – Moos Tower)
2-4 Point-to-Point Wireless Links (Rooftops)
Sector Antenna Coverage (Rooftop - Roadside)

Figure 3.1 Field data retrieval communication system

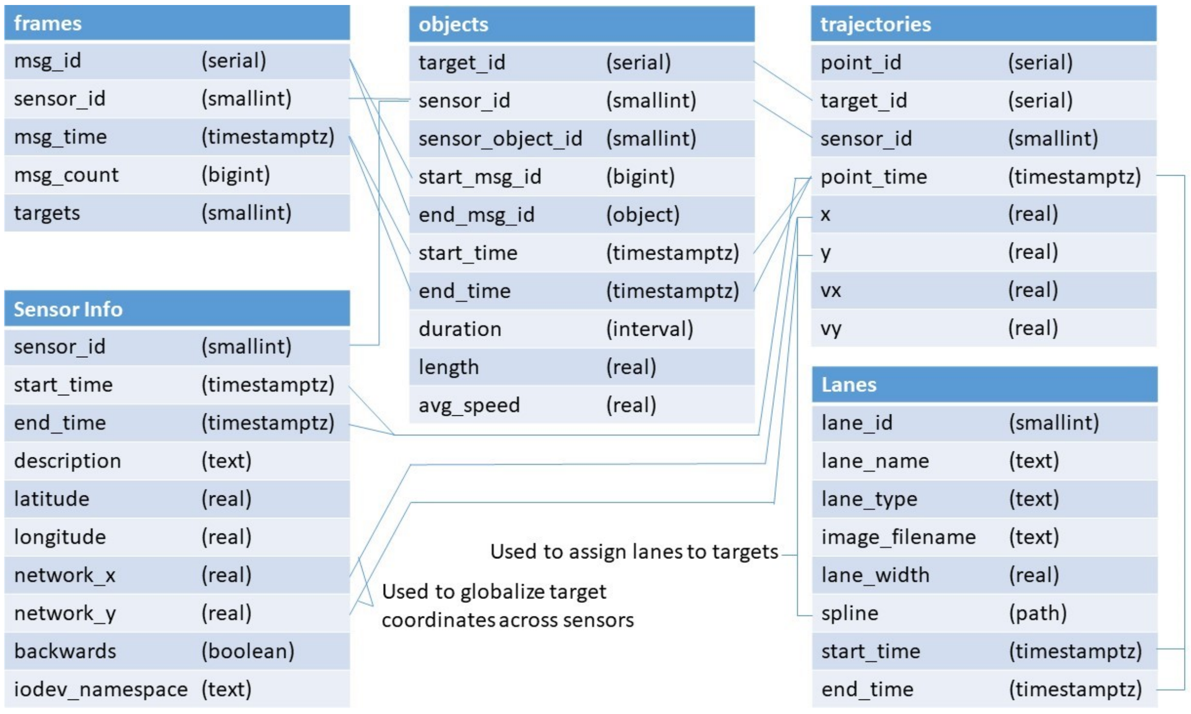
Before discussing software upgrades and maintenance, it is important to understand the system architecture of the MTO's database, which allows access to historical and near-real-time data critical to the development of software discussed in Section 3.4 below.

3.1 DATABASE STORAGE PROCESS

Once the data arrives back at the MTO, generally within one second, it is stored in a PostgreSQL database in real time. This database also contains all historical data from the sensors. The data is saved exactly as it is output by the sensor driver without modification to preserve the original state, allowing further processing to be done without changing the raw data. In addition to the raw sensor data, the database also contains historical and current position and orientation data for each of the sensors, providing an automatic means for combining data from multiple sensors into a single reference frame.

Finally, the database contains lane definitions for the corridor in the reference form of the sensors, allowing target data to be placed into a lane for analysis.

Because single sensor will produce several million target measurements in a single day, data from the sensors is split into individual tables for each date to reduce the maximum query time. Data is organized into three tables for each date to provide different levels of resolution to analyze the data. At the highest resolution available, in the “trajectories_<date>” table, each record consists of a single target measurement, with an X and Y position and velocity value for each target at each point in time. One level further out, in the “objects_<date>” table, each record consists of a single target as viewed by one sensor over its entire life, containing information like the time of its first and last measurements, the estimated length of the target, and its average speed. Finally, in the “frames_<date>” table, information about the sensor frames that were decoded is provided, including the message count from the sensor and the number of targets observed during that instant in time (up to 64). Together, these tables provide a complete picture of the data read from the sensor while reducing the amount of redundant information stored by the database. A diagram showing these tables and their relationships with each other is shown in Figure 3.2.



frames	
msg_id	(serial)
sensor_id	(smallint)
msg_time	(timestamptz)
msg_count	(bigint)
targets	(smallint)

Sensor Info	
sensor_id	(smallint)
start_time	(timestamptz)
end_time	(timestamptz)
description	(text)
latitude	(real)
longitude	(real)
network_x	(real)
network_y	(real)
backwards	(boolean)
iodev_namespace	(text)

objects	
target_id	(serial)
sensor_id	(smallint)
sensor_object_id	(smallint)
start_msg_id	(bigint)
end_msg_id	(object)
start_time	(timestamptz)
end_time	(timestamptz)
duration	(interval)
length	(real)
avg_speed	(real)

trajectories	
point_id	(serial)
target_id	(serial)
sensor_id	(smallint)
point_time	(timestamptz)
x	(real)
y	(real)
vx	(real)
vy	(real)

Lanes	
lane_id	(smallint)
lane_name	(text)
lane_type	(text)
image_filename	(text)
lane_width	(real)
spline	(path)
start_time	(timestamptz)
end_time	(timestamptz)

Used to assign lanes to targets
Used to globalize target coordinates across sensors

Figure 3.2 Database diagram depicting schema of trajectory data from sensors.

In addition to the data tables, there is another table containing both historical and current sensor position and orientation information. In this table, each record contains the sensor ID, the latitude, and longitude of the sensor, the X and Y coordinates of the sensor relative to the first (most-downstream) sensor in the network, and the start and end time for which this configuration is valid. For the current configuration, the end time is left blank to indicate that it is still in place. Additionally, the direction of the sensor (upstream or downstream) is also included, since it is necessary to understand the coordinate system of the sensor relative to the others; while the sensors contain their own orientation configuration that is used to translate ranges and angles into XY coordinates internally, the direction relative to moving traffic is still needed.

The table containing lane information uses an integer ID for each lane, with lane 0 starting at the right-most edge of the road. Lanes are also given a name and designated as either a main or auxiliary lane, as one of the lanes in the corridor exits midway through the sensor installation. The geometry of the lanes is indicated by a spline representing one edge of the lane along with a lane width. This allows targets to be sorted into a lane at each point in time by measuring their distance from the spline and determining if it is within the lane, based on the lane width. Lanes are also defined for a period of time, which allows the system to accommodate changes in road geometry that would occur due to construction. This configuration is utilized by the visualization tools developed during this project.

To simplify the use of the sensor position/orientation table and lane information table for adding information to raw sensor data, a number of SQL functions are also included in the database. These functions use the sensor positions/orientations to transform target coordinates into a global reference frame that is uniform across the network of sensors and use the lane information to assign a lane to each target at each point in time based on that translated position. This allows users to merge the data from multiple sensors and add this lane information in a single query without having to worry about the details of where the sensors or lanes were located at any particular point in time.

3.1.1 Inter-Process Communication for Exchanging Data

In addition to the PostgreSQL database, data is routed through a propriety Inter-Process Communication (IPC) system, meant to decode and share sensor data for time-critical safety applications. The data fields used for radar data, along with their data types, are listed in Table 3.1. These fields are used by software applications to produce visualizations of the data, discussed further in Section 3.4

Table 3.1 Sensor data fields available in IPC data

Field Name	Data Type	Description
time	Double-precision float	Unix timestamp with microsecond resolution
count	Unsigned integer	Cycle count for the radar
status	Integer	Status indicator
targets	Integer	Number of targets in message (up to 64)
id[64]	(Array) Unsigned short integer	Array to hold ID of each target

x[64]	(Array) Float	Array to hold X position of each target
y[64]	(Array) Float	Array to hold Y position of each target
vx[64]	(Array) Float	Array to hold X component of velocity of each target
vy[64]	(Array) Float	Array to hold Y component of velocity of each target
relays[16]	(Array) Boolean	Array to hold status of relay triggers

By abstracting sensor data in this way, the IPC system facilitates quicker development of applications using the data, allows multiple applications to use that data without the potential to interfere with one another, and provides a number of tools to extend the functionality of the system without requiring additional development. While the driver application itself only runs in real-time, making applications responsible for buffering data themselves, the IPC framework comes with utilities for saving data into a binary file and replaying it later. This feature is useful for development, testing, and debugging in that it allows applications to use sensor data without requiring an actual sensor thereby allowing specific conditions to be recreated without complex testing setups.

Figure 3.3 below summarizes the system architecture as a flowchart, for reference.

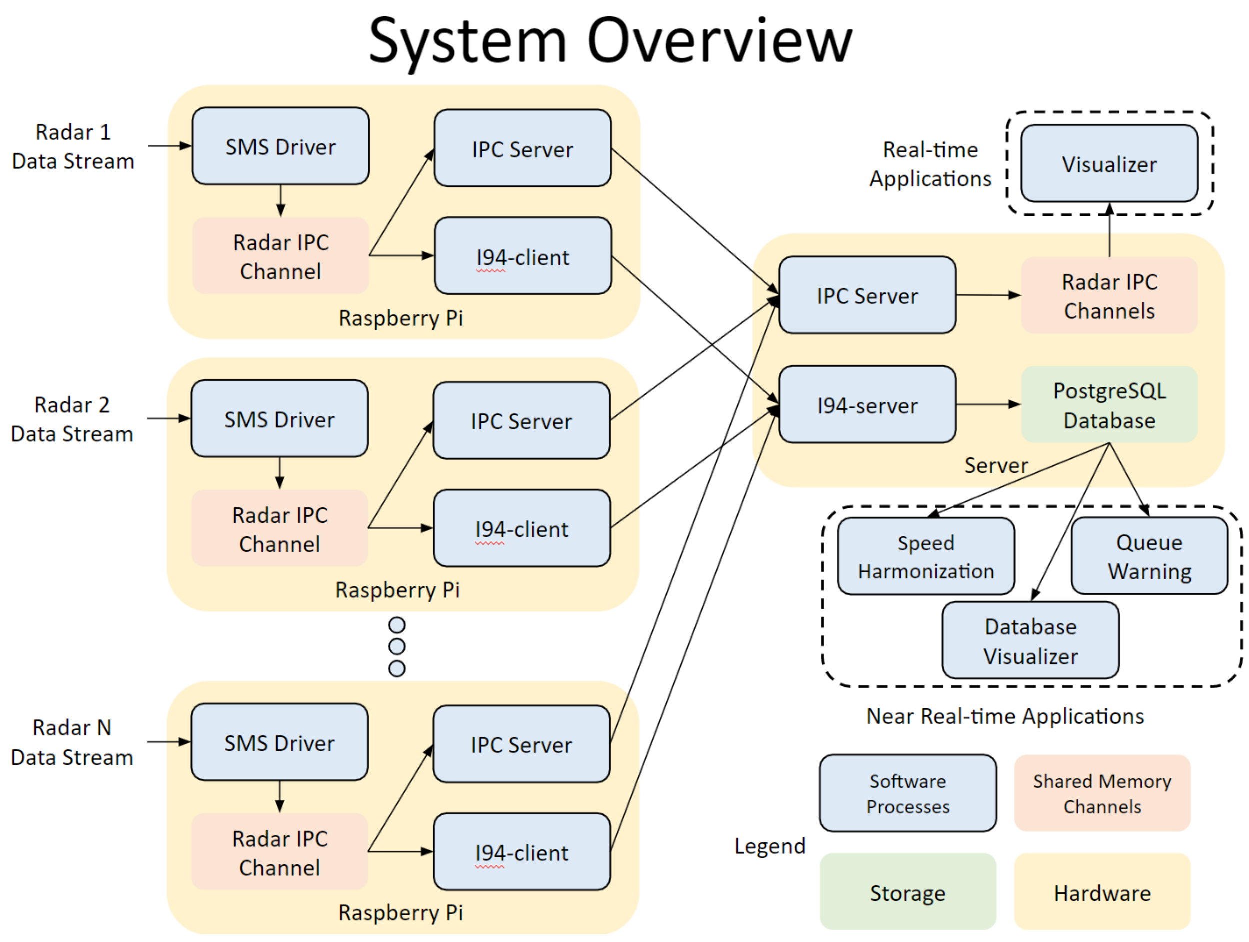


Figure 3.3 Data collection system architecture

All maintenance and upgrades were performed utilizing this system architecture.

3.2 SYSTEM AND DATA SECURITY

The MTO, and by extent the Testbed, is the target of frequent hacking attacks. In 2017 alone, the MTO had to upgrade the database server as well as some of the radio firmware because security flaws were exploited; the University of MN IT Services cut communications until the breach was repaired. During the maintenance period, the current software infrastructure of the station computers and MTO database were inspected and upgraded as necessary to ensure system security.

The upgrades done regarding this project have to do with the creation of tools to view and catalog radar data. Most significant of these tools is a space-time diagram creator that is configured to work with the database warehousing the aforementioned data.

When deployed, all internet-connected CV testbed data collection devices are configured to only use and communicate with computers on the MTO's subnet. For remote access, these devices can be communicated with through the MTO's virtual private network.

Internet access for CV testbed data collection devices is provided through point to point and point to multipoint radios that broadcast through a fiber optic network access point. This network access point is provided by the University of Minnesota, the same provider used by the MTO.

3.3 DEVELOPMENT OF SYSTEM API

Under the original project, a set of Python tools were developed to read the database data under a number of circumstances and perform some of the post-processing that would be required by most applications. These tools are optimized to work in real-time (or simulated real-time), reading data from the database in a separate process to make it available to applications without making the main process wait for the relatively slow query operations. They also abstract the process of reading data from the database, instead providing access to the data via a virtual sensor object that translates raw data into objects with an intuitive application-programming interface. This allows application developers to write applications that use sensor data without needing to write queries for reading the data themselves.

These Python tools were upgraded to Python 3 during the maintenance period and used during the development of visualizing software.

3.4 DEVELOPMENT OF DATA FILTERING AND DATA MINING APPLICATIONS

As discussed in Section 3.1 the CV Testbed captures and saves vehicle trajectories in a raw, sensor-by-sensor format. This was adequate for the scope of the original project. Going forward, many CV applications such as BSM Emulation will require the production of seamless trajectories as well as filling gaps in the records due to sensing issues, and elimination of duplicate and erroneous data.

For both the IPC and database interfaces, the first applications that were developed to demonstrate the functionality were visualizers. These are simple, graphical programs that display data from multiple sensors in a plan view, allowing users to see the data that is collected from the radar in an intuitive way. The programs show that sensors were either working or not during any selected time periods. The IPC-based viewer is very rudimentary, only combining the data from multiple sensors and compensating for the position and orientation of sensors.

3.4.1 IPC Visualizer

The visualizer is a program that displays the most recent sensor measurements by reading information over IPC. It is capable of plotting the positions of the tracked targets and could also be used to plot additional information that is generated by the processing algorithm thereby providing a means to verify the operation of sensors and any applications. The IPC visualizer has a simple interface, only showing targets plotted on a white background and colored by sensor (seen in Figure 3.4). Each target's position is shown as a dot. The differing colors come from different radar sensor data. The boxes around the few dots are a result of selection for output. This tool is useful for real-time validation of data.

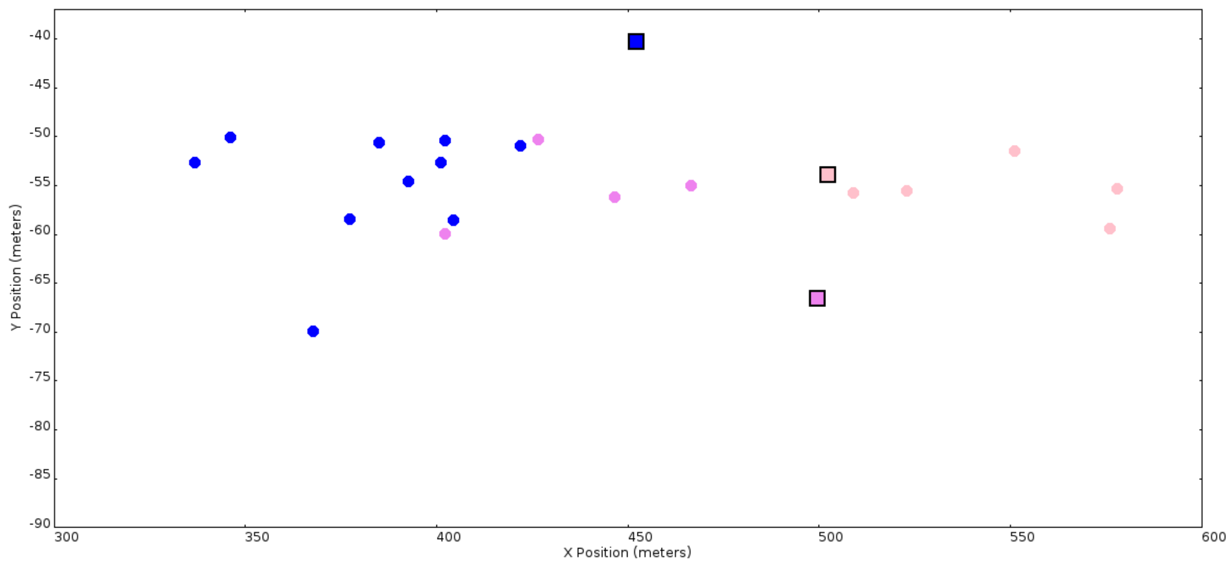


Figure 3.4 Screenshot of the IPC visualizer application

The viewer requires a configuration file to define the locations and orientations of sensors in the field. Data from sensors is read from local IPC channels meaning that, for real-time operation, the IPC server must be used to synchronize data across the network if it is running on a remote computer and to include data from multiple sensors. Because the IPC system also allows data to be recorded and played back later, the visualizer can also read data as it is replayed from these files though it is unable to control the playback.

3.4.2 Database Visualizers

Compared to the IPC visualizer, the database visualizers are much more feature-rich. The first database visualizer developed is capable of playing the locations of targets in near-real-time as data is written to the database, and also provides the ability to seamlessly replay historical data by simply providing a start time in the past. Data is also rendered over an aerial image of the site, which helps provide feedback to the user regarding the operation of the sensors (seen in Figure 3.5). It is designed for smooth playback of data, so it will occasionally pause to buffer data but this provides a more visually-appealing experience. In addition to this, the viewer allows the user to pause playback at a particular point in time and to record snapshots or video clips of the viewer output.

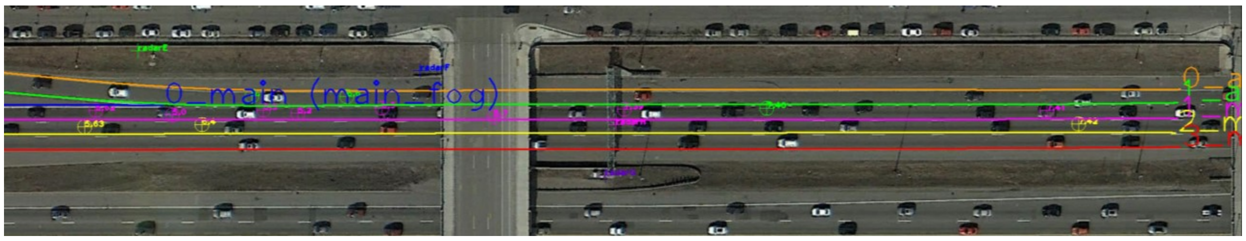


Figure 3.5 Screenshot of the database visualizer application, with targets and lanes over an aerial image

This visualizer requires a configuration file containing sensor positions and orientations but this file contains the same information used to adjust target positions in the database and can also be used to update this information in the database using a simple tool. Like the database, the visualizer also contains definitions of lanes which can be edited by the user using an associated graphical tool, and can sort vehicles into lanes as it plays. Targets can be colored either by the sensor that measured that data or by the lane that is assigned to them.

Currently, only the existing configuration file has been implemented for use in visualizing the raw data. The configuration file does result in an imperfect visualization at points, where vehicles are duplicated or dropped in the programs, or assigned to the wrong lane. Examples of these issues will be shown below. Future configurations and algorithms to interpret the raw data can be implemented to reduce these issues, once developed.

Building on the basic IPC and database tools, three new tools were developed to: see what sensors were active at any given time, visualize vehicle trajectories, and create space-time diagrams of vehicle travel along the Testbed. Full user manuals are available in the Appendices.

It is critical to note that these tools are based on the original alignment of the Testbed (i.e., before any of the disassembly or movement of stations). New configuration files will allow for the new geometry of the Testbed to be displayed in the visualizers, in the future.

3.4.2.1 Sensor Availability

The first tool is a Matlab-based application used to see what sensors were functioning at any point on a given day. The output is shown in Figure 3.6. The Y axis represents the seven sensors, and the X axis is time, which can be anywhere from a five-minute period to an entire day. The tool checks each sensor in the database on the desired day and time to see if it was recording valid data or not. The resulting heat map will look something like Figure 3.6, where a sensor that was functioning is green and any downtime is red.

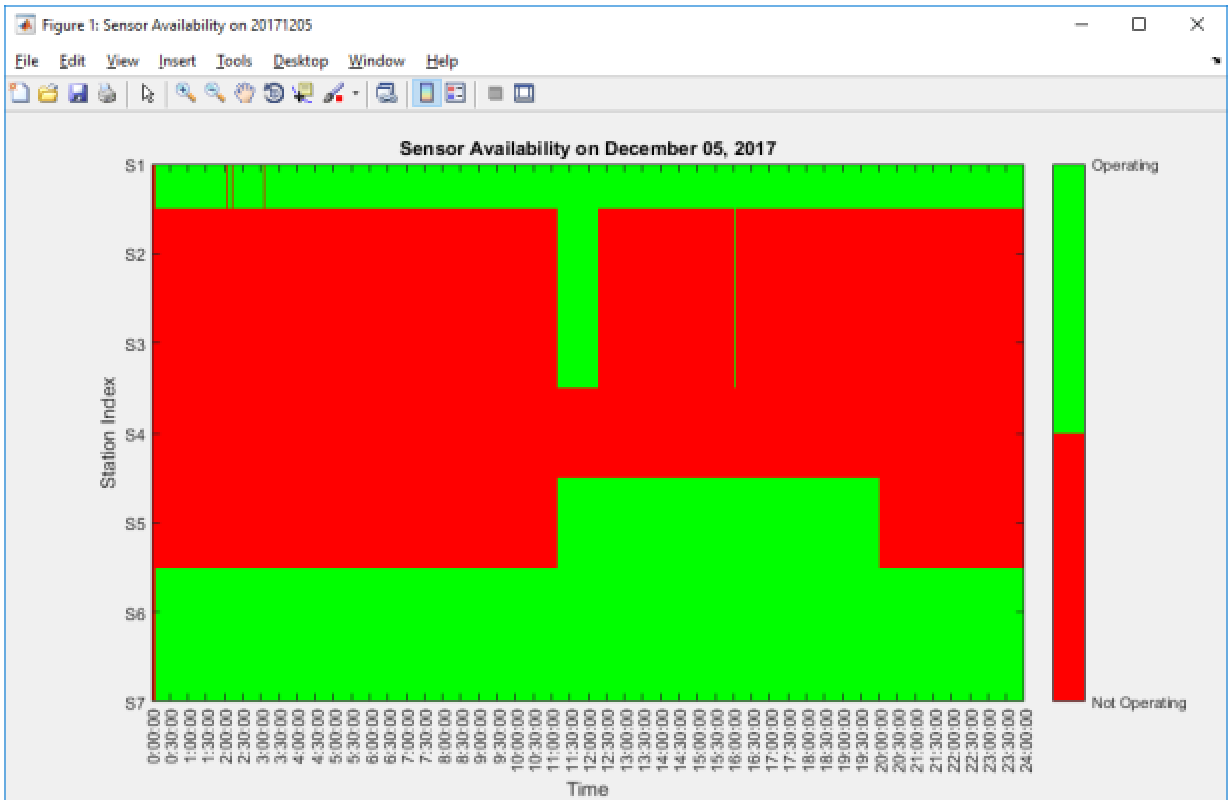


Figure 3.6 Sensor availability heat map.

The full user manual for this tool can be found in APPENDIX A
Trajectory Plotting User Manual.

3.4.2.2 Trajectory Plotting

The next tool is also Matlab based, and pulls vehicle location and velocity data from the database to create individual vehicle trajectory graphs. The user must choose which vehicle ID they would like to graph, as well as the time and day. The output of all four possible graphs is shown in Figure 3.7.

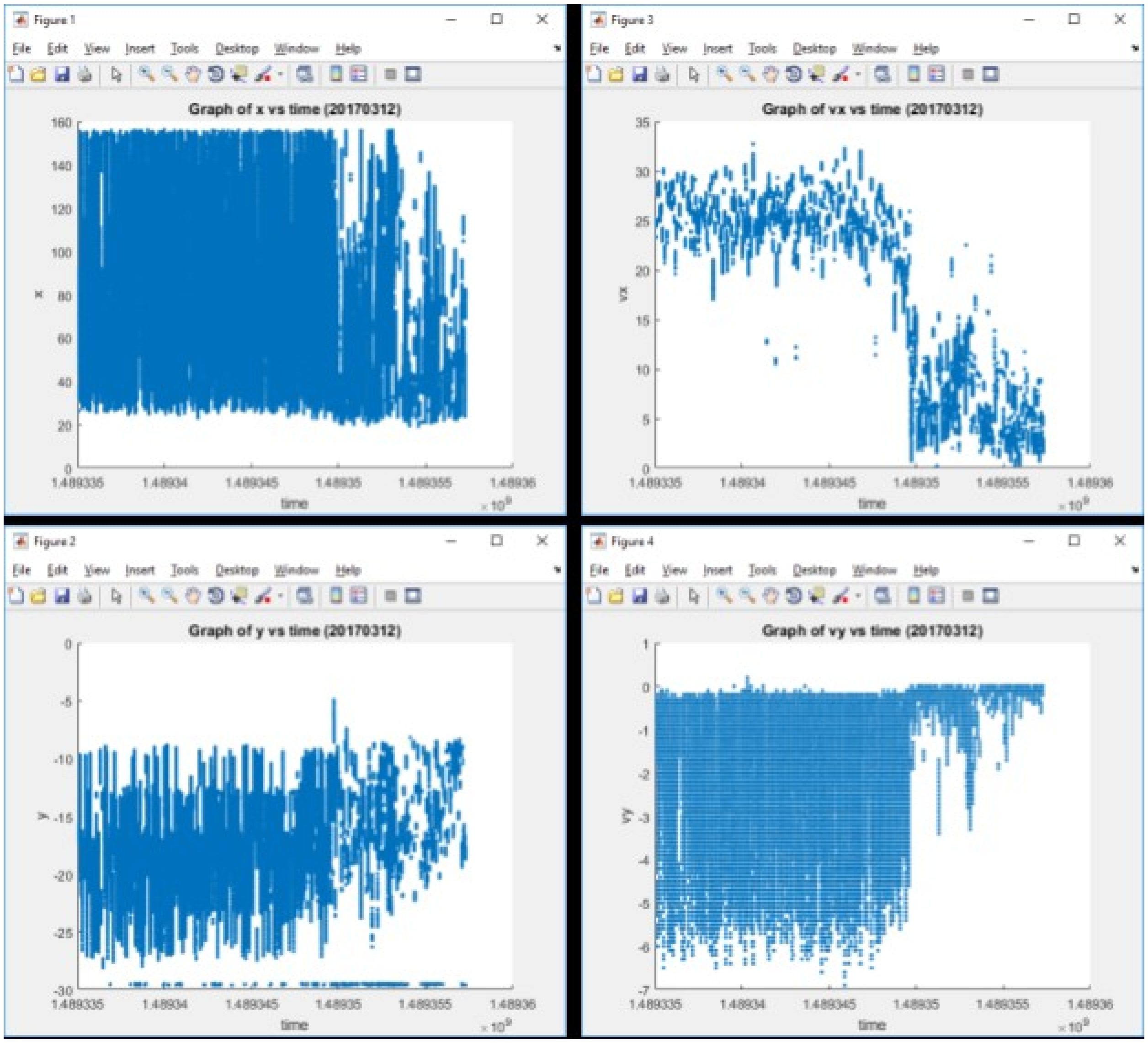


Figure 3.7 The vehicle trajectory plot outputs

There will be 4 graphs generated when all four checkboxes are marked in the user interface.

1. Graph of x vs time ({DATE}): The figure drawn are the vehicle's X distance coordinates in a local system relative to the radar versus time.
2. Graph of vx vs time ({DATE}): The figure drawn are the vehicle's velocity in the direction of X distance coordinates in a local system relative to the radar versus time.
3. Graph of y vs time ({DATE}): The figure drawn are the vehicle's Y distance coordinates in a local system relative to the radar versus time.
4. Graph of vy vs time ({DATE}): The figure drawn are the vehicle's velocity in the direction of Y distance coordinates in a local system relative to the radar versus time.

Figure 3.8 shows a more zoomed in graph of a vehicle's velocity in X relative to the radar versus time. The time is shown in a UNIX timestamp.

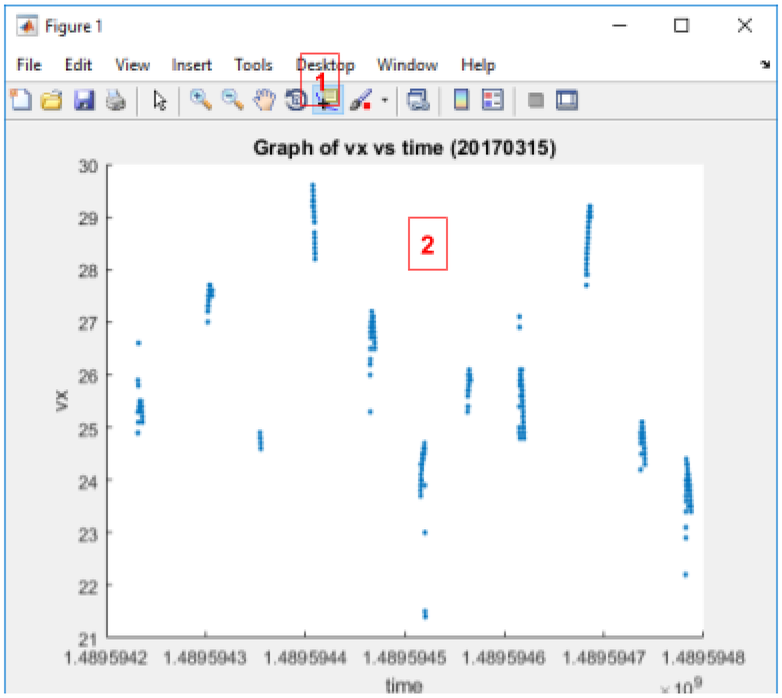
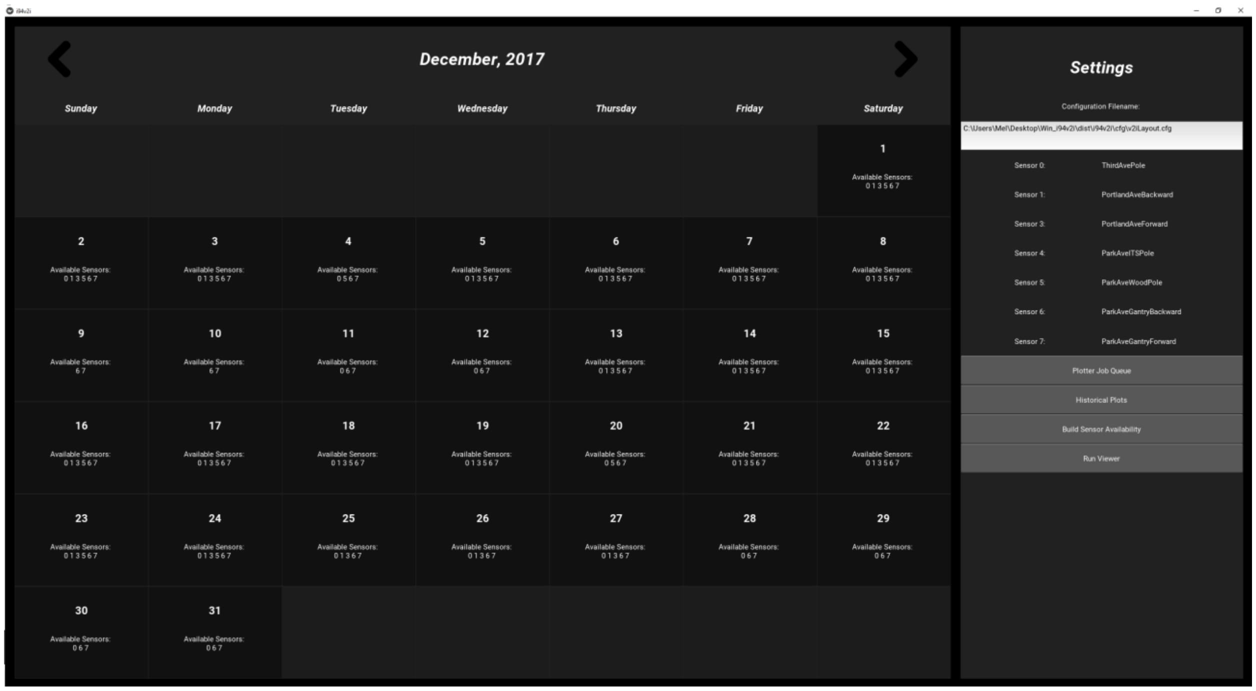


Figure 3.8 A closer look at the output graph of vx vs time

A full user manual for this software can be found in APPENDIX B Trajectory Plotting User Manual.

3.4.2.3 Graphical Interface

The most complex program developed during this project, the Graphical Interface utilizes database visualization in a variety of ways. When users first boot up the program, they are able to choose whether to use the existing configuration file or custom configuration. As mentioned in 3.4.2 , currently only the existing configuration file is available for use. Users will be taken to a home screen, as seen in Figure 3.9.


i94v2i

December, 2017

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
						1 Available Sensors: 0 1 3 5 6 7
2 Available Sensors: 0 1 3 5 6 7	3 Available Sensors: 0 1 3 5 6 7	4 Available Sensors: 0 5 6 7	5 Available Sensors: 0 1 3 5 6 7	6 Available Sensors: 0 1 3 5 6 7	7 Available Sensors: 0 1 3 5 6 7	8 Available Sensors: 0 1 3 5 6 7
9 Available Sensors: 6 7	10 Available Sensors: 6 7	11 Available Sensors: 0 6 7	12 Available Sensors: 0 6 7	13 Available Sensors: 0 1 3 5 6 7	14 Available Sensors: 0 1 3 5 6 7	15 Available Sensors: 0 1 3 5 6 7
16 Available Sensors: 0 1 3 5 6 7	17 Available Sensors: 0 1 3 5 6 7	18 Available Sensors: 0 1 3 5 6 7	19 Available Sensors: 0 1 3 5 6 7	20 Available Sensors: 0 5 6 7	21 Available Sensors: 0 1 3 5 6 7	22 Available Sensors: 0 1 3 5 6 7
23 Available Sensors: 0 1 3 5 6 7	24 Available Sensors: 0 1 3 5 6 7	25 Available Sensors: 0 1 3 6 7	26 Available Sensors: 0 1 3 6 7	27 Available Sensors: 0 1 3 6 7	28 Available Sensors: 0 6 7	29 Available Sensors: 0 6 7
30 Available Sensors: 0 6 7	31 Available Sensors: 0 6 7					

Settings

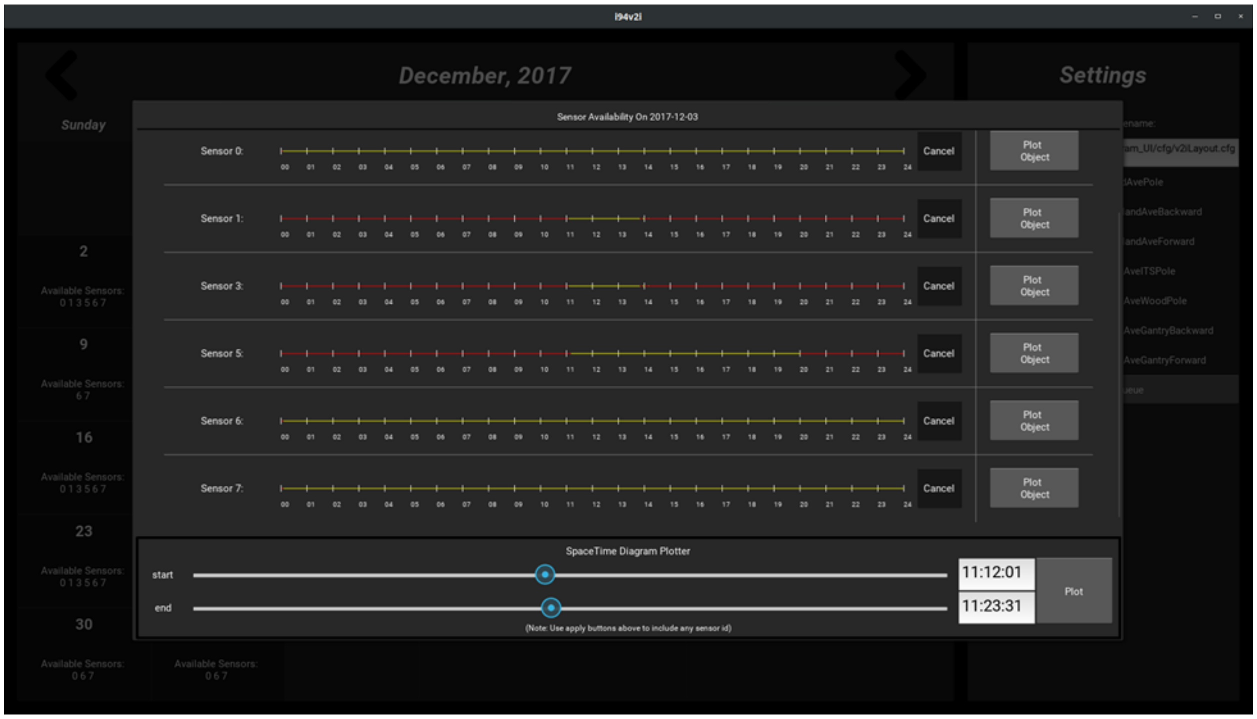
Configuration Filename:

C:\Users\Mel\Desktop\Win_i94v2i\dist\i94v2i\cfg\v2iLayout.cfg

Sensor 0:	ThirdAvePole
Sensor 1:	PortlandAveBackward
Sensor 3:	PortlandAveForward
Sensor 4:	ParkAveITSPole
Sensor 5:	ParkAveWoodPole
Sensor 6:	ParkAveGantryBackward
Sensor 7:	ParkAveGantryForward

Figure 3.9 Home screen of the Graphical Interface tool.

On this home screen, users can see broad sensor availability for a month at a time. Clicking the desired day, users will then be able to see what periods of time all the sensors were available and select the time frame for the output space-time graph.


i94v2i

Sensor Availability On 2017-12-03

```
Sensor 0:  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24   Cancel   Plot Object
Sensor 1:  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24   Cancel   Plot Object
Sensor 3:  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24   Cancel   Plot Object
Sensor 5:  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24   Cancel   Plot Object
Sensor 6:  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24   Cancel   Plot Object
Sensor 7:  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24   Cancel   Plot Object
```

SpaceTime Diagram Plotter

start 11:12:01

end 11:23:31

Plot

(Note: Use apply buttons above to include any sensor id)

Figure 3.10 Sensor availability per hour, and selection of time frame for output.

After the user has selected the sensors and time frame they wish to see, they will click plot and wait for the program to run (this can take anywhere from a few minutes to hours depending on the amount of data being pulled from the database). On completion, the program will display a space-time diagram of all vehicle raw data for that time. Figure 3.11 is an example of a time and day where all sensors were working. This figure has the right lane selected, but users can opt to select the left or middle lanes as well as zoom in to individual trajectories.

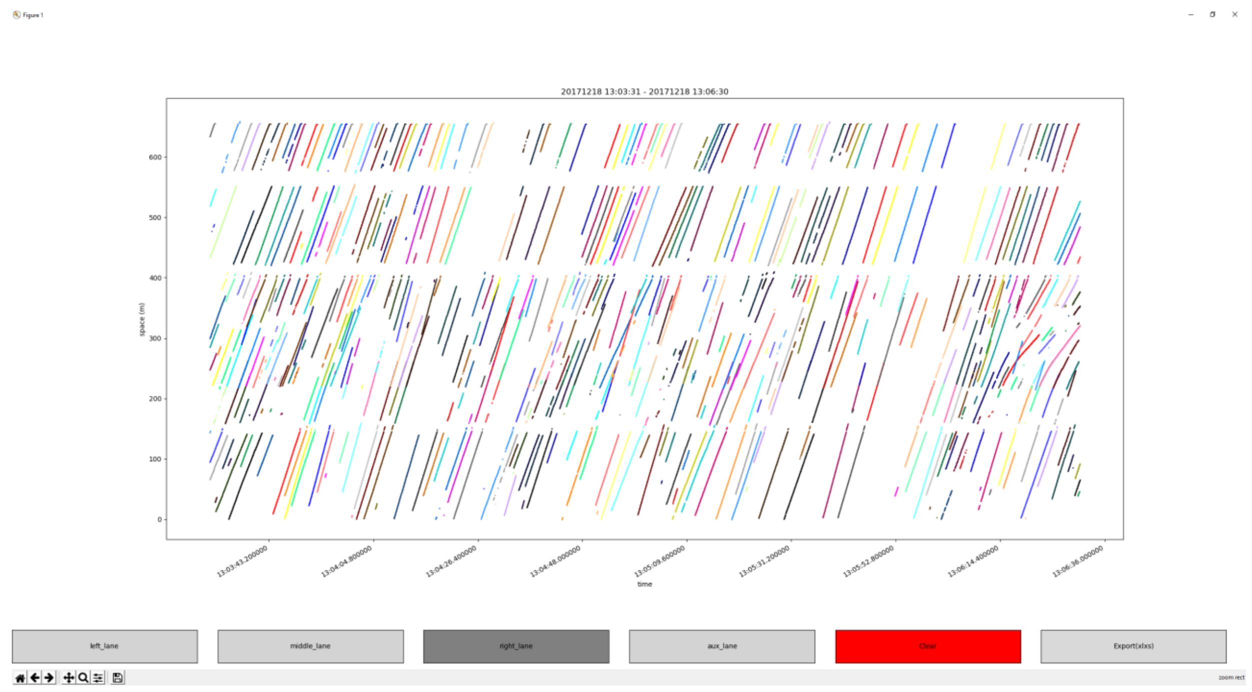


Figure 3.11 All sensor space-time diagram example, all sensors and little congestion.

Figure 3.12 (dynamic) shows a day with significant snowfall, resulting in several sensors not working and more congestion on the roadway.

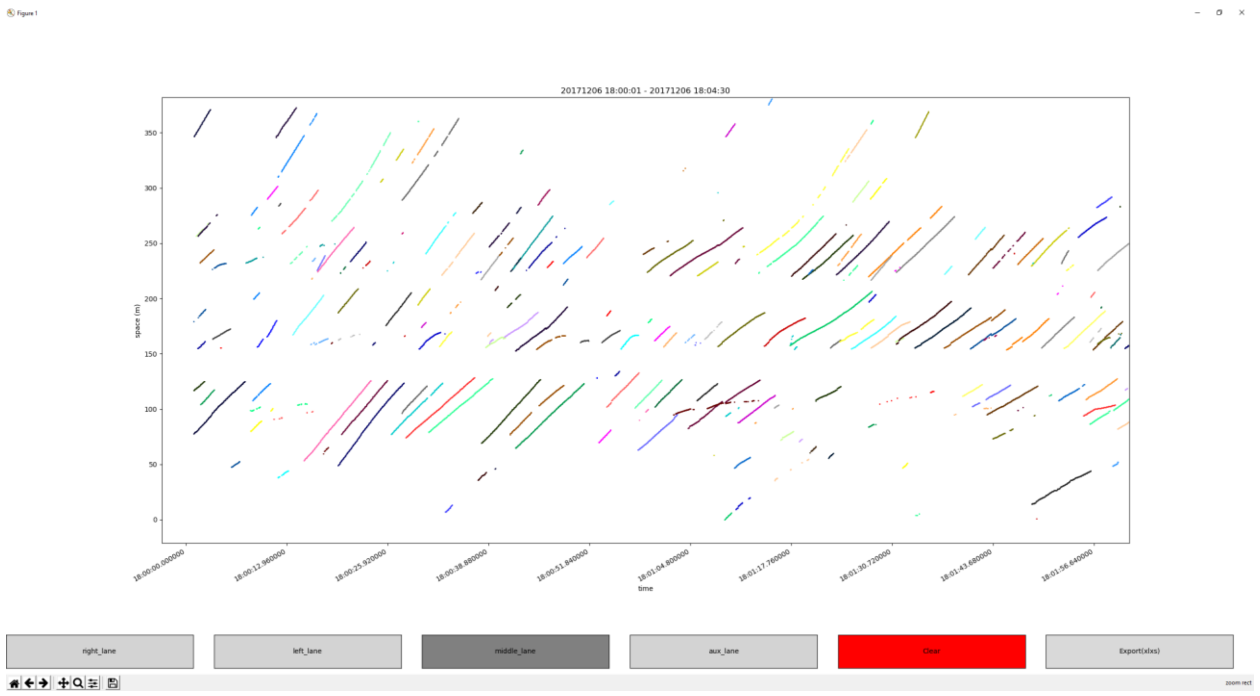


Figure 3.12 Snowy day, with some downed sensors and congestion.

As seen in Figure 3.11 and Figure 3.12, the vehicle trajectories are not continuous. Figure 3.13 breaks down what sensor data is being displayed in each portion of the graph. Figure 3.14 shows the sections of radar coverage on a map overlay.

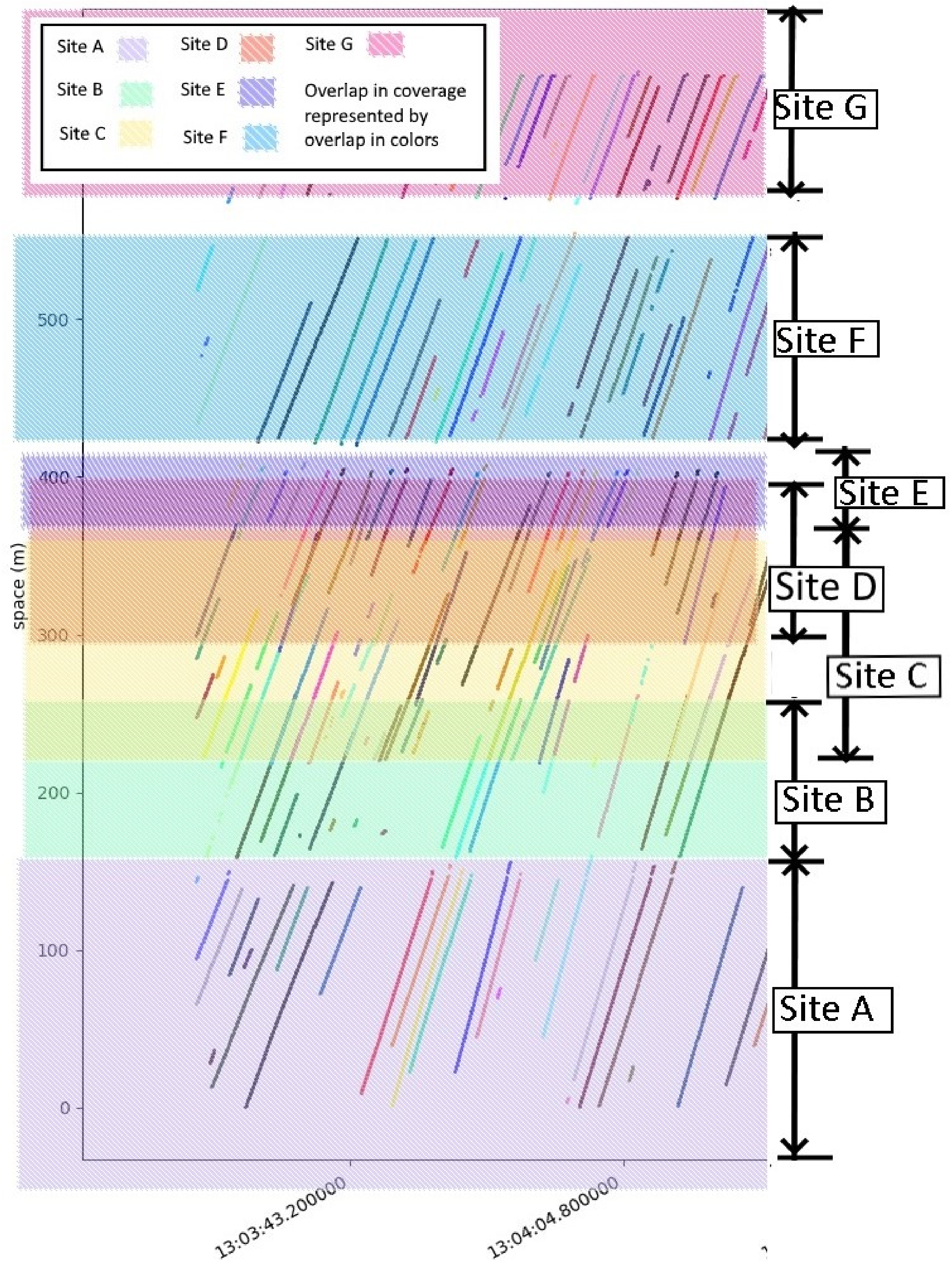


Figure 3.13 Space-time broken down by sensor.

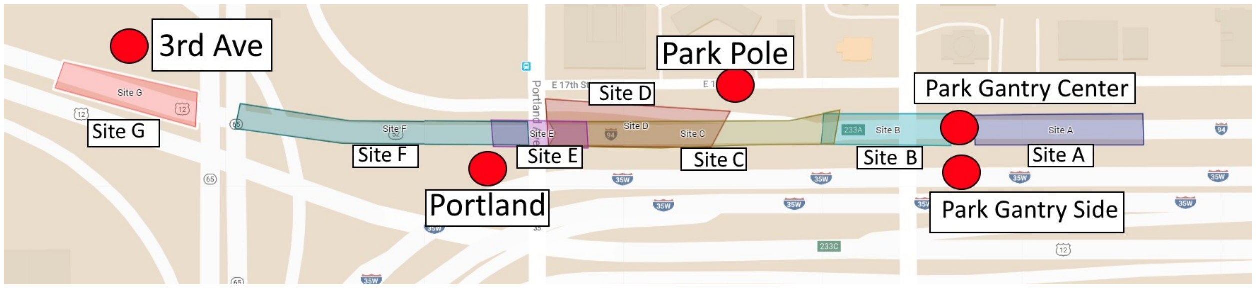


Figure 3.14 Radar (red dots) coverage areas (shaded), with overlaps.

In Figure 3.13 and Figure 3.14, the vehicles are traveling from east to west and encountering the radar coverage in alphabetical order. Site A is data from the region between Chicago Ave and the gantry at Park Ave. Site B is from the gantry at Park Ave, covering the section underneath the Park Ave overpass. That is why the trajectories in Site B of Figure 3.13 change color – clearly, it is the same vehicle with the same trajectory. Site B overlaps with Site C on the west side of the Park Ave overpass. For a limited time, targets are duplicated while the vehicles are picked up by two different sensors. Site C extends all the way to the Portland Ave overpass and overlaps with Site E. Site D is completely inside Site C, and a small part of Site E. There is a small blind spot at the west edge of the Portland Ave overpass, as Site E and Site F sensors are looking in opposite directions with no coverage directly between them. Site F is westward facing from Portland Ave, and Site G is eastward facing from 3rd Ave. The break between F and G is due to the TH65 bridge, where the sensors were placed with as much line of site as possible but still couldn't pick up the space directly under the bridge.

In addition to overlap caused by radar placement, as mentioned in 3.4.2 the lane configuration file sometimes misinterpreted data, leading to duplicate and misaligned vehicles. Figure 3.15 shows one such misalignment circled in red; vehicles are shown to have crossed paths, but no other signs of an incident appear. If the trajectories crossed like on the roadway, a crash would result, and the trajectories would not continue in such an even way.

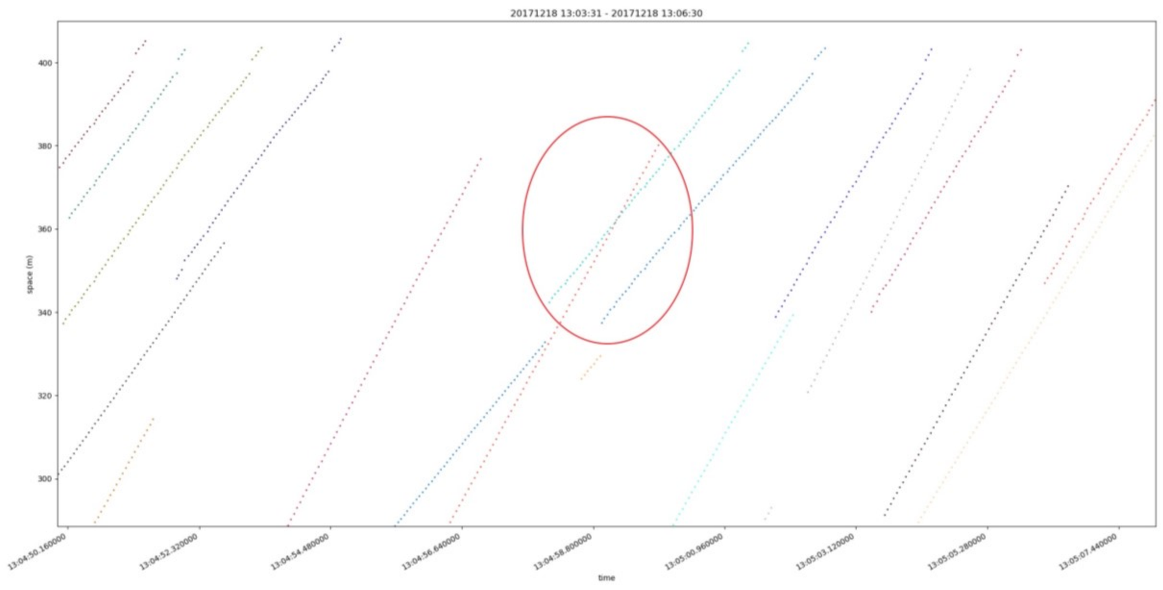


Figure 3.15 Misaligned vehicle due to configuration file.

In addition to the misaligned vehicle, several duplicate trajectories can be pointed out. These result from radar overlap, and the lane configuration algorithm inappropriately placing them on the roadway. Figure 3.16 shows the same screenshot, but with duplicate trajectories circled.

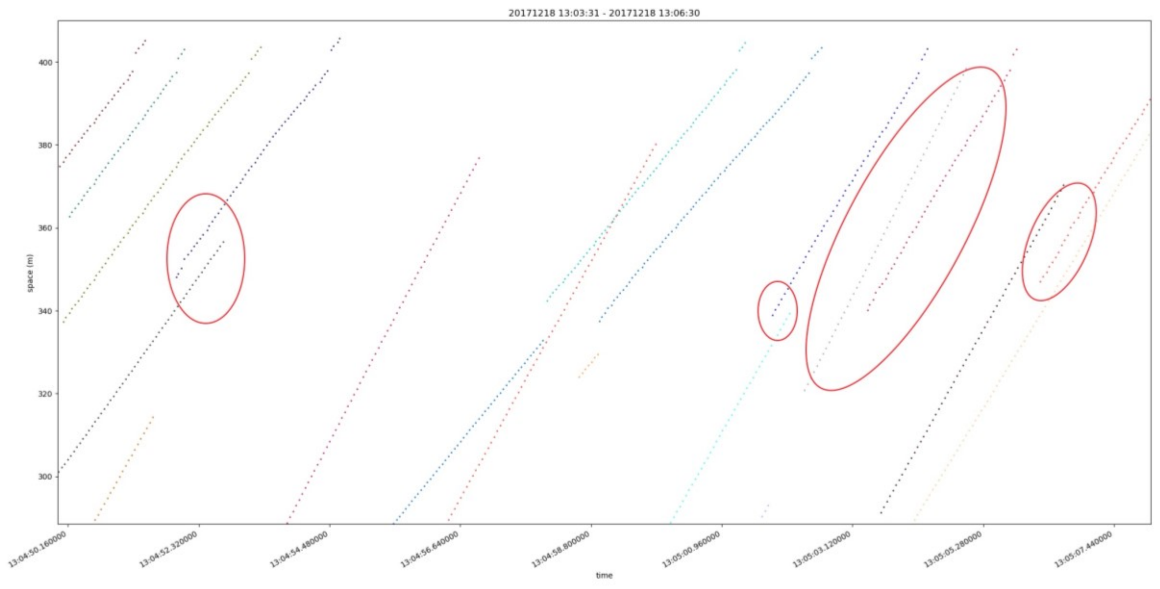


Figure 3.16 Duplicate trajectories.

Within the space-time diagram, users can also select specific trajectories that are of interest and export the raw data for only those trajectories into an Excel file for further analysis. Figure 3.17 shows a zoomed in look at a single trajectory that has been selected for analysis.

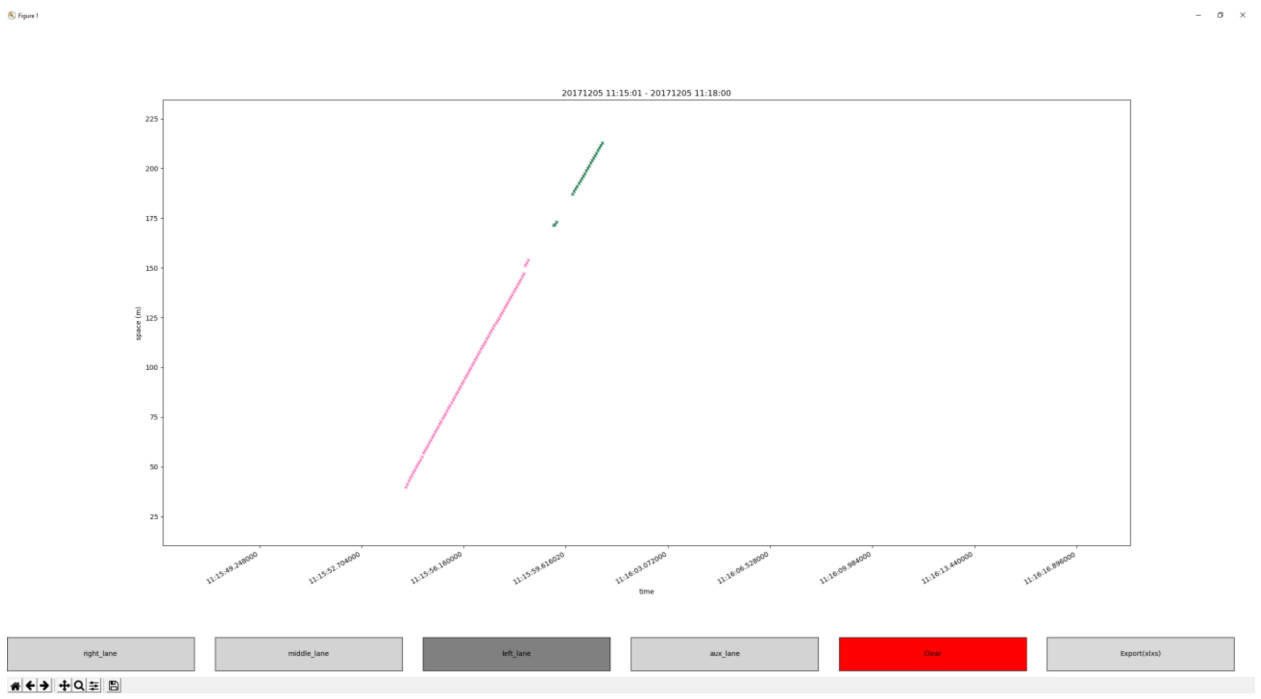


Figure 3.17 Example selected trajectory. (icons have changed from dots to crosses)

Table 3.2 shows an example of the output, where "object_ID" is the vehicle ID. This table contains position, velocity and trajectory information.

Table 3.2 Output of selected trajectory, showing time, vehicle ID, sensor ID, position and velocity.

	timestam	count	object_id	sensor_id	x	y	vx	vy	v (magnitu	length	global x	global y	latitude	longitude
0	1.51E+09	3.21E+08	32	7	104.45	-6.72	-26.9	0	26.9	3.2	606.9528	-58.7051	44.96601	-93.2634
1	1.51E+09	3.21E+08	32	7	102.91	-6.78	-26.9	0	26.9	4.4	605.4128	-58.7651	44.966	-93.2634
2	1.51E+09	3.21E+08	32	7	101.12	-6.53	-26.9	0	26.9	5	603.6228	-58.5151	44.96601	-93.2634
3	1.51E+09	3.21E+08	32	7	99.84	-6.21	-26.9	0	26.9	5	602.3428	-58.1951	44.96601	-93.2634
4	1.51E+09	3.21E+08	32	7	98.5	-5.95	-26.9	0	26.9	5	601.0028	-57.9351	44.96601	-93.2634
5	1.51E+09	3.21E+08	32	7	97.22	-5.76	-27	0	27	5	599.7228	-57.7451	44.96601	-93.2634

One more functionality the Graphical Interface has is animated database visualization. This builds on the database viewer shown in Figure 3.5. From the main page of the software, the user will click "run viewer" and input a date and time in YYYYMMDD HH:MM:SS format. On clicking play, an animation of the vehicle trajectories will be displayed over a still aerial image of the Testbed. Figure 3.18 shows an example day.

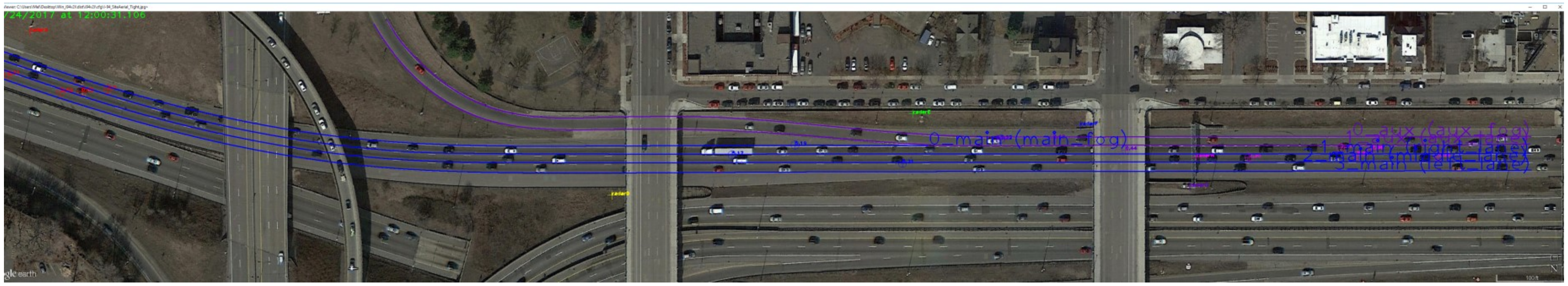


Figure 3.18 An animated vehicle trajectory overlay on an aerial image of the Testbed.

The testbed is quite long, so Figure 3.18 is much more compressed than the viewer looks on a screen. The viewer has lane delineations and represents different sensor data with different colors as vehicles travel the corridor.

The full user manual for this software can be found in Appendix C
Graphical Interface Tool User Manual

3.4.2.4 Future Software Developments

As mentioned in 3.4 future Testbed applications such as BSM emulation will require seamless vehicle trajectories along the length of the Testbed. Algorithms will need to be developed to knit together trajectories based on raw radar data in the database.

Further, the basic configuration file currently used to assign raw data to a lane will need to be upgraded to reduce the number of overlapping, duplicate, and “ghost” vehicles (trajectories that suddenly disappear from a sensor and do not seem to reappear on a sensor down the corridor).

CHAPTER 4: OTHER DEVELOPMENTS

One part of the original proposal, the CV Testbed Dataset, has not been addressed in this report. In addition, during the maintenance period, a spontaneous opportunity to send a vehicle through the Testbed was provided by the Mechanical Engineering Department at the University of Minnesota. Finally, not falling under any specific maintenance or upgrade, a testing system for the MTO's video recording system (streamrecorder) was developed.

4.1 CV TESTBED DATASET

In the original maintenance and upgrade proposal, the MTO was to prepare a portion of the existing CV Testbed data to conform with the USDOT RDE format. In view of the work that is happening because of a new project involving Basic Safety Messages, this dataset task was moved to that project instead.

4.2 MECHANICAL ENGINEERING ON-BOARD UNIT TEST

While the Testbed was ongoing, the Mechanical Engineering department at the U of M was working on a separate CV project. They were developing a drive train that could be controlled through CV technology, given V2V and V2I information. A CV was simulated as being sent through a connected corridor using the software VISSIM; during the simulation, the real-world drive train would respond as if receiving V2V and V2I information. The experimenter outfitted his own vehicle with an on-board device (OBD), but ran into the issue of having no real-world CV environment to test functionality and verify the OBD was functioning correctly, and that GPS data used by the OBD was accurate.

The CV Testbed team was concurrently looking to calibrate the radars along the Testbed. It was determined the test vehicle could aid in doing that by using a known target vehicle in conjunction with sensor and camera data overlaid on a 3D high-resolution model of the Testbed.

In conjunction with the CV Testbed team, a course through the Testbed was determined such that the vehicle would be able to be identified on camera and in the raw dataset. The experimenter, with the OBD activated and identifying marks on his vehicle for the cameras, drove the course many times. Figure 4.1, Figure 4.2, and Figure 4.3 all show the vehicle on camera, identified by the white stripes on the top and rear of the vehicle.

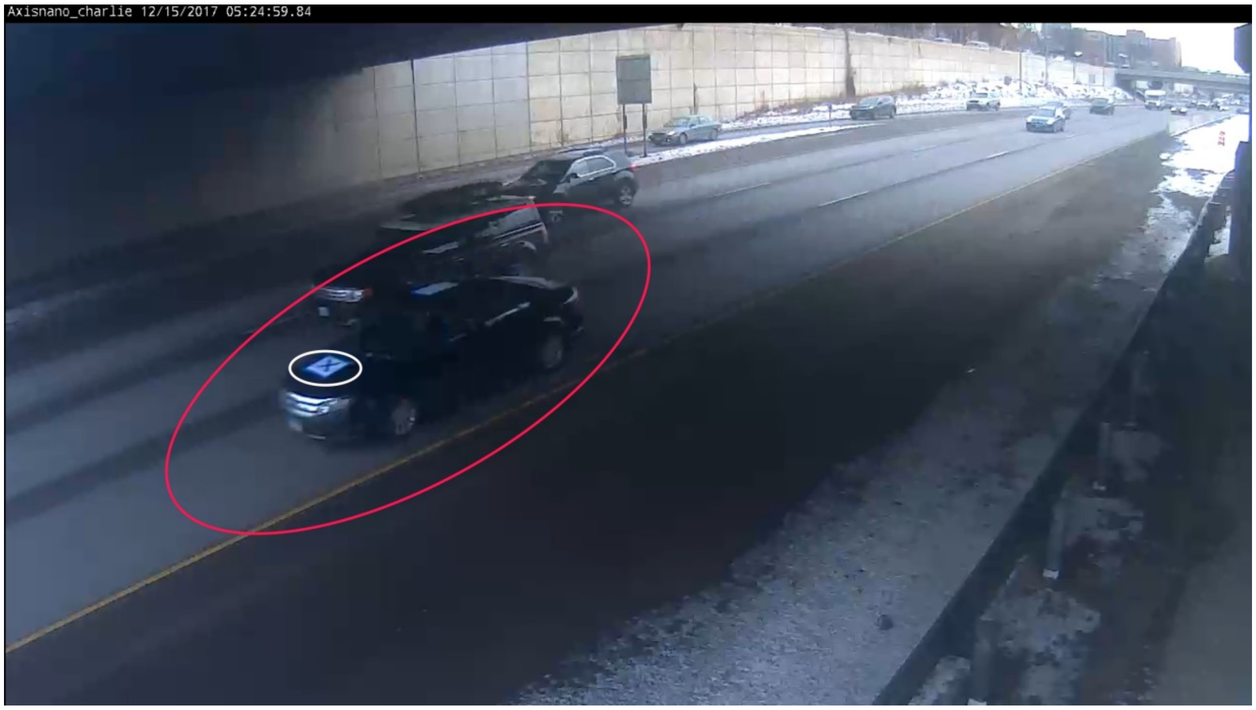


Figure 4.1 Test vehicle. Note the markings on the front, top, and rear of vehicle for identification.

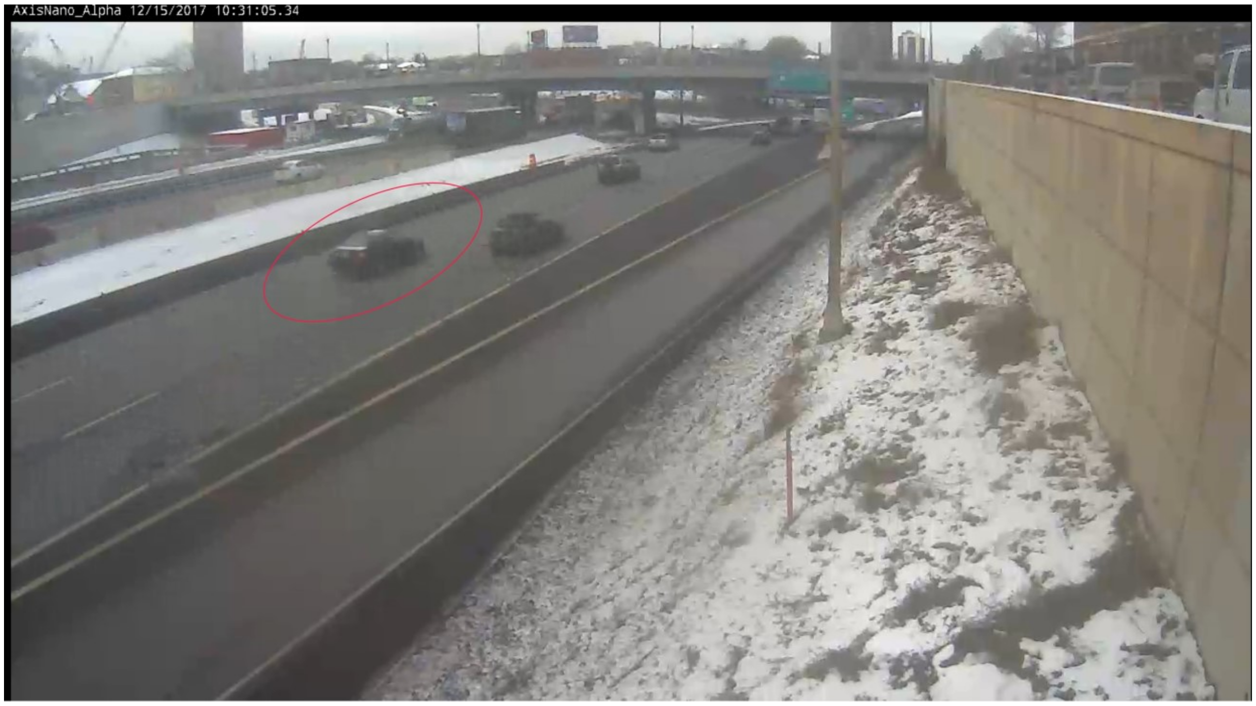


Figure 4.2 Test vehicle from a different angle.

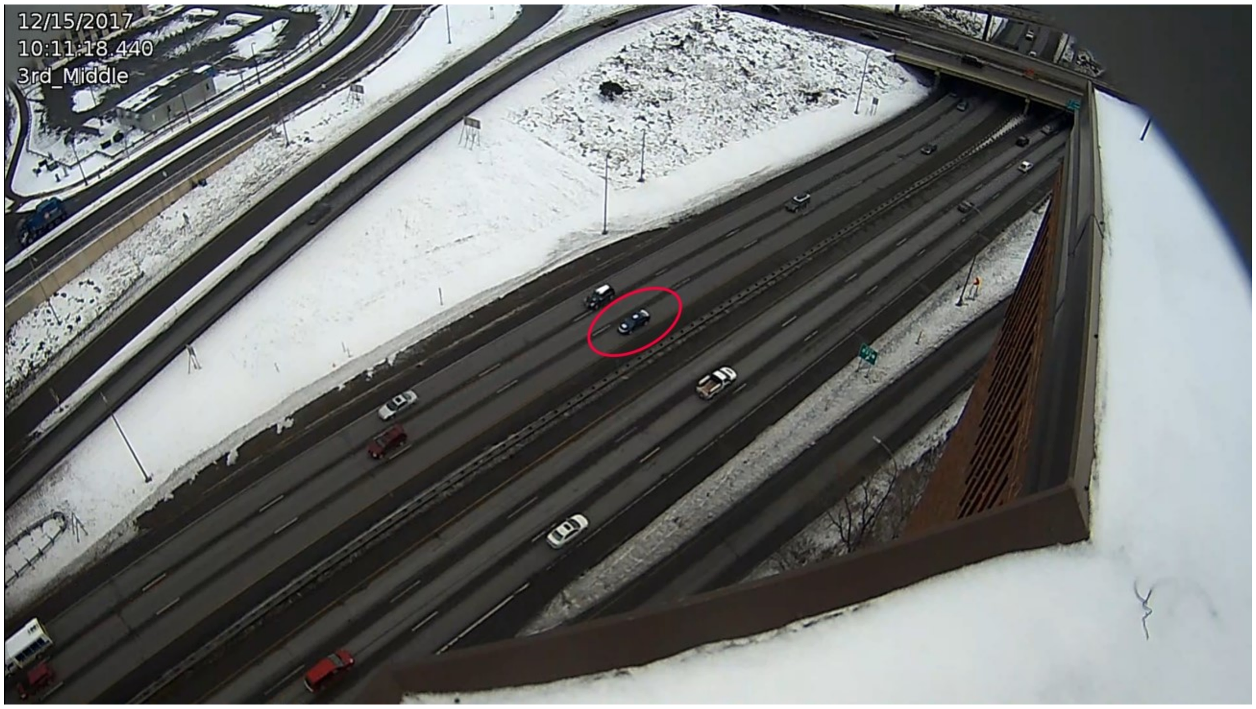


Figure 4.3 Test vehicle from rooftop station.

After video footage and data from all sensors were captured of the vehicle, the CV Testbed team went through and identified the vehicle ID in all passes. This vehicle ID was entered into the Trajectory Plotting tool (see 3.4.2.2) to identify the vehicle-specific radar location data and compare it to the GPS data gathered by the OBD. Verification of both the radar and OBD data is planned by projecting the datasets onto the 3D high resolution map of the Testbed to verify lane accuracy.

4.3 STREAMRECORDER

Streamrecorder is a proprietary MTO software developed in 2002, before the initial deployment of the I-94 Field Lab. It is a way to gather video data on schedule and transport it back to the MTO. The original streamrecorder code was developed to ensure data is being recorded properly (through checking on-board storage for corruption and reporting statistics on how much data has been recorded) and to promote system stability through nightly resets. Over the years, upgrades and maintenance to the code have occurred; in this project, a testing system to ensure streamrecorder is recording accurately and reliably was developed.

In the Testbed configuration, eight IP cameras distributed across the three rooftops provide constant HD video streams of the corridor, captured by a computer on each rooftop. Part of the upgrades to the CV Testbed included the development of Python code used to test the streamrecorder used to capture this video. The testing system applies a set of unit tests that are run whenever a change is made to the code on GitHub. It also applies at set of long-term tests that can be used to identify issues that appear when the system runs continuously for long periods of time, as typically occurs in production. The testing system uses a set of unit tests and configuration tests to make sure that the code operates as intended. The unit tests provide a quick feedback to make sure developers didn't make any easily catchable mistakes; whereas, the configuration tests take longer and are able to find problems that take longer to materialize.

The automated testing system is in place to trigger the tests after any change is detected in the streamrecorder repository. The testing system runs the two sets of the tests described above whenever changes are pushed to any branch on GitHub. The first set of tests, consisting of unit tests and testing using short video files, takes less time and quickly makes sure there are no major flaws. As it is currently configured, this tests two different streams recording files in three-minute intervals, one on a schedule and one constantly. The entire test takes approximately five minutes.

The other set of tests runs for a longer period of time to discover bugs that can only be found after streamrecorder has been running for a while. This test records one-hour videos over a period of 24 hours to match a more typical production environment.

Together, these tools provide a framework for streamlining development of the streamrecorder project without sacrificing reliability. Automated testing allows feature requests or bug fixes to be implemented quickly and immediately verified to ensure the changes don't break existing functionality. This system has already allowed for several known bugs to be fixed, along with several improvements to the architecture of the code and its usability. Though it is usable in its current form, improvements to the system can be made in the future such as: automated notification after tests are run, granular control of which tests are applied when changes occur on specific branches in the repository, and automatic merging of branches on successful completion of tests.

CHAPTER 5: CONCLUSION

By maintaining and upgrading both the hardware and software of existing Testbed stations as well as moving stations due to construction while maintaining the capacity of the original Testbed, the MTO has created a fully functional CV testbed uniquely situated to attract freeway safety-oriented V2I and V2V safety application development, implementation, and evaluation projects.

Future projects planned with the Testbed include:

- I-94 Queue Warning system refactoring — a reimplementation of the MN Q-WARN system with improvements
    - MN-QWARN has been in operation since June 2016 and originally utilized machine vision sensors, but since radar data became available, it has been modified to also use that data.
- Basic Safety Message Emulation
    - A project spearheaded by Dr. Levin at the University of Minnesota Twin Cities is using data gathered by the Testbed to emulate Basic Safety Messages.

In addition, the CV Testbed could be used for:

- Seamless vehicle trajectories and travel time information
- More OBD testing
- Safety, security and environmental application testing

Overall, the CV Testbed represents a stage beyond simulation but not yet the full implementation of CVs on the roadway. Given the unique nature of the system deployed in a real-world setting, any CV applications requiring interaction with a large number of unconnected vehicles could be tested in this facility. It is an important stepping stone for research as CVs start to enter the marketplace.

REFERENCES

Balke, K., Charara, H., & Sunkari, S. (2014). *Report on Dynamic Speed Harmonization and Queue Warning Algorithm Design.* Washington, DC: United States Dept. of Transportation.

Davis, G., & Hourdos, J. (2012). *Development of Analysis Methods Using Recent Data.* Washington, DC: Transportation Research Board.

Dirks, P., Liu, Z., & Hourdos, J. (2016). Freeway Traffic Queue-Warning System: Deployment and First Impressions. Presented at the Transportation Research Board 96th Annual Meeting. Washington DC: Transportation Research Board.

Federal Communications Commission. (2013). FCC 13-98: Report and Order and Further Notice of Proposed Rulemaking. Retrieved from https://apps.fcc.gov/edocs_public/attachmatch/FCC-13-98A1_Rcd.pdf

Federal Communications Commission. (2015). *Operation of Radar Services in the 76-81 GHz Band.* Washington, DC: Federal Communications Commission.

Greenshields, B. D. (1961). Quality of Traffic Flow, Quality and Theory of Traffic Flow. Presented at the Symposium, Bureau of Highway Traffic, Yale University, New Haven, CT.

Helly, W., & Baker, P.G. (1965). Acceleration noise in a congested signalized environment. Vehicular Traffic Science. (pp. 55–61) *Proceedings of the Third International Symposium on the Theory of Traffic Flow*. New York: American Elsevier.

Hill, C., & Krueger, G. (2015). *ITS ePrimer Module 13: Connected Vehicles*. Retrieved from https://www.pcb.its.dot.gov/eprimer/module13.aspx

Hourdakis, J., Michalopuloulos, P., & Morris, T. (2004). Deployment of Wireless Mobile Detection and Surveillance for Data-Intensive Applications. *Transportation Research Record*, 1900, 140–148.

Hourdakis, J., Morris, T., Michalopoulos, P., & Wood, K. (2005). *Advanced Portable Wireless Measurement and Observation Station.* Minneapolis, MN: Intelligent Transportation Systems Institute, Center for Transportation Studies, University of Minnesota.

Hourdos, J. (2005) Crash prone traffic flow dynamics: Identification and real-time detection. Ph.D. dissertation, University of Minnesota, Minneapolis.

Hourdos, J., & Zitzow, S. (2014). *Investigation of the Impact of the I-94 ATM System on the Safety of the I-94 Commons High Crash Area.* St. Paul: Minnesota Department of Transportation Research Services & Library.

Hourdos, J., Garg, V., & Michalopoulos, P. (2008). *Accident Prevention Based on Automatic Detection of Accident Prone Traffic Conditions: Phase I.* Minneapolis: Intelligent Transportation Systems Institute, University of Minnesota.

Hourdos, J., Liu, Z., Dirks, P., Liu, H. X., Huang, S., Sun, W., & Xiao, L. (2017). *Development of a Queue Warning System Utilizing ATM Infrastructure System Development and Field-Testing.* St. Paul: Minnesota Local Road Research Board, Minnesota Department of Transportation Research Services & Library.

Hourdos, J., Xin, W., & Michalopoulos, P. (2008). *Development of Real-Time Traffic Adaptive Crash Reduction Measures for the Westbound I-94/35W Commons Section.* Minneapolis, MN: Intelligent Transportation Systems Institute, University of Minnesota.

Jones, Trevor R., & Potts Renfrey, B. (1962). The Measurement of Acceleration Noise-A Traffic Parameter. *Operations Research, 10*(6), 745–763.

Parikh, G., & Houdos, J. (2014). *Implementation of High Accuracy Radar Detectors for Traffic Safety Countermeasure Evaluation.* Minneapolis: Center for Transportation Studies, University of Minnesota.

Phillips, W. F. (1979). A kinetic model for traffic flow with continuum implications. *Transportation Planning and Technology, 5*, 131–138.

Rindels, M., Zitzow, S., & Hourdos, J. (2016). *Evaluation of the Effectiveness of ATM Messages Used During Incidents.* St. Paul: Minnesota Department of Transportation Research Services & Library.

Smart Microwave Sensors GmbH. (2017, November 9). *UMRR Traffic Sensor Type 29 Data Sheet*. Retrieved from http://www.smartmicro.de/fileadmin/user_upload/Documents/TrafficRadar/UMRR_Traffic_Sensor_Type_29_Data_Sheet.pdf

Smart Microwave Sensors GmbH. (2017, November 9). *UMRR Traffic Sensor Type 30 Data Sheet*. Retrieved from http://www.smartmicro.de/fileadmin/user_upload/Documents/TrafficRadar/UMRR_Traffic_Sensor_Type_30_Data_Sheet.pdf

Xin, W., Hourdos, J., & MIchalopoulos, P. (2008). *Enhanced Micro-Simulation Models for Accurate Safety Assessment of Traffic Management ITS Solutions.* Minneapolis, MN: Intelligent Transportation Systems Institute, University of Minnesota.

APPENDIX A

TRAJECTORY PLOTTING USER MANUAL

This software is designed to identify and display which sensors were available on a selected date and time range during the operation of the I-94 CV Testbed.

A.1 INTRODUCTION

A high-resolution trajectory for each vehicle is calculated by radar stations along the I-94 CV Testbed. Six of the seven detectors are on a relatively straight piece of road, while the last one is on a curve. Existing configuration parameters needed to be revisited throughout the project as sensors were moved over the course of construction. If a sensor went down for any period of time, other sensors were still able to collect data with reduced accuracy. Each trajectory is the result of multiple time series. A time series is defined as a series of values of a quantity obtained at successive times, often with equal intervals between them.

The radars actively collect data at 20 hertz and cover a distance of roughly 300ft. The values collected are X and Y distance coordinates in a local system relative to the radar. The speed of the vehicle is calculated based on the shift in sinusoidal frequency between the observed and the emitted frequency of a wave for a vehicle relative to the radar (Doppler Effect). It is represented as an X and Y coordinate set of the vehicles' time mean speed. If the location data is poor, the speed of the vehicle can still be used to create a trajectory.

Each vehicle that the radar senses is assigned with a unique ID ranging from 0 to 63. (This is due to the use of Type 29 and Type 30 radar along the Testbed; only 64 unique objects can be tracked by those types of radar at any given time.) This can cause problems because sensors will run out of IDs to assign vehicles, meaning different vehicles could have the same ID.

A script has been written in Matlab to retrieve the data from the database and generate a figure showing the availability of the radars throughout the day. The data is stored in the PostgreSQL database at the Minnesota Traffic Observatory under "*testsensoravailability*".

A.1.1 Using the Interface

To make the code user-friendly, a User Interface for the Matlab code SensorAvailability.m has been developed and should be activated to begin the plotting process. Figure A.1 shows the layout of the User Interface upon activation.

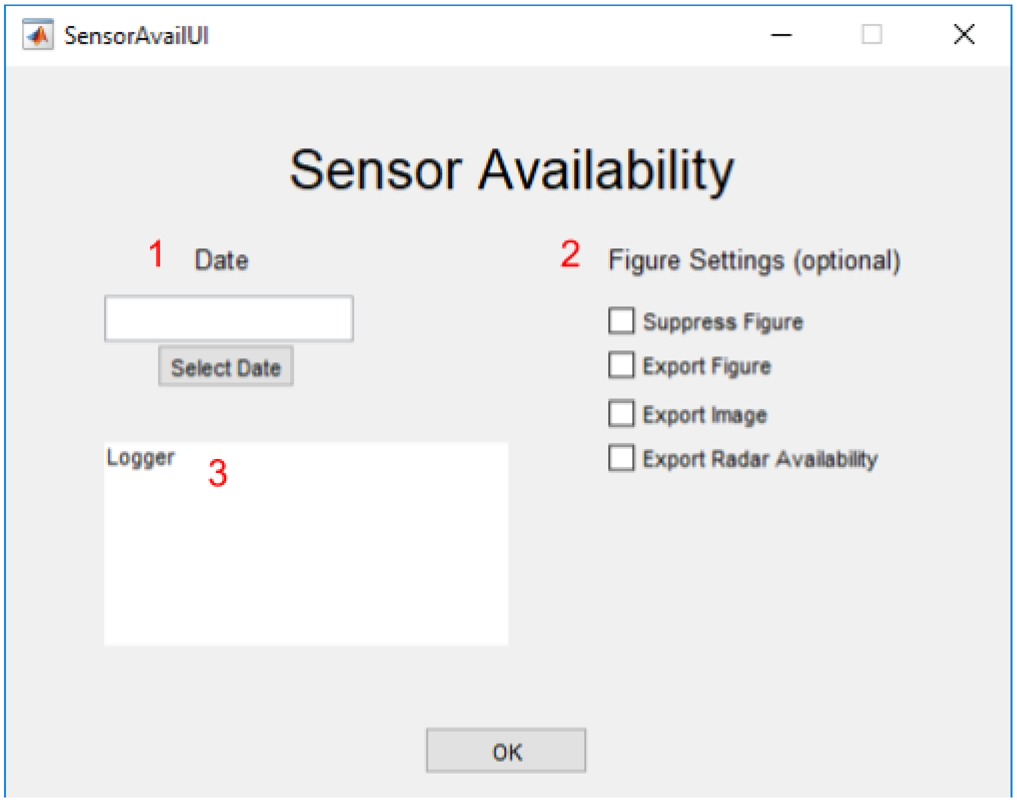


Figure A.1 The Matlab Sensor Availability user interface

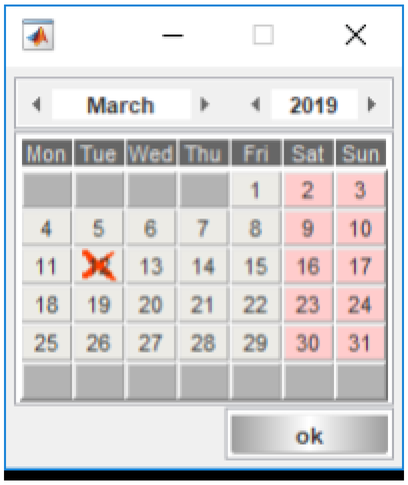
SensorAvailability.m requires the following arguments:

1. Date: Input a date in this format, 'yyyymmdd'. {e.g. '20171218'}
2. Figure Settings
    a. Suppress Figure: On ticking the checkbox, the figure will not pop up.
    b. Export Figure: On ticking the checkbox, the figure generated will be exported and saved as a figure file.
    c. Export Image: By ticking the checkbox, the figure generated will be exported and saved as a .jpg file.
    d. Export Radar Availability: By ticking the checkbox, the query request from the database will be saved as an Excel.

To generate a figure showing the availability of sensor for a day, the user will only need to enter the date. In Figure A.1, the number one highlighted in red shows where user should input the date. By clicking the "Select Date" button, a calendar User Interface will pop out and user can select a date from it as shown in Figure A.2.

After that, the number two highlighted in red shows the optional settings that user can select from, as described above under Figure Settings.

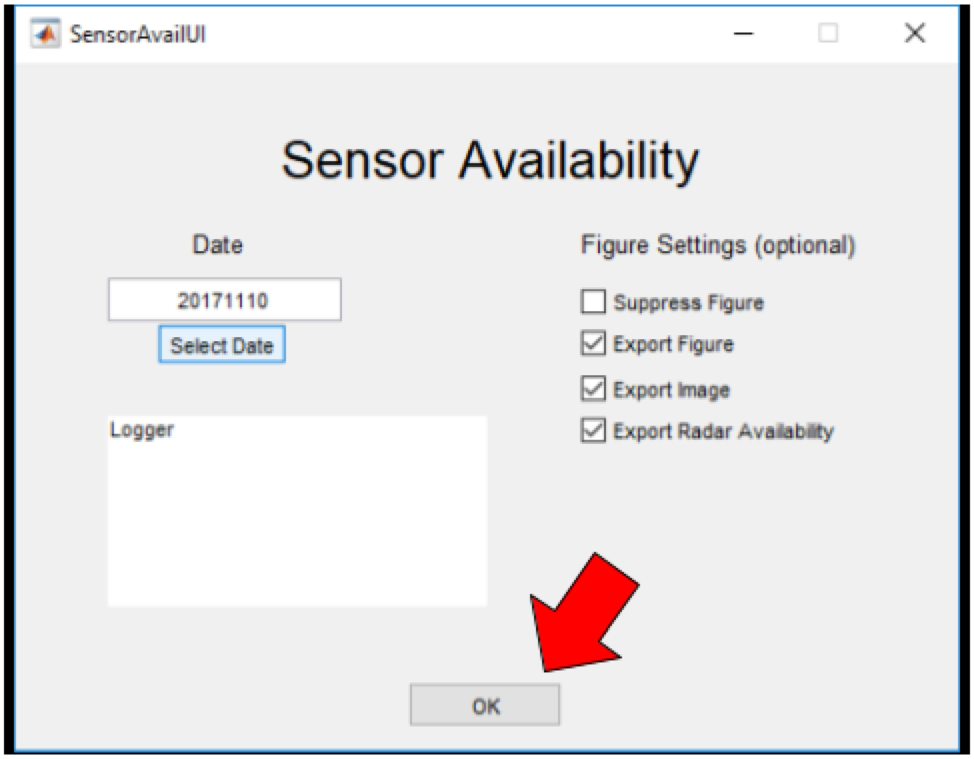
The number three highlighted in red shows the logger for the script. Any error encountered will be logged into the logger for review.



Mon	Tue	Wed	Thu	Fri	Sat	Sun
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Figure A.2 Calendar User Interface

After the arguments are filled, press the OK button as shown in Figure A.3 to generate the figure.


SensorAvailUI

Sensor Availability

Date: 20171110

Select Date

Logger

Figure Settings (optional)

- [ ] Suppress Figure
- [x] Export Figure
- [x] Export Image
- [x] Export Radar Availability

Figure A.3 The UI, ready to generate the figure.

Figure A.4 shows the output from SensorAvailability.m. The y-axis represents the individual station indexes while the x-axis represents the time throughout the day in thirty minute intervals. If the sensor at a particular station is operating at a time, it will be color coded in green. Otherwise, it will be color coded in red. Note the small red lines on sensor 1 from 2:00-4:00, and the small line of green at about

16:00 for sensors 2, 3, and 4--while the output shows thirty minute increments, the sensor data is captured every minute and is represented in full resolution here.

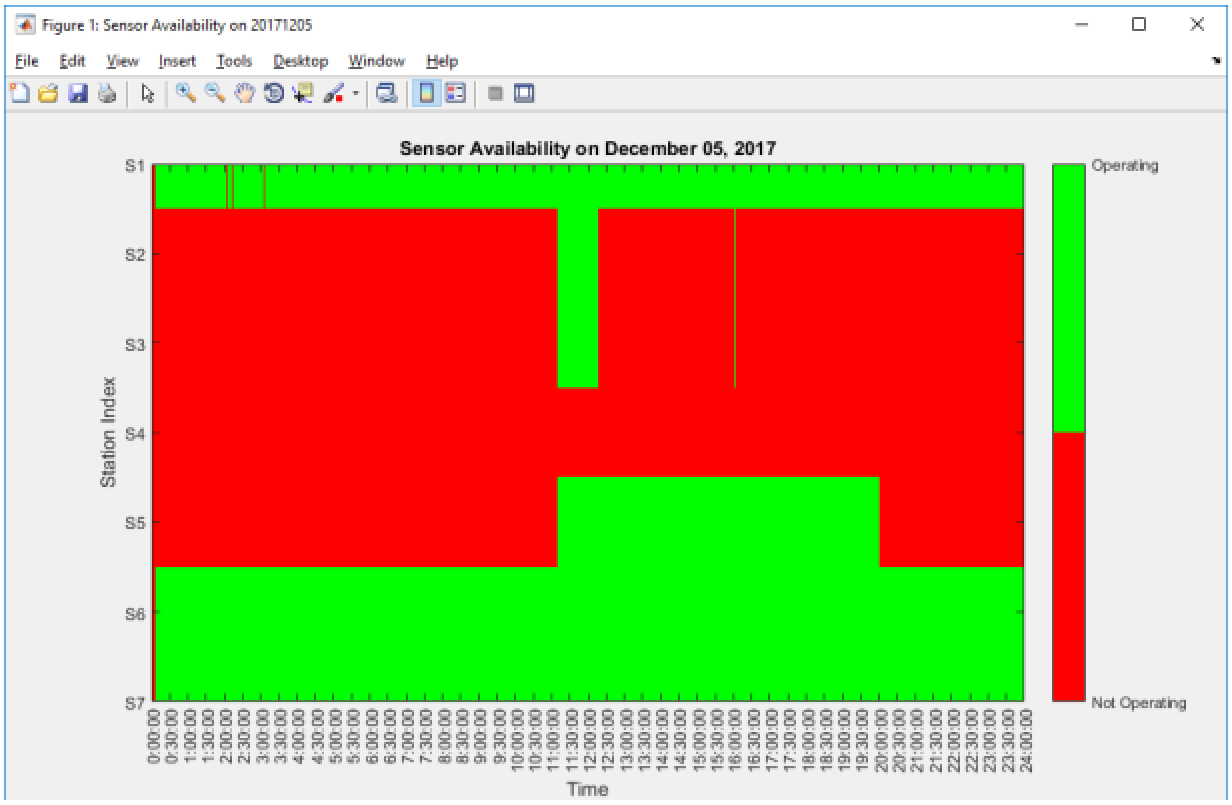


Figure A.4 The output showing sensor availability for the selected day in 30 minute intervals

Users can then zoom in and out of the graph to get the sensor availability data required.

APPENDIX B

TRAJECTORY PLOTTING USER MANUAL

This software is designed to identify and display which sensors were available on a selected date and time range during the operation of the I-94 CV Testbed.

B.1 INTRODUCTION

A high-resolution trajectory for each vehicle is calculated by radar stations along the I-94 CV Testbed. Six of the seven detectors are on a relatively straight piece of road, while the last one is on a curve. Existing configuration parameters needed to be revisited throughout the project as sensors were moved over the course of construction. If a sensor went down for any period of time, other sensors were still able to collect data with reduced accuracy. Each trajectory is the result of multiple time series. A time series is defined as a series of values of a quantity obtained at successive times, often with equal intervals between them.

The radars actively collect data at 20 hertz and cover a distance of roughly 300ft. The values collected are X and Y distance coordinates in a local system relative to the radar. The speed of the vehicle is calculated based on the shift in sinusoidal frequency between the observed and the emitted frequency of a wave for a vehicle relative to the radar (Doppler Effect). It is represented as an X and Y coordinate set of the vehicles' time mean speed. If the location data is poor, the speed of the vehicle can still be used to create a trajectory.

Each vehicle that the radar senses is assigned with a unique ID ranging from 0 to 63. (This is due to the use of Type 29 and Type 30 radar along the Testbed; only 64 unique objects can be tracked by those types of radar at any given time.) This can cause problems because sensors will run out of IDs to assign vehicles, meaning different vehicles could have the same ID.

A script has been written in Matlab to retrieve the data from the database and generate a figure showing the availability of the radars throughout the day. The data is stored in the PostgreSQL database at the Minnesota Traffic Observatory under "*testi94*".

B.1.1 Using the Interface

To make the code user-friendly, a User Interface for the Matlab code PlotSensorData.m has been developed and should be activated to begin the plotting process. Figure B.1 shows the layout of the User Interface upon activation.

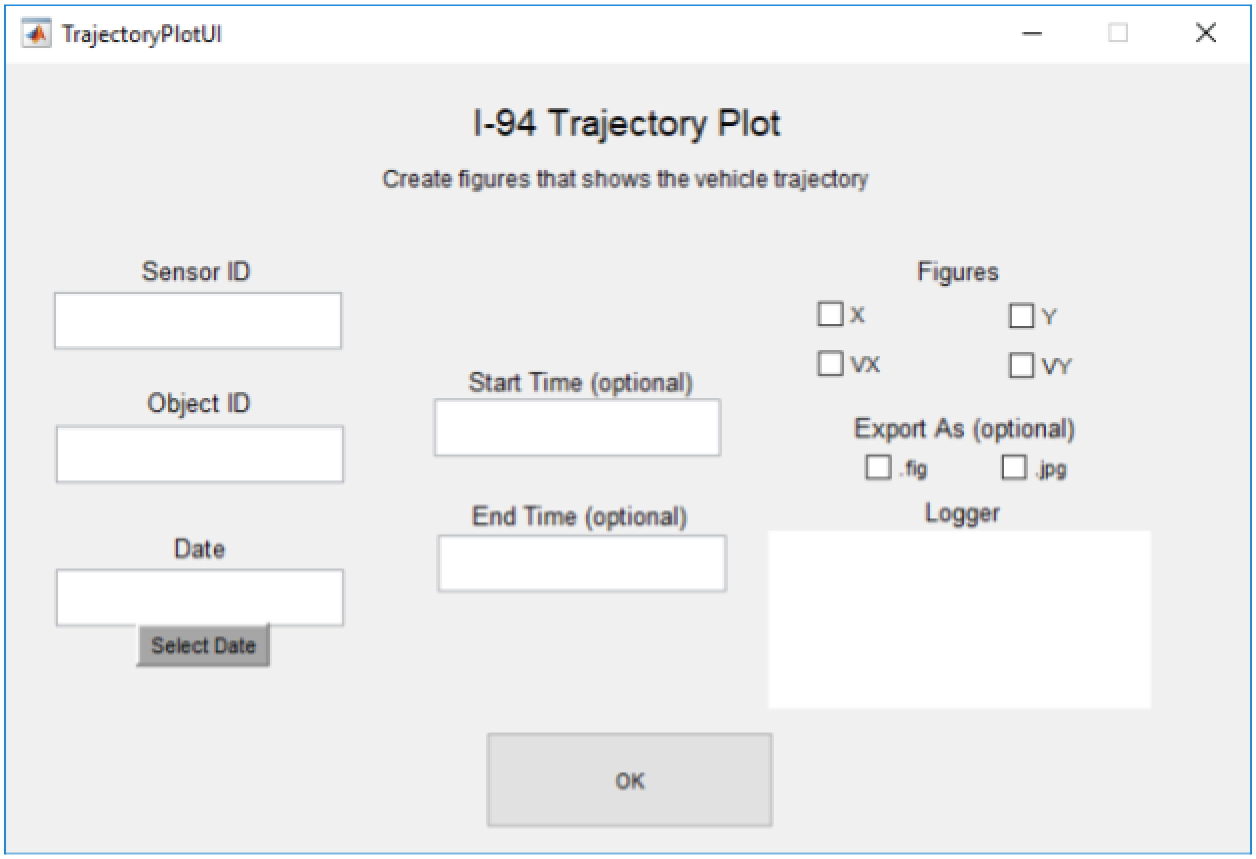
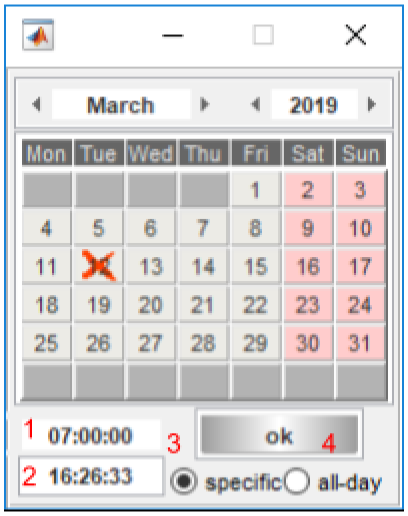


Figure B.1 The Trajectory Plotting User Interface

PlotSensorData.m requires the following arguments:

1. Date: Input a date in this format, ‘yyyymmdd’. {e.g. ‘20171218’}
2. Start_time (optional): Input a time in this format ‘hh:mm:ss’ {e.g. ‘10:00:00} (optional)
3. End_time (optional): Input a time in this format ‘hh:mm:ss’ {e.g. ‘16:00:00} (optional)
4. Sensor_id: The id of the sensor in integer type.
5. Object_id: The id of the object in integer type.
6. Plot_request: An array with 4 elements that tells Matlab to plot the figure requested. (0 to skip, 1 to plot.) {e.g. [0 0 1 1] Plot only vx and vy.}
7. Export Figure (optional): By ticking the checkbox, the figure generated will be exported and save as a figure file.
8. Export Image (optional): By ticking the checkbox, the figure generated will be exported and save as a .jpg file.

To generate figures showing the trajectories for a day, the user **must enter the date, object id, and sensor id.** Shown in Figure A.2, a calendar pop-out will appear when the “Select Date” button is clicked to aid the user in inputting a date and time.



Mon	Tue	Wed	Thu	Fri	Sat	Sun
				1	2	3
4	5	6	7	8	9	10
11	✗	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

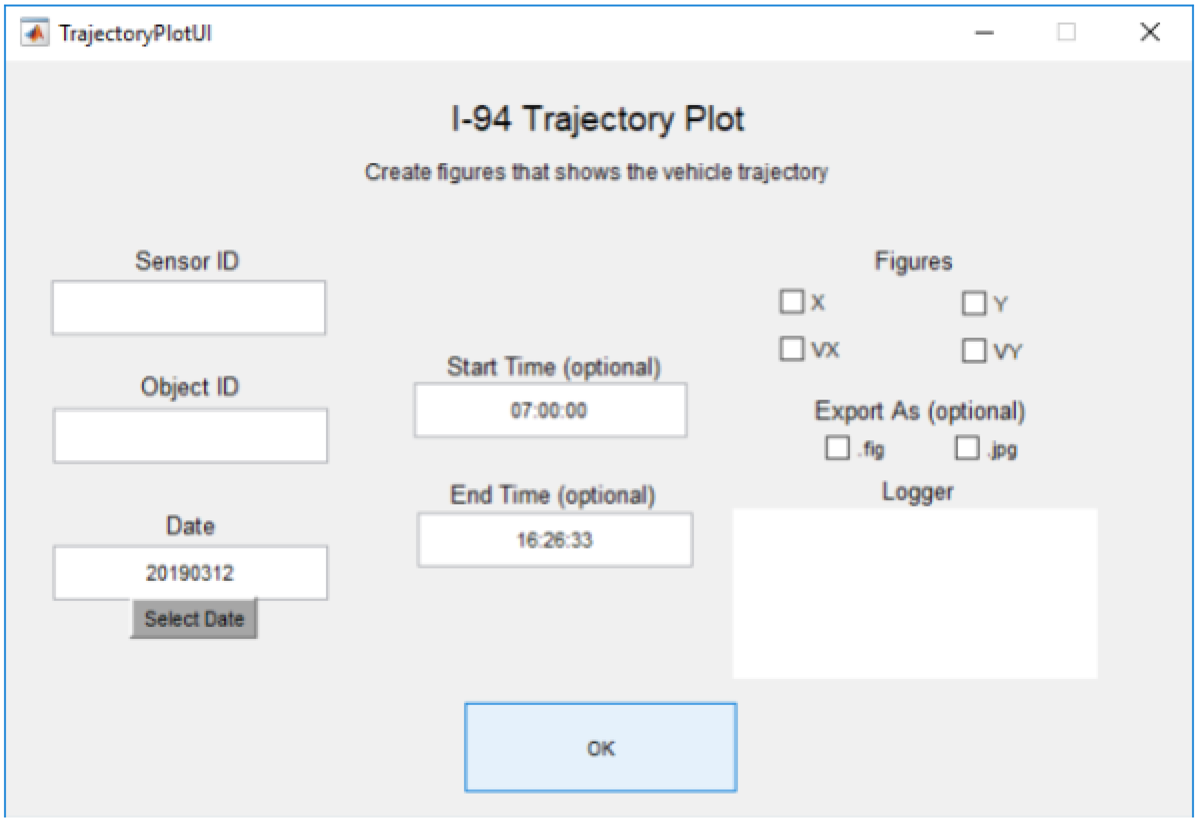
07:00:00
16:26:33
[✓ specific] all-day

Figure B.2 Calendar User Interface

The numbers labeled in Figure B.2 are:

1. Start time
2. End time
3. Specific: when this setting is chosen, 1 and 2 **must be filled**. This setting tells the program to pull a specific time period with a start and end point.
4. All-day: When this setting is chosen, 1 and 2 can be ignored. This setting retrieves the entire day's data.

After a date has been chosen, and all required inputs are filled in, the user can click the ok button to return to the main interface, which will look like Figure B.3.



Date: 20190312

Start Time (optional): 07:00:00

End Time (optional): 16:26:33

Figure B.3 The UI after selecting a date and desired time range

The user **must** fill in the Sensor ID and Object ID of their target trajectory. After that, desired figures must be chosen. The figures can be exported as a figure (.fig) file or a jpeg (.jpeg) file. Once everything is filled in, the figures can be generated clicking the OK button. Figure B.4 shows the output of the code.

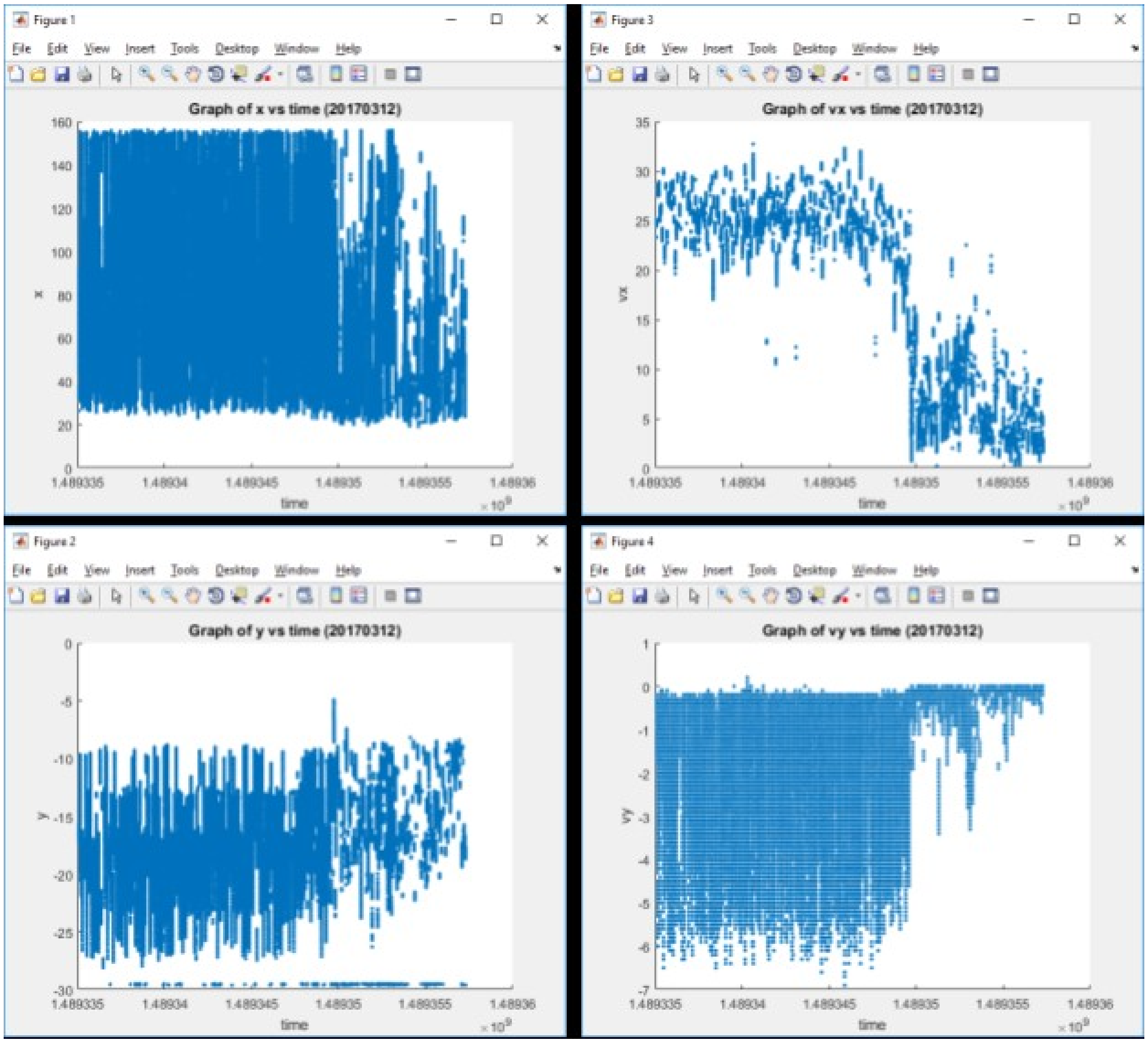


Figure B.4 The vehicle trajectory plot outputs

There will be 4 graphs generated when all four checkboxes are marked in the UI.

1. Graph of x vs time ({DATE}): The figure drawn are the vehicle's X distance coordinates in a local system relative to the radar versus time.
2. Graph of vx vs time ({DATE}): The figure drawn are the vehicle's velocity in the direction of X distance coordinates in a local system relative to the radar versus time.
3. Graph of y vs time ({DATE}): The figure drawn are the vehicle's Y distance coordinates in a local system relative to the radar versus time.
4. Graph of vy vs time ({DATE}): The figure drawn are the vehicle's velocity in the direction of Y distance coordinates in a local system relative to the radar versus time.

NOTE: The time shown is in UNIX timestamp. A tool has been developed to convert the UNIX timestamp into a readable form. Figure B.5 shows the output of running the tool on the given dataset.

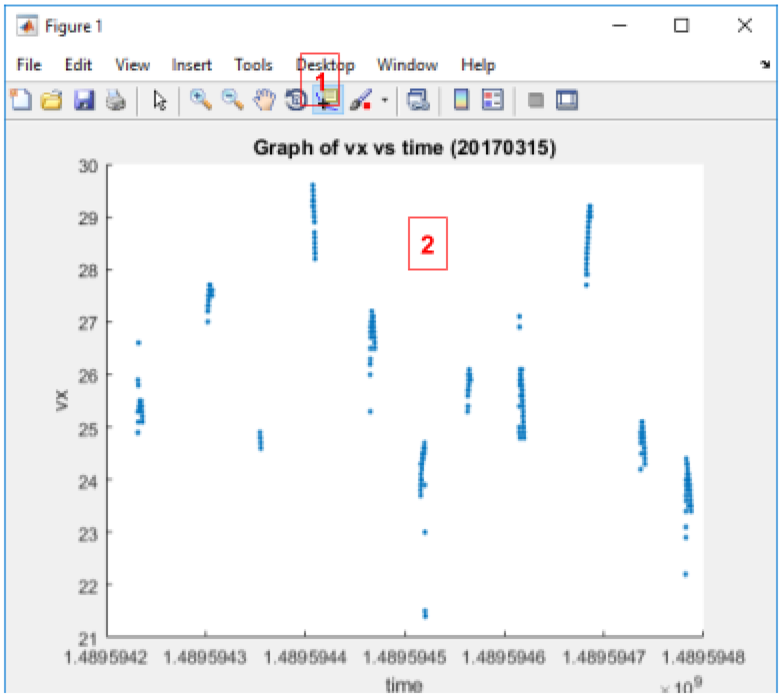


Figure B.5 Data output showing a readable form

To run the tool:

1. Click the Data Cursor button shown in Figure B.5 Data output showing a readable form.
2. Within the white box as shown in Figure 5, right click and then locate “Select Text Update Function...” and left click it.
3. Locate the file name called TipTool_PSD.m.
4. The Data Cursor will show the value of the data and the time stamp corresponding to it.

Users can then save the figure for analysis.

APPENDIX C

GRAPHICAL INTERFACE TOOL USER MANUAL

This software is designed to be a full graphical interface for the PostgreSQL database, including: sensor availability heatmapping, trajectory plotting via space-time maps, and animated overlays of vehicle trajectories.

C.1 INTRODUCTION

A high-resolution trajectory for each vehicle is calculated by radar stations along the I-94 CV Testbed. Six of the seven detectors are on a relatively straight piece of road, while the last one is on a curve. Existing configuration parameters needed to be revisited throughout the project as sensors were moved over the course of construction. If a sensor went down for any period of time, other sensors were still able to collect data with reduced accuracy. Each trajectory is the result of multiple time series. A time series is defined as a series of values of a quantity obtained at successive times, often with equal intervals between them.

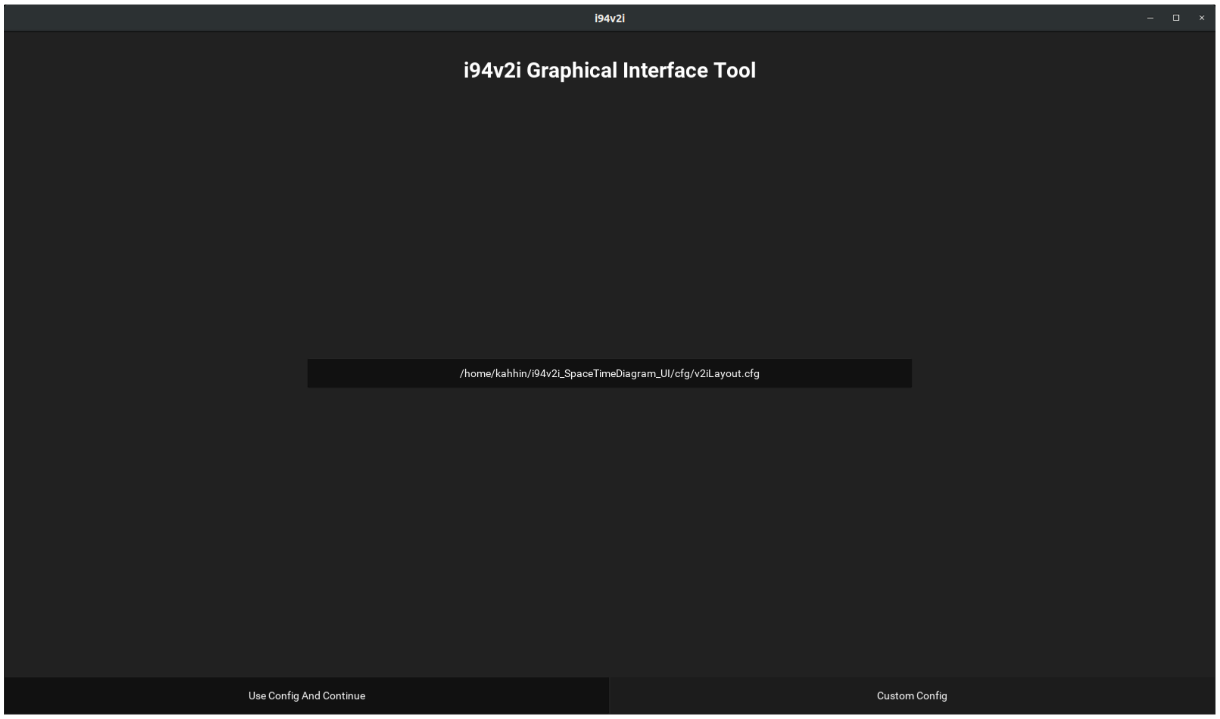
The radars actively collect data at 20 hertz and cover a distance of roughly 300ft. The values collected are X and Y distance coordinates in a local system relative to the radar. The speed of the vehicle is calculated based on the shift in sinusoidal frequency between the observed and the emitted frequency of a wave for a vehicle relative to the radar (Doppler Effect). It is represented as an X and Y coordinate set of the vehicles' time mean speed. If the location data is poor, the speed of the vehicle can still be used to create a trajectory.

Each vehicle that the radar senses is assigned with a unique ID ranging from 0 to 63. (This is due to the use of Type 29 and Type 30 radar along the Testbed; only 64 unique objects can be tracked by those types of radar at any given time.) This can cause problems because sensors will run out of IDs to assign vehicles, meaning different vehicles could have the same ID.

A script has been written in Python to retrieve the data from the database and generate various graphical interfaces with it. These interfaces include a calendar-style view of available sensors per day, a space-time plot of sensor data over a user-selected date and time, capacity to select individual trajectories and export them for analysis, a sensor availability heatmap, and an animation of vehicle trajectories overlaid on an aerial image of the Testbed.

C.1.1 Using the Interface

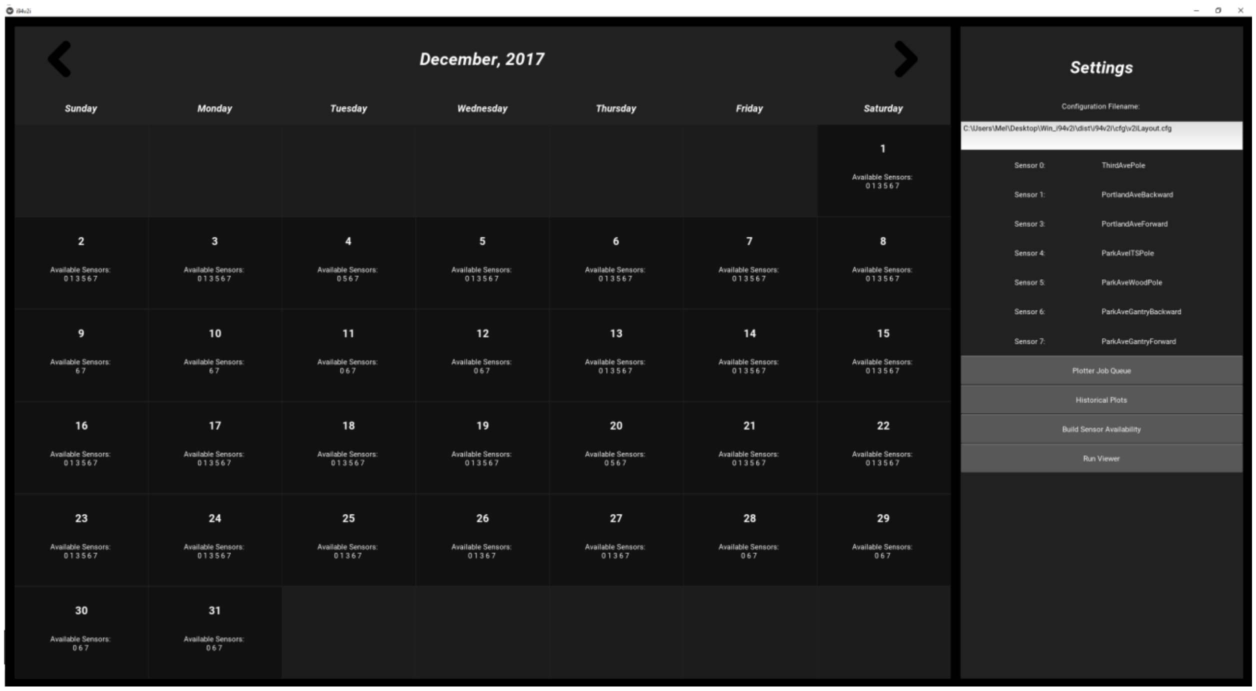
Figure C.1 shows the first page a user will see when opening the Graphical Interface tool. For now, only "Use Config and Continue" is a supported feature but "Custom Config" will be implemented to give users greater control over data selection. "Custom Config" allows for the current lane assignment algorithm to be updated in the future for greater accuracy of data placement, or to adjust for the changing geometry of the Testbed due to construction.



/home/kahhin/i94v2i_SpaceTimeDiagram_UI/cfg/v2iLayout.cfg

Figure C.1 The Graphical Interface Tool User Interface

To continue, users will click “Use Config And Continue”. Figure C.2 shows the main page of the program, showing a calendar-like interface of which sensor data is available for each day. Users can select a day of their choice to see sensor availability data broken down by hour for that day.



Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
						1 Available Sensors: 0 1 3 5 6 7
2 Available Sensors: 0 1 3 5 6 7	3 Available Sensors: 0 1 3 5 6 7	4 Available Sensors: 0 5 6 7	5 Available Sensors: 0 1 3 5 6 7	6 Available Sensors: 0 1 3 5 6 7	7 Available Sensors: 0 1 3 5 6 7	8 Available Sensors: 0 1 3 5 6 7
9 Available Sensors: 6 7	10 Available Sensors: 6 7	11 Available Sensors: 0 6 7	12 Available Sensors: 0 6 7	13 Available Sensors: 0 1 3 5 6 7	14 Available Sensors: 0 1 3 5 6 7	15 Available Sensors: 0 1 3 5 6 7
16 Available Sensors: 0 1 3 5 6 7	17 Available Sensors: 0 1 3 5 6 7	18 Available Sensors: 0 1 3 5 6 7	19 Available Sensors: 0 1 3 5 6 7	20 Available Sensors: 0 5 6 7	21 Available Sensors: 0 1 3 5 6 7	22 Available Sensors: 0 1 3 5 6 7
23 Available Sensors: 0 1 3 5 6 7	24 Available Sensors: 0 1 3 5 6 7	25 Available Sensors: 0 1 3 6 7	26 Available Sensors: 0 1 3 6 7	27 Available Sensors: 0 1 3 6 7	28 Available Sensors: 0 6 7	29 Available Sensors: 0 6 7
30 Available Sensors: 0 6 7	31 Available Sensors: 0 6 7					

Configuration Filename:
C:\Users\Mel\Desktop\Win_i94v2i\dist\i94v2i\cfg\v2iLayout.cfg

Sensor 0:	ThirdAvePole
Sensor 1:	PortlandAveBackward
Sensor 3:	PortlandAveForward
Sensor 4:	ParkAveITSPole
Sensor 5:	ParkAveWoodPole
Sensor 6:	ParkAveGantryBackward
Sensor 7:	ParkAveGantryForward

Figure C.2 The main page of the program, showing a calendar of available sensors

Figure C.3 is an image of sensor availability per hour on the selected date. Available hours are shown on a timeline in green, with unavailable hours shown in red. Figure C.3 shows sensors 0 through 6, but all sensors are accessible by scrolling down in the program window. Sensors can be added to the space-time diagram with the “apply” button, which will change to “cancel” when selected as shown in Figure C.4. Note “Plot Object” is planned future functionality.

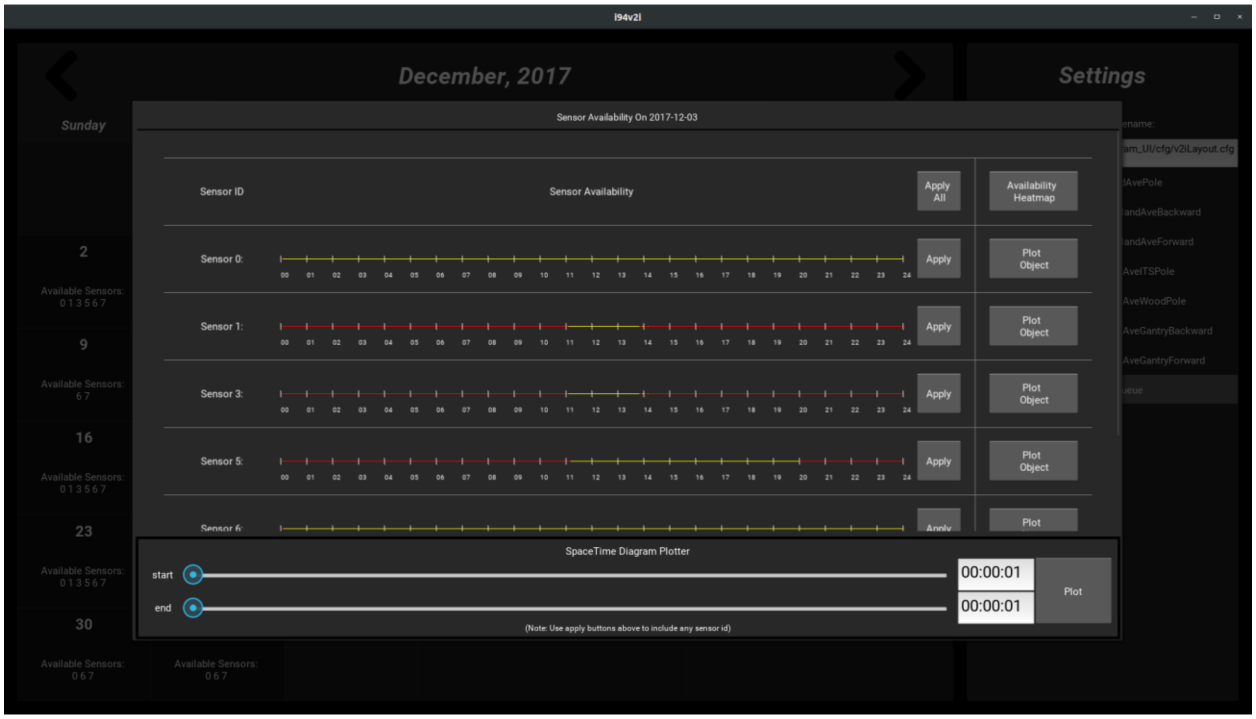


Figure C.3 Sensor Availability per hour on selected date (December 3, 2017)

Users then can select the desired start and end times of the space-time diagram by moving the “start” and “end” sliders along the bottom of the program. Figure C.4 shows an example of all sensors selected on December 3, 2017, and the start time of 11:12:01 and end time of 11:23:31 input.

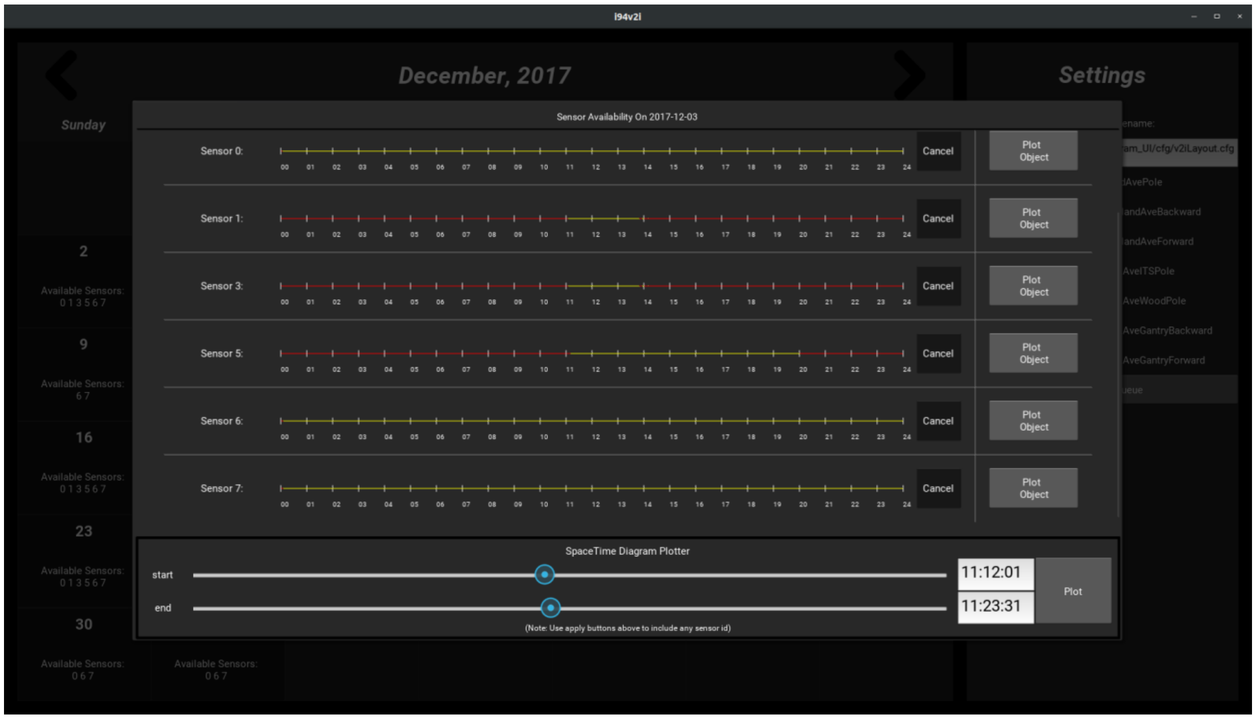
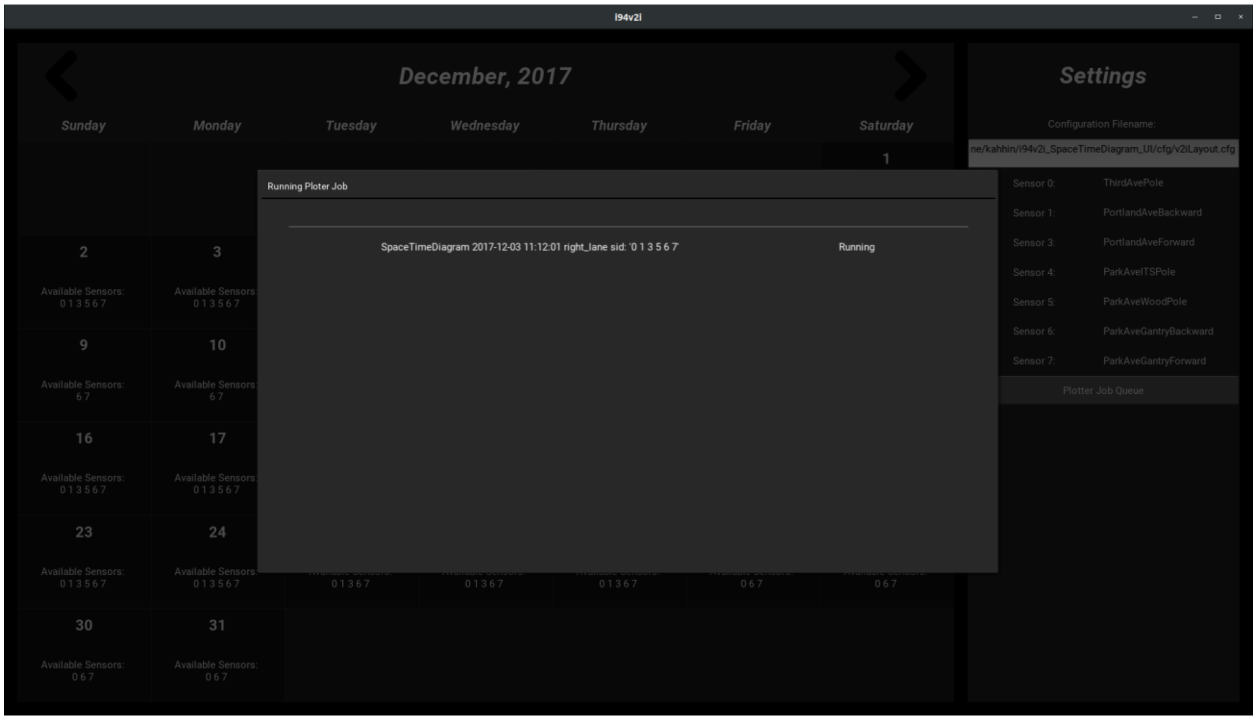


Figure C.4 Sensors have been selected and start and end times set by the user.

Once desired sensors are selected, and start and end times input, users can proceed by clicking "Plot". This tells the program to pull all available sensor data for the selected date and time, and create a space-time diagram with it. Depending on the amount of data required, the job may take a very short time or the program may need to run overnight. Status of the job can be monitored by returning to the main page and clicking "Plotter Job Queue", which will bring up a status window as shown in Figure C.5.



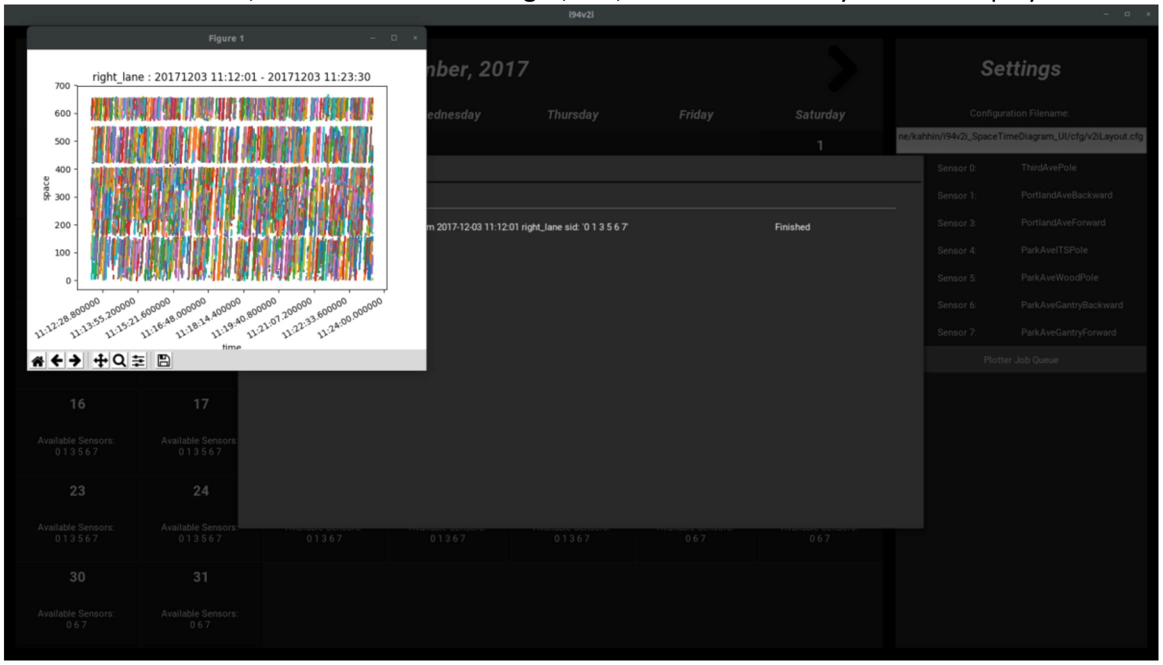
Configuration Filename: ne/kahhin/i94v2i_SpaceTimeDiagram_UI/cfg/v2iLayout.cfg

Sensor 0:	ThirdAvePole
Sensor 1:	PortlandAveBackward
Sensor 3:	PortlandAveForward
Sensor 4:	ParkAveITSPole
Sensor 5:	ParkAveWoodPole
Sensor 6:	ParkAveGantryBackward
Sensor 7:	ParkAveGantryForward

```
SpaceTimeDiagram 2017-12-03 11:12:01 right_lane sid: '0 1 3 5 6 7'        Running
```

Figure C.5 Job Status Window

One the job is finished, the graph will display as shown in Figure C.6. Users can then enlarge the graph, and zoom in and out, as well as select the right, left, middle or auxiliary lane for display.



```
m 2017-12-03 11:12:01 right_lane sid: '0 1 3 5 6 7'        Finished
```

Figure C.6 The program displaying a completed space-time graph

Figure C.7 shows the full-window display of the space time graph. Note, the middle lane is currently selected (indicated by the dark grey). The right, left and auxillary lanes can be displayed by clicking the appropriate button.

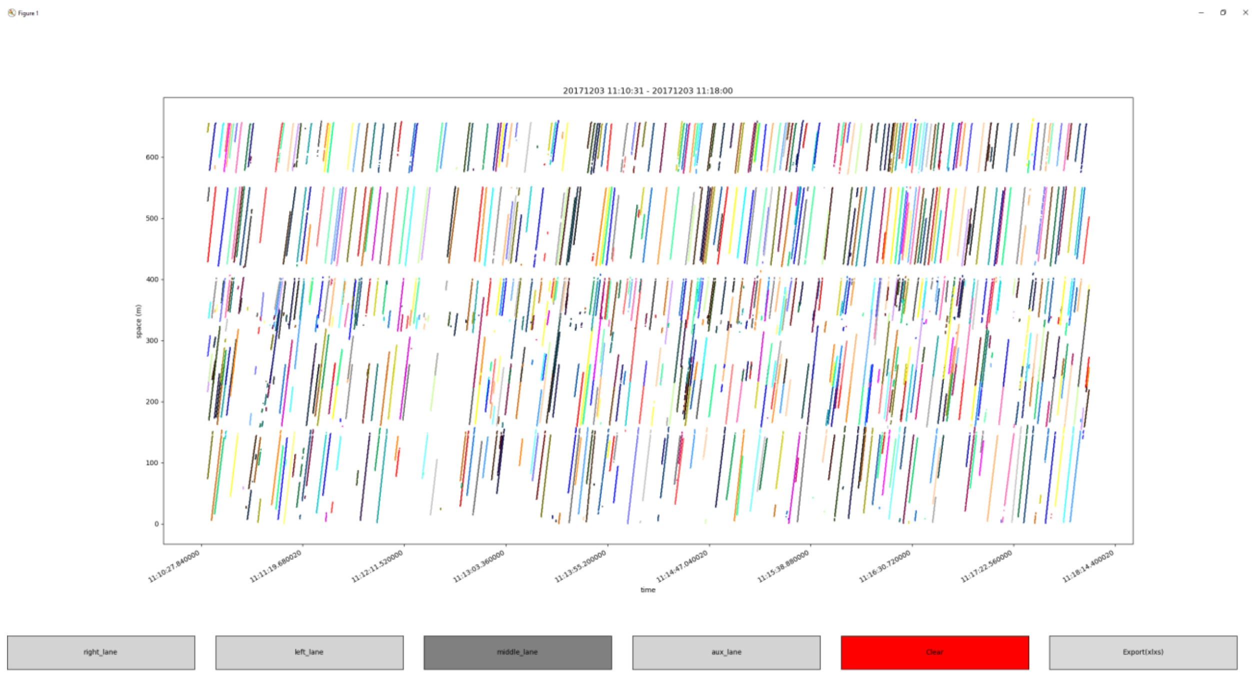


Figure C.7 The enlarged space-time diagram

From this view, users can perform the following tasks: zoom in, reset the view, move between different selected views, pan across the image (when zoomed), configure subplots (adjusts the entire graph aspect ratio), and save and export the figure. This menu is shown in Figure C.8.

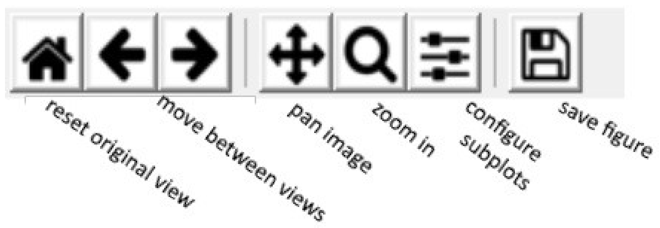


Figure C.8 Figure menu, with options labeled.

If the user wishes to select a vehicle trajectory (or multiple), they simply click on the desired trajectories in the graph. The display will change from dots to crosses, indicating the trajectory is selected, as seen in Figure C.9

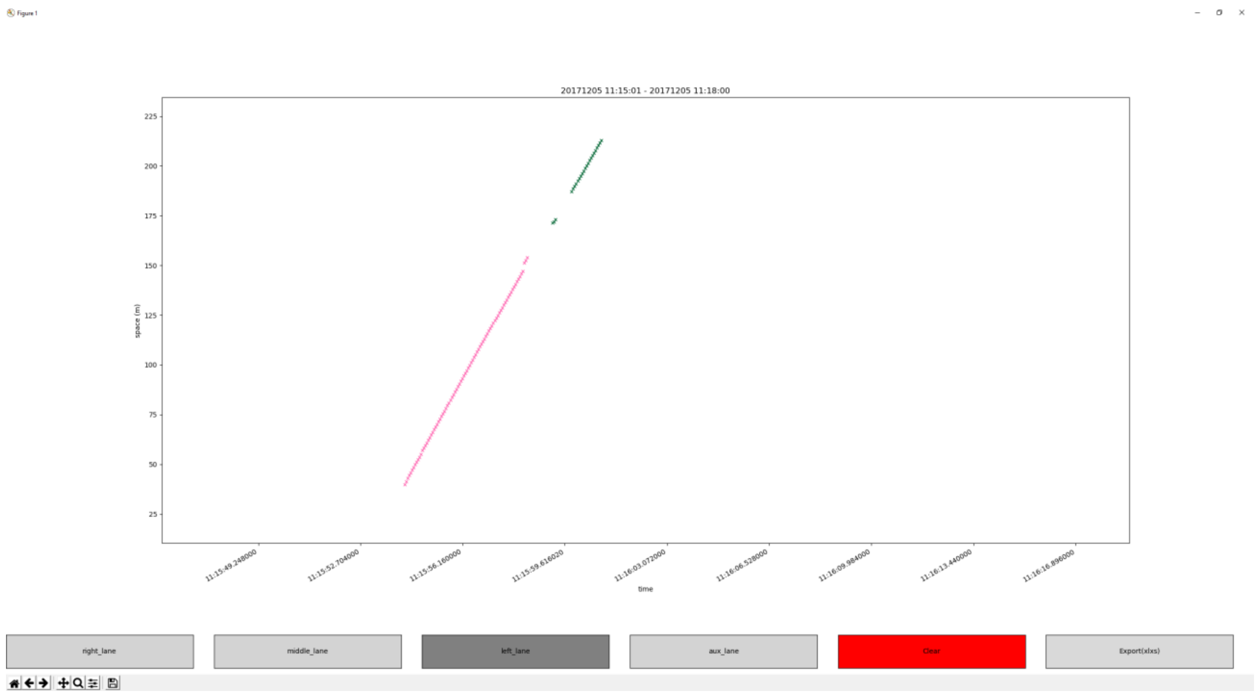


Figure C.9 Selected trajectory.

Users can then clear (red button) their selections or export the selections' raw data to a .csv file. If export is selected, the program will generate a file similar Table C.1, showing the time, count, vehicle ID, sensor ID recording the measurement, position, magnitude vector, and global and local location.

Table C.1 Output of selected trajectory, showing time, vehicle ID, sensor ID, position and velocity.

	timestam	count	object_id	sensor_id	x	y	vx	vy	v (magnitu	length	global x	global y	latitude	longitude
0	1.51E+09	3.21E+08	32	7	104.45	-6.72	-26.9	0	26.9	3.2	606.9528	-58.7051	44.96601	-93.2634
1	1.51E+09	3.21E+08	32	7	102.91	-6.78	-26.9	0	26.9	4.4	605.4128	-58.7651	44.966	-93.2634
2	1.51E+09	3.21E+08	32	7	101.12	-6.53	-26.9	0	26.9	5	603.6228	-58.5151	44.96601	-93.2634
3	1.51E+09	3.21E+08	32	7	99.84	-6.21	-26.9	0	26.9	5	602.3428	-58.1951	44.96601	-93.2634
4	1.51E+09	3.21E+08	32	7	98.5	-5.95	-26.9	0	26.9	5	601.0028	-57.9351	44.96601	-93.2634
5	1.51E+09	3.21E+08	32	7	97.22	-5.76	-27	0	27	5	599.7228	-57.7451	44.96601	-93.2634

One more functionality the Graphical Interface has is animated database visualization. From the main page of the software, Figure C.2, the user will click "run viewer" and input a date and time in YYYYMMDD HH:MM:SS format. On clicking play, an animation of the vehicle trajectories will be displayed over a still aerial image of the Testbed. Figure C.10 shows an example day.

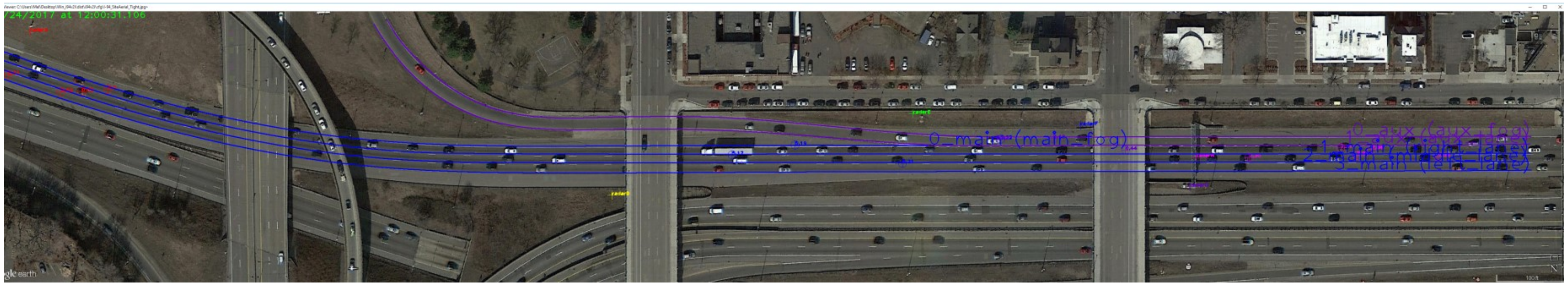


Figure C.10 Animated vehicle trajectories overlaid on an aerial image of the Testbed.

The testbed is quite long, so Figure C.10 is much more compressed than the viewer looks on a screen. The viewer has lane delineations and represents different sensor data with different colors as vehicles travel the corridor.

The user must stop the animation using space bar before closing the animation window, to prevent a program crash.