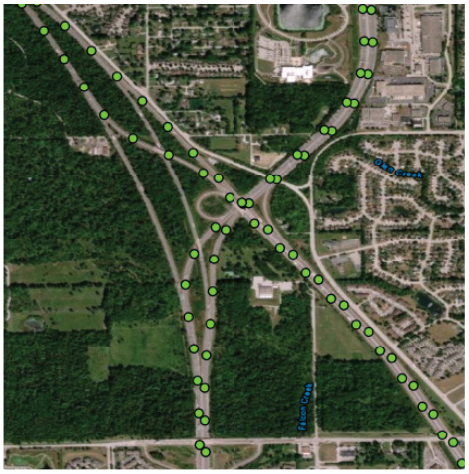
JOINT TRANSPORTATION
RESEARCH PROGRAM
INDIANA DEPARTMENT OF TRANSPORTATION
AND PURDUE UNIVERSITY

An Integrated Critical Information Delivery Platform for Smart Segment Dissemination to Road Users



Lingxi Li, Yaobin Chen, Renran Tian, Feng Li, Howell Li, James R. Sturdevant

SPR-4440 • Report Number: FHWA/IN/JTRP-2021/34 • DOI: 10.5703/1288284317440

RECOMMENDED CITATION

Li, L., Chen, Y., Tian, R., Li, F., Li, H., & Sturdevant, J. R. (2021). *An integrated critical information delivery platform for smart segment dissemination to road users* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2021/34). West Lafayette, IN: Purdue University. https://doi.org/10.5703/1288284317440

AUTHORS

Lingxi Li, PhD
Professor of Electrical and Computer Engineering
School of Engineering and Technology
Indiana University-Purdue University Indianapolis
(317) 274-3643
LL7@iupui.edu
Corresponding Author

Yaobin Chen, PhD
Director of TASI
Chancellor's Professor of Electrical and Computer Engineering
School of Engineering and Technology
Indiana University-Purdue University Indianapolis

Renran Tian, PhD
Assistant Professor of Computer and Information Technology
School of Engineering and Technology
Indiana University-Purdue University Indianapolis

Feng Li, PhD
Professor of Computer and Information Technology
School of Engineering and Technology
Indiana University-Purdue University Indianapolis

Howell Li
JTRP Senior Software Engineer
Lyles School of Civil Engineering
Purdue University

James R. Sturdevant, PE
Director of Traffic Management
Indiana Department of Transportation

JOINT TRANSPORTATION RESEARCH PROGRAM

The Joint Transportation Research Program serves as a vehicle for INDOT collaboration with higher education institutions and industry in Indiana to facilitate innovation that results in continuous improvement in the planning, design, construction, operation, management and economic efficiency of the Indiana transportation infrastructure. https://engineering.purdue.edu/JTRP/index_html

Published reports of the Joint Transportation Research Program are available at http://docs.lib.purdue.edu/jtrp/.

NOTICE

ACKNOWEDGMENTS

This work was supported by the Joint Transportation Research Program (JTRP) administered by the Indiana Department of Transportation (INDOT) and Purdue University. The authors would like to thank Tim Overman for providing invaluable guidance, support, and oversight to the data at the Traffic Management Center (TMC), Doug Meyer for facilitating and maintaining network security and access between the universities and INDOT, Wayne Bunnell for field support, ground truthing and quality assurance of the developed products, Woosung Kim and Benjamin Scholer for implementing the outcomes of the research in the web applications, Sarah Ann Guy and Rahul Sakhare for their assistance to the development of the smart segment platform, Jairaj Desai for developing the ingestion system for integrating AVL data into the dashboards, Dan Shen and Keyu Ruan for the driving simulator setup and scenario design, Zhengming Zhang for data collection and data analysis, Zahra Yarmand and Sheetal Prasanna for developing the Android-based smartphone application, and Taryn Spisak for developing the iOS-based smartphone application.

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. FHWA/IN/JTRP-2021/34	**2. Government Accession No.**	**3. Recipient's Catalog No.**
4. Title and Subtitle An Integrated Critical Information Delivery Platform for Smart Segment Dissemination to Road Users		**5. Report Date** October 2021 **6. Performing Organization Code**
7. Author(s) Lingxi Li, Yaobin Chen, Renran Tian, Feng Li, Howell Li, and James R. Sturdevant		**8. Performing Organization Report No.** FHWA/IN/JTRP-2021/34
9. Performing Organization Name and Address Joint Transportation Research Program Hall for Discovery and Learning Research (DLR), Suite 204 207 S. Martin Jischke Drive West Lafayette, IN 47907		**10. Work Unit No.** **11. Contract or Grant No.** SPR-4440
12. Sponsoring Agency Name and Address Indiana Department of Transportation (SPR) State Office Building 100 North Senate Avenue Indianapolis, IN 46204		**13. Type of Report and Period Covered** Final Report **14. Sponsoring Agency Code**

15. Supplementary Notes

Conducted in cooperation with the U.S. Department of Transportation, Federal Highway Administration.

16. Abstract

An integrated critical information delivery platform for smart segment dissemination to road users was developed. A statewide baseline milepost geodatabase was created at 0.1-mile resolution with tools, protocols, and interfaces that allow other data sources to be efficiently utilized. A variety of data sources (e.g., INRIX, CARS, Doppler, camera images, connected vehicle data, automated vehicle location) were integrated into existing and new dashboards for stakeholders to monitor roadway conditions and after-action reviews. Additionally, based on these data sources, algorithms were developed and an API was created to identify hazardous road conditions when the location of the end-user mobile device was given. Message delivery schemes were successfully implemented to issue alerts to drivers, which were integrated with two in-vehicle smartphone applications. The performance of the integrated platform was evaluated using both the driving simulator and a number of simulated and on-road tests. The results demonstrated the system was able to disseminate data in real-time using the developed platform.

17. Key Words smart segment, milepost mapping, driving simulator, real-time driver alert, motorist traffic, back of queue, application programming interface, dashboards, connected vehicle, weather data	**18. Distribution Statement** No restrictions. This document is available through the National Technical Information Service, Springfield, VA 22161.

19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 47	**22. Price**

EXECUTIVE SUMMARY

Introduction

As more datasets have been made available to transportation industry practitioners that contain increasing volumes, varieties, and frequencies, a consistent mapping scheme has become necessary for the efficient uptake, transformation, and integration of this data into existing data systems and business processes. A streamlined process can quickly turn massive amounts of data into critical information delivered to stakeholders and motorists in real-time with potentially life-saving impact (Li et al., 2015).

Commonly-occurring use cases in Indiana are secondary crash incidents that significantly impact the safety of road users. Locations with stopped queues have an increased crash factor of up to twenty-four times more than when there were no queues (Mekker et al., 2020). Recent innovations in back-of-queue detection using crowdsourced connected vehicle (CV) data, paired with statewide road status information and weather data, have allowed practitioners to monitor highway traffic conditions and respond based on the information. However, it's been a major challenge to consolidate an evolving assemblage of data sources so that user interface points—e.g., overhead and portable dynamic message sign (DMS) boards, traffic management applications, traffic information systems, mobile phones, and vehicle infotainment systems— can receive this data effectively and consistently without frequent reengineering due to changes in the underlying data sources. This project developed an information delivery platform that consistently segments data statewide to integrate and consume, generate, prioritize, and send alert messages based on the synthesis of underlying sources with the developed heuristics.

Methodology

The research team collaborated with INDOT Traffic Management Center (TMC) colleagues to inventory and consolidate existing data resources for this project. By combining resources with new techniques and applications, a statewide baseline milepost geodatabase, dubbed "smart segment," was developed. A real-time web application programming interface (API) was created to deliver roadway milepost and direction, speed, road status, and weather information to a requestor after a set of latitude and longitude points were provided. The message delivery system was developed and tested with both Android-based and iOS-based smartphone applications in the field. In addition, a driving simulator was employed to test the effects of the system on human drivers in the virtual environment.

The smart segment platform was implemented in both new and existing web dashboards to align sources of data to Indiana roads. Figure 1.1 shows the heatmap dashboard with trajectory traces of CV data linear referenced using the smart segment platform during a 12-mile queuing event on July 14, 2021. The orange, red, and pink areas are locations of slowdowns that visually allow operators to assess roadway conditions as an incident unfolded.

The human-machine interface (HMI) simulator, developed to test the back-of-queue warning system for smartphones, provided a safe and flexible environment for user evaluations on the developed applications and to document driver responses to roadway incidents without exposing drivers to real-world dangers. The results of the tests provided quantifiable data for implementations that were developed and deployed in parallel, such as the ongoing Protect the Queue efforts in Indiana (INDOT, n.d.a). Figure 1.2 shows an example of a virtual vehicle and a field-deployed vehicle for queue alerting.

Findings

The findings of this research were as follows.

1. The team successfully developed a smart segment geodatabase platform that consistently provides route, direction, and mileposts statewide for efficiently integrating existing and future data sources.
2. The team successfully developed an API for quickly processing GPS location information along sections of roadway that can be used by external systems for decision-making and information dissemination.
3. The team successfully developed three new web dashboards, one new data service, two smart phone applications, and updated two legacy applications for traffic operations and information dissemination use.
4. The team evaluated the performance of the integrated platform using both the driving simulator and on-road tests.

Implementation Recommendations

Near- and medium-term recommendations are as follows.

1. Near-term (6–18 months)
   a. Integration with winter weather operations.
   b. Integration with workzone operations.
   c. Distribution of the smartphone apps to the selected group of users.

2. Long-term (18 months or longer)
   a. Integration of the developed smartphone apps with the existing INDOT app.
   b. Assessing roadway maintenance needs.
   c. Institutionalizing geodatabase for future mapping needs.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

API	Application Programming Interface
AVL	Automatic Vehicle Location
CAN	Controller Area Network
CPU	Central Processing Unit
CV	Connected Vehicle
DMS	Dynamic Message Sign
HMI	Human-Machine Interface
HSR	Hybrid Scan Reflectivity
HTTP	Hyper Text Transfer Protocol
INDOT	Indiana Department of Transportation
IoT	Internet-of-Things
iOS	iPhone Operating System
ITAP	Information Technology at Purdue
ITS	Intelligent Transportation System
GB	Gigabyte
GIS	Geographic Information System
GNSS	Global Navigation Satellite System
GPIO	General-Purpose Input/Output
GPS	Global Positioning System
JSON	JavaScript Object Notation
LED	Light-Emitting Diode
LTE	Long-Term Evolution
LTS	Long Term Support
MAC	Media Access Control
MOT	Maintenance of Traffic
MPH	Miles Per Hour
NOAA	National Oceanic and Atmospheric Administration
OBD-II	On-Board Diagnostics Version 2
OS	Operating System
PCIe	Peripheral Component Interconnect Express
SQL	Structured Query Language
TB	Terabyte
TMC	Traffic Management Center
VPN	Virtual Private Network
XML	Extensible Markup Language
Z	Equivalent Reflectivity Factor

1. PROJECT OVERVIEW

1.1 Introduction

This project aims at developing an integrated critical information delivery platform for smart segment dissemination to road users. Toward this goal, the research team has completed the following tasks: (1) developed a statewide baseline milepost geodatabase along with the integration of a variety of data sources (see Figure 1.1); (2) developed approaches for generating alerting messages based on the integrated data, given the location of the end-user on the highway; (3) devised message delivery schemes and implemented message delivery functionalities (e.g., DMS boards and smartphone applications); and (4) evaluated the performance of the integrated platform via the driving simulator study and a limited scope of road tests (see Figure 1.2).

Based on the outcomes and deliverables developed for the research project, opportunities for near- and long-term integration of the critical information delivery platform to external systems is as follows.

1. Near-term (6–18 months)
   a. *Integration with winter weather operations.* Mapping AVL locations and processing of metrics on the smart segment platform, such as dash cam images, route and directions plowed, number of passes, and the cumulative amount of salt applied. There is also an opportunity to share pertinent winter maintenance and operations information updates in real-time to motorists through developing relationships and collaboration with the state public information office and media channels.
   b. *Integration with workzone operations.* The geodatabase can be leveraged in the near-term during the workzone season to identify critical locations for safety, queue management, and congestion and capacity management. Relationships with workzone supervisors, traffic control vendors, contractors, and Indiana State Police can be developed to ensure critical infrastructure assets (moving and stationary) such as queue trucks, barriers, lane configuration, worksite entrances, and so forth can be mapped to a unified base map.
   c. *Distribution of the smartphone apps to the selected group of users.* A selected group of early adopters can use the application during the first year of deployment. The group can provide feedback on the functionality. If feasible, a technical support entity can be identified to record and make minor bug fixes during this period.

2. Long-term (18 months or longer)
   a. *Integration of the developed smartphone apps with the existing INDOT app.* Based on near-term feedback, a larger pool of users can be identified and targeted for deployment in coordination with TMC stakeholders. The developed apps can be good additions to the existing INDOT smartphone applications that can be widely disseminated and deployed.
   b. *Assessing roadway maintenance needs.* The smart segment geodatabase can be integrated with asset tracking applications and resources to conform numerous and varied asset data sets managed by the state. For example, sign inventory, mowing operations, and sweeping. Additionally, emerging enhanced CV data sources that provide enhanced attributes such as uneven wheel speeds, headlights, and windshield wipers during storm events, lane marking quality, and road friction values can be conformed to smart segment route and direction mileposts and integrated with existing asset systems. It is important to identify and develop relationships with public and private sector entities to ensure successful implementation.
   c. *Institutionalizing geodatabase for future mapping needs.* Develop strategy and tactics with TMC and central office data owners to establish protocols for

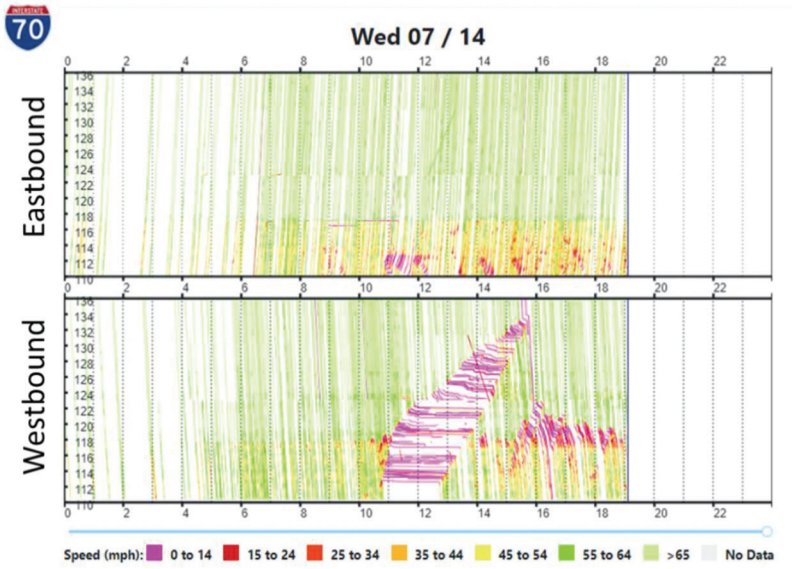


Figure 1.1 Connected vehicle (CV) position data colorized by speed, automatically referenced to 0.1-mile segmented geodatabase on Interstate 70, mile 110 to 138.

(a) Simulated back-of-queue vehicle

(b) Field deployment of back-of-queue vehicle, April 2021

Figure 1.2 Back-of-queue alert in simulation and in the field.

conforming new data sources as they are developed to expedite any potential mapping processes by making use of the statewide smart segment milepost database.

1.2 Dissemination of Research Results

The following lists papers prepared in part during the course of this project.

- Zhang, Z., Shen, D., Tian, R., Li, L., Chen, Y., Sturdevant, J., & Cox, E. (2021, September). Implementation and performance evaluation of in-vehicle highway back-of-queue alerting system using the driving simulator. *Proceedings of the 2021 IEEE International Conference on Intelligent Transportation Systems,* pp. 1753–1759.
- Shen, D., Zhang, Z., Ruan, K., Tian, R., Li, L., Li, F., Chen, Y., Sturdevant, J., & Cox, E. (2021, January). *Assessing the effectiveness of in-vehicle highway back-of-queue alerting system* [Paper presentation]. 2021 Transportation Research Board Annual Meeting, Washington, USA.

The technical papers were prepared throughout the project and distributed to INDOT stakeholders to communicate research milestones, and to facilitate early implementation of the research findings. The following sections of the technical report summarize key findings from these papers.

2. DEVELOPMENT OF STATEWIDE MILEPOST GEODATABASE

The statewide smart segment geodatabase was developed as a baseline mapping system for integrating existing INDOT GIS resources as well as external geospatially-referenced data resources. The initial task of identifying, inventorying, and establishing access to resources needed from the TMC was performed. Hardware and software resources, network connections, and code repositories were identified and procured as part of the initial phase. Then, an interstate and state route milepost database were developed using existing GIS resources and applying a new methodology to synthesize and quality check for anomalies. Finally, data from external sources were identified and integration with the smart segment geodatabase was performed.

2.1 Procured Resources

Four different resource categories were procured for the development and testing of the statewide milepost geodatabase system (smart segments). Hardware and software resources were procured for mapping, integrating, storing and serving the data. Additionally, network connectivity to the TMC was established at the Purdue University traffic lab for the transfer of geographic data that facilitates the mapping process. Finally, code repositories for the project were created on Purdue and IUPUI GitHub services to store the applications' source code and to keep track of change revisions during development and testing.

2.1.1 Hardware for Running Host Services

Two Linux servers were procured to (1) host the database service for the data schema and data; (2) run an automated service to download traffic speed data from the cloud and associate the speeds to smart segments in near-real time; and (3) run a web service to deliver real-time speed, weather, and road condition data on request from any number of message delivery applications.

The team made use of existing hardware infrastructure and processing capacity available at Purdue. One server hosted the database system while the other server hosted the download service and web service. The database host was a two-processor, 24-core Intel Xeon system with 32 GB of RAM and 1.7 TB of hard disk storage capacity running on a RAID 10 configuration. An additional upgrade of two 2 TB SSD cards using PCIe adapter plug-ins increased the speed and capacity of the system for storing and querying of the speed data.

The data download and web service server was a single-processor, 8-core Intel Xeon system with 16 GB of RAM and 11 TB of hard disk a RAID 5 configuration. The host shares processing and storage capacity with other services and applications and does not fully make use of the machine's resources.

2.1.2 Software for Database and Web Hosting

The Linux OS software used for the servers was Ubuntu Server 18.04.4 LTS. The database software used was PostgreSQL version 10.17. The web hosting software used was Nginx version 1.14.0. Both the automated traffic speed download service and web service was written in Python 3.6.9.

2.1.3 VPN Connectivity to the TMC

A VPN was established between the Purdue traffic lab and the TMC network via an encrypted connection to allow for computing resources to be accessed across the two networks (see Figure 2.1). The VPN was used to transfer Indiana milepost, route polyline, and ITS asset information from the TMC network, and provide access to the developed schemas and real-time speed data to the TMC stakeholders.

2.1.4 GitHub

Code repositories were created at Purdue University's GitHub server and IUPUI's GitHub server to store the source code and subsequent modifications, improvements, revisions and forks throughout the development and testing phases of the project (GitHub, n.d.). The GitHub repository also allowed code to be shared internally with team members, between investigators and to stakeholders.

Figure 2.1 INDOT and Purdue teams at Purdue traffic lab during VPN setup.

2.2 Data Sources

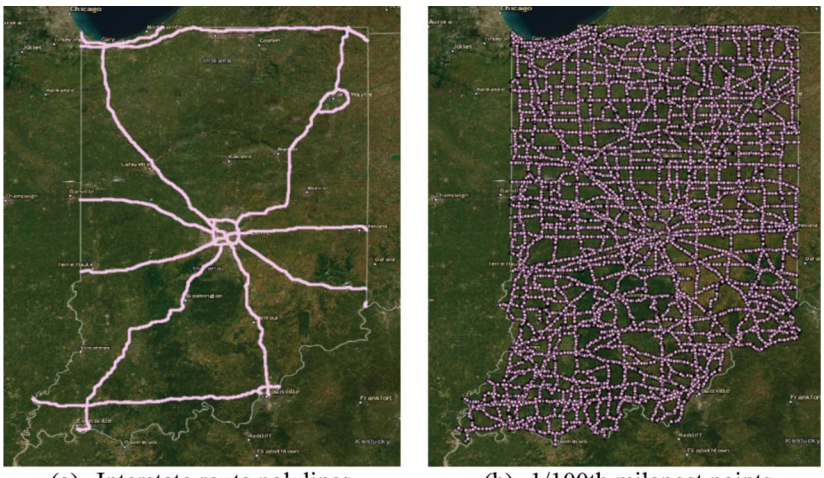
Five data sources were used to develop the smart segment and message delivery system. The data were procured from open-source repositories, TMC resources, a proprietary traffic speed API, and connected vehicles.

2.2.1 Indiana GIS Assets

Indiana milepost, route polyline, and ITS asset information were retrieved from the TMC database to expedite the development of the smart segment geodatabase (see Figure 2.2). The milepost data is a collection of GIS points that include route name, direction, and milepost at 1/100th mile increments for each point. The route polyline data consists of contiguous GIS polylines of interstate routes statewide that can be subdivided into any spatial fidelity. The milepost and polyline data were used for the development of the smart segment geodatabase. Additionally, the ITS asset information consists of GPS point locations of moving assets such as Hoosier Helper vehicles and snow plows, as well as stationary assets such as ITS cameras and speed sensors. A mix of both TMC database and external vendor sources were brought in. Outputs from these data were later integrated into the platform geospatially for real-time asset and performance after-action reviews.

2.2.2 INRIX

Proprietary route segmentation and its associated traffic speed data for the State of Indiana were retrieved from the INRIX web application and API (INRIX, n.d.). The speed data were provided for directional segments of roadway 250 meters in length (INRIX, 2013).



(a) Interstate route polylines (b) 1/100th milepost points

Figure 2.2 Existing INDOT GIS resources from TMC database.

The data were downloaded from the API using an automated service at 1-minute intervals for the entire state. The speed data were mapped to the smart segment geodatabase in near real-time.

2.2.3 CARS

Real-time traveler information data is retrieved from the Indiana CARS Program (INDOT, n.d.b) through an API. The data was provided in XML format and retrieved from the CARS 511 API every minute. The data contains roadway status information such as restrictions, roadwork, closures, incidents, and winter events.

2.2.4 Doppler

Doppler data at 1-kilometer, 2-minute resolution from the National Severe Storms Laboratory online file repository (NSSL, n.d.a). The data provides reflectivity values in decibels relative to Z (dBZ), precipitation type, and quantitative precipitation estimation (QPE) in millimeters-per-hour. The data were mapped to the smart segment geodatabase to provide additional information about roadway conditions for messaging alerts.

2.2.5 CV

Vehicles that are instrumented with embedded computers linked via OBD-II CAN connectors and have connections to the cellular network can upload data aligned to the smart segment geodatabase. CV data include outputs from different sensors on the vehicle such as speed, hazard lights, turn signal, brake pressure, and wheel rotation (Li et al., 2020). Additionally, third-parties have provided CV data from crowdsourcing that includes vehicle position, heading, and speed, time stamp, and when accelerometer values cross certain thresholds (Desai et al., 2021; Li et al., 2016). Performance measures from these other vehicles provide additional information about roadway conditions for messaging alerts to the public.

2.3 Development of the Smart Segments Geodatabase

The Indiana milepost and route polyline data were used to develop the smart segments geodatabase. A smart segment consists of a spatial point with latitude and longitude, a route name, route direction, milepost, and heading azimuth of the road at that physical location. The resulting geodatabase contained sets of spatial points for each state-owned route, US highway, and interstate in Indiana. A combination of point and polyline data was used for synthesizing the smart segments.

2.3.1 Creating Smart Segment Points

The source point data consisted of route and milepost locations at 1/100th of a mile, while the source polyline data were contiguous for each direction of travel of each route. The data were retrieved from the TMC database and loaded on to the PostgreSQL host at the Purdue traffic lab. A method using ArcMap created the smart segments by (1) segmenting the polylines into 0.1 mile sections for each route and direction of travel; (2) creating endpoints using each 0.1 mile section that will be used as smart segment points; and (3) assigning milepost attribute to each newly-created smart segment point by joining to the nearest source point containing milepost location (see Figure 2.3). The resulting output consisted of an ArcMap layer of points with latitude, longitude, route name, route direction, and milepost attributes.

Additionally, quality checks were performed on the source data to look for (1) inconsistent spacing between points that needed adjustment; (2) missing points that needed to be filled in manually; (3) extra points that needed to be deleted; and (4) overlapping polylines on opposite directions on a single route that needed to be separated.

2.3.2 Loading Smart Segments Into a Geodatabase

Once the smart segment points have been compiled in ArcMap layers, using the program's built-in feature the points were then exported to PostgreSQL. Each point was assigned a unique integer identifier called the reference point identifier. Finally, using pairs of consecutive smart segment points, vectors were created between each point to determine the directionality (heading) of each point on a roadway. This heading was used to integrate various data sources when ensuring the correct alignment between the road and a vehicle's direction.

2.4 Integration of Data Sources

Data sources integrated with the smart segment platform were categorized as either statically- or dynamically-mapped. Statically-mapped sources were datasets that had pre-defined locations for which data was provided over time, and the mapping did not change or rarely changed throughout the life of the application. Dynamically-mapped sources consisted of vehicle or event data that were reported with specific GPS locations and needed to be referenced on-the-fly to the smart segment geodatabase.

2.4.1 Statically Mapped Sources

Two statically-mapped sources were integrated for this project: INRIX speed data and doppler data. INRIX speed data were retrieved on a minute-by-minute basis for pre-defined segments of roadway statewide. A mapping table within the PostgreSQL database was created to associate INRIX segments to smart segment points. This mapping table was updated bi-annually to correspond to the vendor's map update schedule. Once the mapping table was created, each speed record was then stored with a smart segment reference identifier.

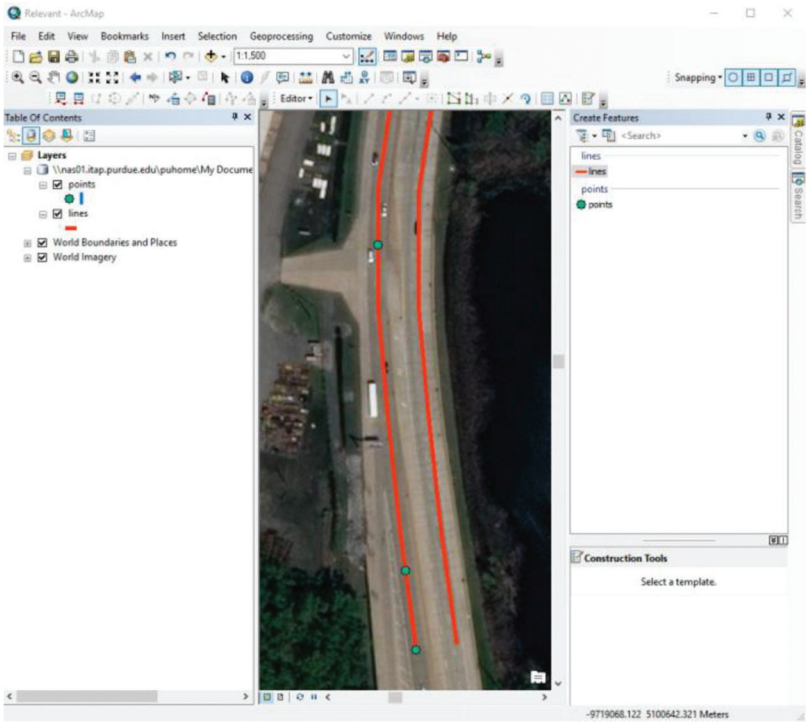


Figure 2.3 Creating smart segment points using directional road polylines in ArcMap.

The doppler data retrieved from the NSSL repository were referenced to geographic center points at 1-kilometer intervals that did not change over time. A mapping table was created to perform the translation to smart segment points. These translations were performed during the download process as new data were retrieved and stored in the database.

2.4.2 Dynamically Mapped Sources

The following four dynamically-mapped sources were integrated for this project: (1) CARS real-time traveler information; (2) connected vehicles; (3) Hoosier Helpers; and (4) snow plows. These data sources were provided through web APIs or a direct database connection interface and contained latitude and longitude attributes. This location was where a vehicle or event was detected via on-board sensors or as reported by traffic management operators.

To associate locations of incoming data to points in the smart segment geodatabase, a multi-dimensional search tree was developed to facilitate an efficient mapping process (Bentley, 1975). As this data were coming in real-time at up to 10,000 to 20,000 records-per-second, developing a system with sufficient capacity was high-priority. A Python implementation making use of libpysal was developed to perform this task (Libpysal, n.d.). The points from the smart segment geodatabase were loaded into a k-d tree data structure in a Linux service application, and lookups were made against this data structure. Once a matching smart segment point was found, the associated attributes of route name, milepost, and heading were applied to the incoming data.

3. MESSAGE DELIVERY AND PRIORITIZATION PLATFORM

3.1 Overall Architecture

Based on the developed statewide smart segment geodatabase and integrated data sources, a message delivery and prioritization platform is devised, which includes an information dissemination software platform, message delivery protocol, and prioritized messages based on the locations and use cases. The overall structure of the platform is illustrated in Figure 3.1.

3.2 Real-Time API for Data Retrieval

An API was developed by Purdue traffic lab to enable mobile applications to retrieve speed, event, and doppler data based on current device location. The data were then used by the messages and prioritization system to determine the appropriate message for dissemination using the developed heuristics.

Figure 3.2 shows an example HTTP POST request to the API. The request body contains a raw JSON-encoded element with the following attributes.

1. *App*: the application used for the call ("1" for static devices and "2" for mobile devices).
2. *Mac*: the unique MAC address of the device making the request.
3. *lat1/lon1/lat2/lon2:* a pair of consecutive latitude and longitude coordinates from the device.

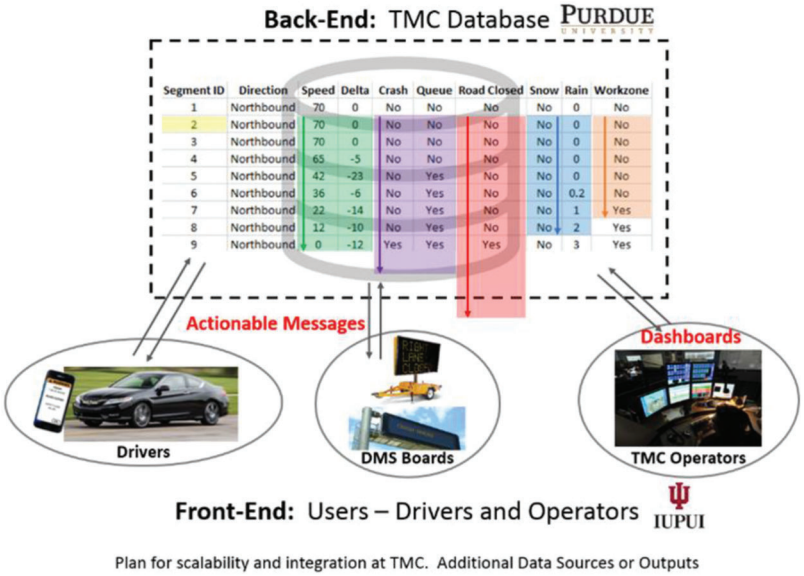


Figure 3.1 Integrated data sources platform and optimized message delivery.

```
{
    "app": 2,
    "mac": "00-14-22-01-23-45",
    "lat1": 39.297910,
    "lon1": -85.967592,
    "lat2": 39.296988,
    "lon2": -85.967571,
    "lookAheadMiles": 0.25
}
```

Figure 3.2 HTTP POST request to API to retrieve downstream speed.

4. *lookAheadMiles:* the number of miles ahead on the matched route to return data for.

The request is processed by the web service, and if a matching smart segment point is found using a proximity and vector heading match then the speed data, CARS event (if any), and doppler data are returned to the requester.

3.2.1 Speed Data

If a matching smart segment point is found in the proximity of the latitude and longitude pair of the request in the correct direction of travel, speed data associated from the INRIX segments up to the distance downstream on the roadway as defined by the *lookAheadMiles* parameter. Figure 3.3 shows an example JSON response that includes the closest waypoint's attributes, associated route attributes of name, direction, and milepost, with three individual downstream smart segment point elements including identifier number, speed, and time stamp of the latest data along the route in an array under the *speeds* element. Additionally, a vector is computed from the pair of latitude and longitude points provided by the request, and a heading value is derived and returned in the *veh_heading* parameter. The distance from the second provided latitude and longitude pair to the closest smart segment point is also returned in the *waypoint_dist_to_point2* parameter.


```
{
    "waypoint_id": 6801,
    "waypoint_lat": 39.297826,
    "waypoint_lon": -85.967572,
    "waypoint_dist_to_point2": 93.02,
    "routename": "I-65 S",
    "mile_marker": 75.1,
    "veh_heading": 178,
    "speeds": [
        {
            "waypoint_id": 6801,
            "waypoint_lat": 39.297826,
            "waypoint_lon": -85.967572,
            "mm": 75.1,
            "route_heading": 178,
            "tstamp": "2021-07-01 14:52:20",
            "speed_mph": 63
        },
        {
            "waypoint_id": 6802,
            "waypoint_lat": 39.296341,
            "waypoint_lon": -85.967521,
            "mm": 75.0,
            "route_heading": 167,
            "tstamp": "2021-07-01 14:52:20",
            "speed_mph": 63
        },
        {
            "waypoint_id": 6803,
            "waypoint_lat": 39.29489,
            "waypoint_lon": -85.967107,
            "mm": 74.91,
            "route_heading": 157,
            "tstamp": "2021-07-01 14:52:20",
            "speed_mph": 63
        }
    ]
}
```


Figure 3.3 HTTP JSON response from API with speed and milepost information.

3.2.2 CARS Data

In addition to speed data, CARS event data is also returned by the API. Figure 3.4 shows the web dashboard and XML source from the CARS 511 data repository for Indiana. The repository updates the data

from the XML repository in real-time and the data is associated to a downstream milepost within the proximity of *lookAheadMiles* downstream of the vehicle when a call is made to the Purdue traffic lab API.

If a CARS event is found, an additional element of *CARS_event_msg_nbr* containing an array of one or more CARS event numbers is returned in a *speeds* element such as the example in Figure 3.5. The details of any number of included CARS events are returned at the parent level of the JSON as an array element *CARS_events*, with the extent of the event in terms of the mileposts, type and message of the event, and event expiration. An example of this element is provided in Figure 3.6.

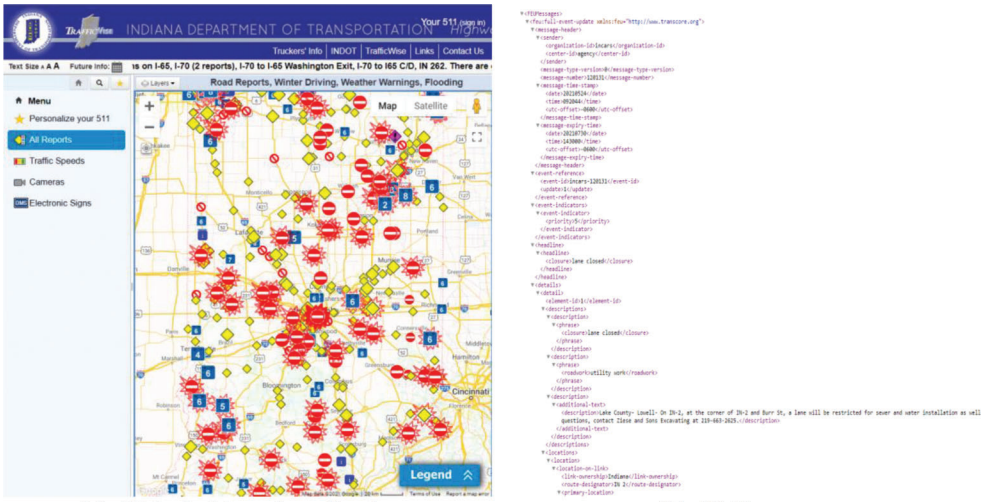
3.2.3 Doppler Data

If a doppler data indicates that precipitation is detected within the look-ahead proximity downstream of the supplied latitude and longitude coordinates in the request, an additional *MRMS* element is provided to the individual *speeds* element. Figure 3.7 shows classifications of precipitation intensities from the National Weather Service and their associated dBZ and precipitation (in millimeters-per-hour) values that are used for the classification in the returned data.

Additionally, the MRMS data contains a precipitation type flag that describes the type of precipitation witnessed. Table 3.1 shows the enumerated flag values and their corresponding precipitation types. Figure 3.8 shows an example response JSON element that describes the doppler data. The *MRMS* element is nested in an individual speeds element and contains the time stamp of the sample taken, reflectivity value, precipitation flag, precipitation rate in millimeters-per-hour, and the past hour's cumulative precipitation amount.

3.3 Message Prioritization Heuristics and Platform

The first approach deployed for message prioritization is the Decision Tree (Kamiński et al., 2017), which is a basic classification and regression method. In a



(a) Web dashboard application (b) XML source

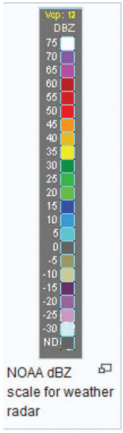
Figure 3.4 Indiana CARS 511 interfaces.

```
"CARS_event_msg_nbr": [
        104050
]
```

Figure 3.5 HTTP JSON response element for CARS event message number.

```
"CARS_events ": [
    {
        "mmmin": 157.44,
        "mmmax": 158.45,
        "type": "restriction",
        "msg": "restrictions",
        "msg_text": "bridge shoulder strengthening will be conducted at each of the
north and south bound lanes right shoulder. Restriction time frames are from 9:00 PM
to 6:00 AM with only 1 lane restriction at any given time. \nContract: B-39578",
        "msg_nbr": 104050,
        "expire_time_utc": "2020-04-17 10:00",
    }
]
```

Figure 3.6 HTTP JSON response element describing a CARS event.



NOAA dBZ scale for weather radar

Reflectivity in dBZ versus Rainrate

L_Z (dBZ)	*R* (mm/h)	*R* (in/h)	Intensity
5	0.07	< 0.01	Hardly noticeable
10	0.15	< 0.01	Light mist
15	0.3	0.01	Mist
20	0.6	0.02	Very light
25	1.3	0.05	Light
30	2.7	0.10	Light to moderate
35	5.6	0.22	Moderate rain
40	11.53	0.45	Moderate rain
45	23.7	0.92	Moderate to heavy
50	48.6	1.90	Heavy
55	100	4	Very heavy/small hail
60	205	8	Extreme/moderate hail
65	421	16.6	Extreme/large hail

Figure 3.7 Doppler reflectivity values and intensities (Desktop Doppler, n.d.).

TABLE 3.1
Precipitation classification in HSR data (NSSL, n.d.b)

Flag	Description
-3	No coverage (value used depends on file format MRMS binary vs. GRIB2)
0	No precipitation
1	Warm stratiform rain
3	Snow
6	Convection
7	Hail
10	Cool stratiform rain
91	Tropical stratiform rain
96	Tropical convective rain

Flag values 2, 4, 5, 8, 9 are not being used at this time.

decision tree, a set of classification rules are summarized from the given training set to correctly classify the target set. For this project, the following key attributes from different data sources were first identified: (1) the delta speed of adjacent waypoints is the key information for segment speed via INRIX; (2) the headline types and messages are key information via CARS; and (3) the extreme weather conditions such as tropical rain and heavy hail are key information from the doppler data. The data from INRIX, CARS, and doppler were combined and studied to generate possible alerting messages. The disadvantage of this approach is the message is highly dependent on the pre-determined thresholds on data. So, it is not a good option as it hard-coded the messages to be sent. Thus, the Decision Tree-based approach was not adopted for the final implementation.

The second approach we use is to categorize the messages based on severity for information delivery. Each message type is defined in different levels: L1-Informational, L2-Low, L3-Moderate, L4-High, L5-Critical. Messages with the highest severity category will be sent out. Since CARS data lack quantitative measures on event types, the weather data can be seen by drivers so a long message can increase their distraction level; the emphasis is put on the segment speed data via INRIX. The current threshold setting for alerting message delivery is 0.5 or 1 mile from the queue. This distance-based alerting worked very well according to the subject feedback in the driving simulator study.

3.4 Smartphone Applications for Message Delivery

3.4.1 Message Delivery System—Android-Based Smartphone Application

3.4.1.1 Introduction. The objective of this task is to use highway traffic database information to find queues in the heading direction of the user. Once queues are located, they are shown on the map. Then, from as far as 1 mile before the queue, users are alerted of queues with information about the distance of the slowdown. An Android app was developed to achieve the goal of message delivery. An emulator and phone were used to test the app by simulating routes and by testing the app live on the road.

3.4.1.2 Technical approach

3.4.1.2.1 Android studio. The software for the Android app was developed using Android Studio, the official integrated development environment (IDE) for Android application development. It is based on the IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools. Android Studio is available for Mac, Windows, and Linux desktop platforms. Since we used MacBook Pro for coding in this project, we used the Mac version of Android Studio (version 4.1.1).

3.4.1.2.2 Existing algorithm. The preliminary application used real-time traffic data from INDOT via the Purdue API to get data about queues. It also used the Google Maps API to recognize interstate names and the phone's internal GPS to access the current location and heading direction. From the obtained information, calculations were made to get delta-speed events and alerts were issued accordingly (see Figure 3.9).

3.4.1.2.3 Improvements

1. *Notification and Location as Service Threads*
   Once the code was functioning as expected some of the tasks were split into separate threads to improve responsiveness and smoothness of app functionality. The notification task, which checks for delta-speed events and generates alerts for the user, was split into its own thread. The thread is started in the main activity and goes on forever unless the user forcefully quits, as explained below. The location service thread is responsible for retrieving the current GPS coordinates of the user and runs continuously such that the app receives the user's location even when exited. This feature of threads allows the app to alert the

```
"MRMS": {
   "mrms_tstamp": "2020-06-04 11:32:00",
   "dbz": 24.3,
   "precipFlag": 1,
   "precipRate": 1.9,
   "RadarOnlyQPE01H": 1.2
}
```


Figure 3.8 HTTP JSON response element from API for doppler data.

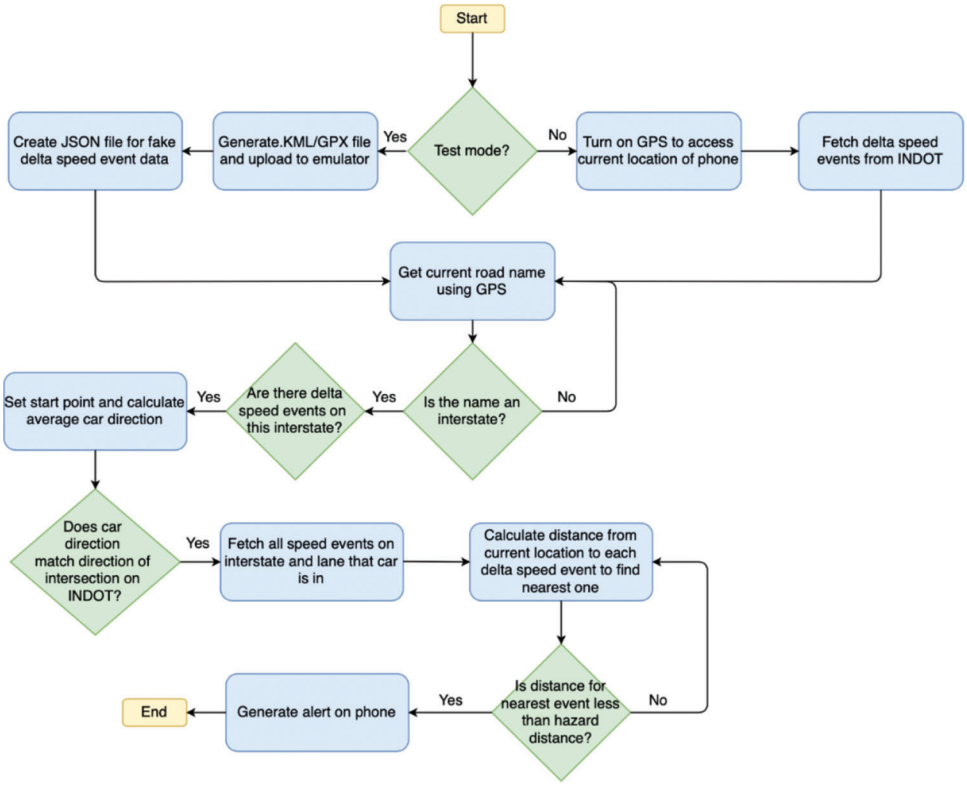


Figure 3.9 Flow chart of existing algorithm.

user of slowdowns whether the app is in the foreground or background.

2. *Restart*
   In order to run notifications in the background, a broadcast receiver was used. When the app is exited, the code enters the onDestroy method. Instead of stopping the app, however, we call the broadcast receiver with a new intent. This receiver starts both the notification and location services again. This way, even when the app is exited, the functionality continues to be executed. The restart feature allows users to enjoy the app's benefits without actively opening and running it each time.
3. *Force Quit Button*
   While the background functionality is automatically enabled, users must have the ability to force the app to quit when they do not want notifications. Therefore, a button was added to allow the user to truly quit the app and its functionality when needed.
4. *Continuous Notification Fix*
   One issue that persisted throughout testing was continuous notifications. This caused speech to constantly override itself, making it very difficult to understand the notification. Two changes were made to the code to fix this problem. First, if the previous notification was the same as the current one, it would be skipped. And second, the speech engine was stopped after each notification which gave the program enough time to completely speak the notification before going to the next one.
5. *Notification Till a Tenth of a Mile and Provides Fractional Values*
   A minor inconsistency was that the mileage in the notifications was being rounded to integers, so anything less than a mile would be declared a zero. This was fixed to include floating-point values rounded to two decimal places for more accuracy. Additionally, notifications are only provided until the user's location is a tenth of a mile away from the queue. When they get closer to the slowdown, notifications stop as the user should see the traffic by then. The improved process is illustrated in Figure 3.10. See Table 3.2 for a step-by-step guide to install the application.

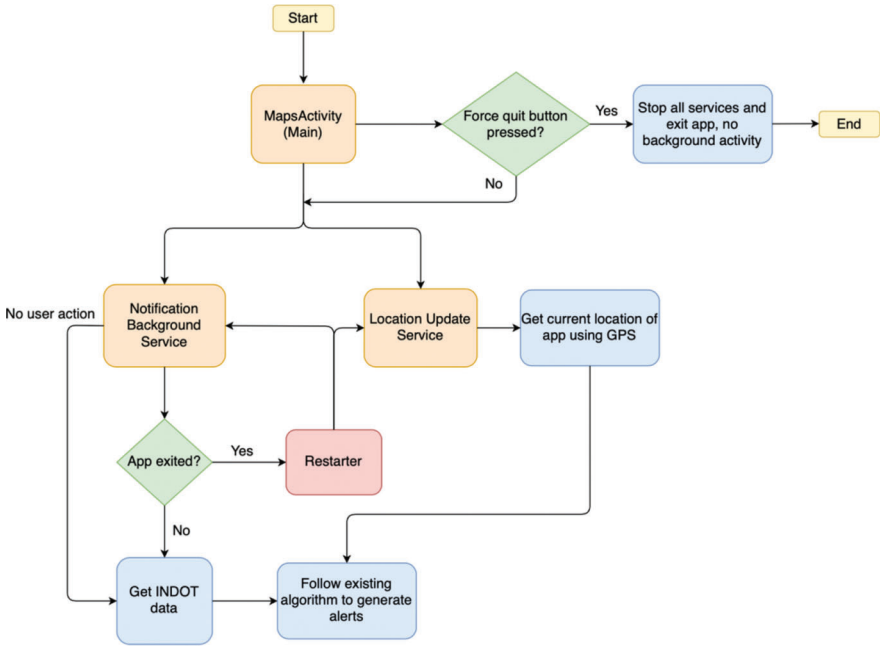


Figure 3.10 Flow chart of the improved program with threads (color coding: threads in orange, actions in blue, event checking in green, and restarter in red).

3.4.1.2.4 Results

Testing and Debugging

1. *On-road Testing*
   The app was tested on different highways at different times to check functionality. Overall, the app behaved consistently and provided warnings that were helpful for the driver to be prepared to break. The false positives were noticed to be in areas where traffic is usually dense but has reduced due to the pandemic. False positives were mostly caused by latencies and inconsistencies in the database itself. Some screenshots of the app notifications are included in Figure 3.11. Battery consumption was observed to be around 20% for a 45-minute drive which is very close to the battery consumption of Google Maps on an iPhone. Figure 3.12 shows the on-road testing results of the app. Logs display date, time, weather, highway, and details about the notification that was generated. The tabular data was analyzed to understand the efficiency of the app. From Figure 3.13, we can see that around 97% of the tests led to successful warnings. From Figure 3.14 we can see that there are around 27% of false-positive cases, which were possibly caused by database latencies. From Figure 3.15 we can see that the positive fault cases do not vary significantly when the app is open or closed.
2. *Emulator*
   Most of the preliminary testing was done using the built-in emulator on Android Studio. Routes were generated using GPX files through an online website called https://mapstogpx.com. The emulator was especially useful when adding the improvements discussed above to ensure that the main functionality was maintained. Figure 3.16 shows two screenshots of the emulator testing.

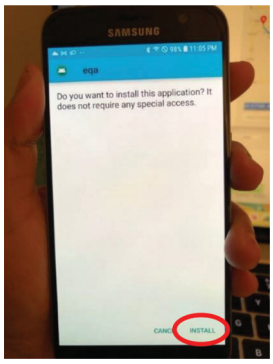
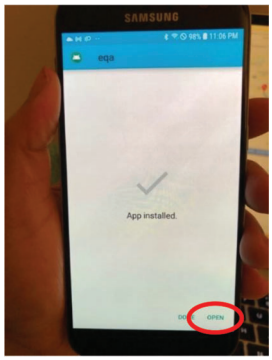
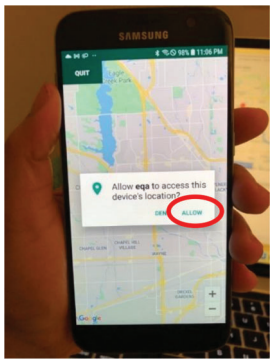
Summary

In this section, the software development of the Android-based in-vehicle message delivery system was summarized. APIs were used to obtain traffic data and the current location of users. Existing algorithms were used to determine delta-speed events, which in turn were used to issue back-of-queue alerts. Additional features such as notifications and locations in separate threads and background running functionality were discussed. Options for installation and the advantages and disadvantages of each were also summarized. Finally, testing results were shared along with the statistical accuracies of the notifications.

3.4.2 Message Delivery System—iOS-Based Smartphone Application

3.4.2.1 Introduction. To calculate the highway delta-speed events in Indiana, we developed an application in Swift to call the Purdue API and fetch data every couple of seconds. The application will handle calculations, repetitive event points, connecting to the API server, and obtaining the user's location. Similar to that of the Android testing, the project plan has two phases of testing in iOS. The first one is testing the core algorithm using manually generated GPS coordinates but real-time data in a simulated environment (shown

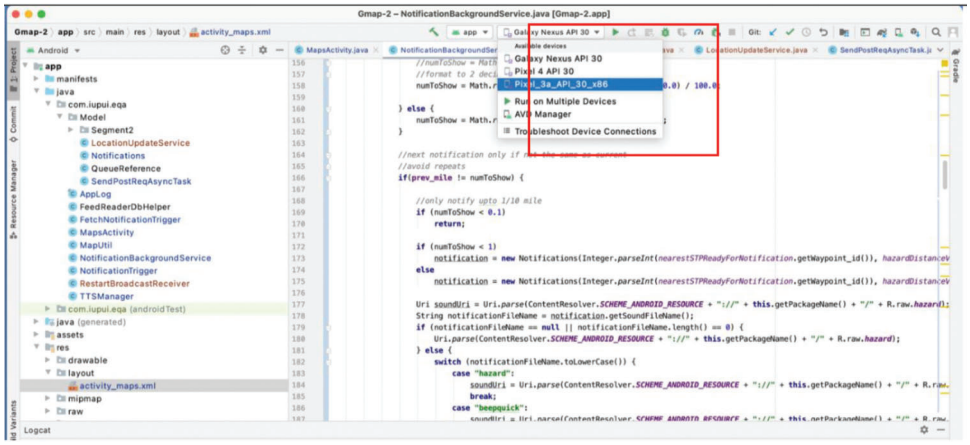
TABLE 3.2
Installation guide of the Android-based application.

Step 1: Install Android application package (apk)
1. Download the apk from Google Drive or Teams to your phone.
2. Your phone will ask you if you want to install—click "Install."

3. The app will get installed and you can open it.

4. When the app is first opened, your phone will ask you to permit it to use location—click "Allow."

5. Ensure notifications and location services when the app is closed are enabled. The advantages of installing an apk is that it easy shareable and does not require a laptop. The disadvantage is that each time changes are made to the code, the apk will need to be re-generated and re-shared.
Step 2: Flash from Android Studio
1. Open Android Studio project. 2. Connect your phone to your laptop using a USB cable. 3. From the devices dropdown, choose your phone.

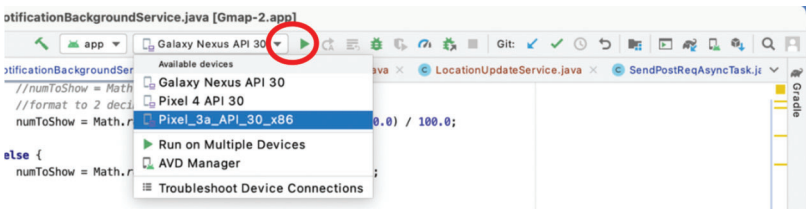
Continued on next page

TABLE 3.2
(Continued)

Step 1: Install Android application package (apk)



4. Click the green run button in the toolbar to flash the program.



5. The app will start running once the code is built.
   The advantage of installing the app by flashing it directly to the phone is that it is fast. Modifications in the code can be applied and the app can quickly and easily be re-flashed. The disadvantage is that the entire program will need to be shared, which is inconvenient.

(a) I-465 N near exit 27 (app open)

(b) I-465 N between exit 115 and 116 (app closed)

Figure 3.11 Android-based app message notifications.

in Figure 3.17). The second one is issuing an alert message to the user by driving on the road and receiving real-time traffic data (shown in Figure 3.18).

3.4.2.1.1 Technical approach. To access the Purdue API data successfully, several steps needed to be implemented. Some parameters need to be set before getting the data back from the API. Once the parameters were set and acquired it would then a post request would be created. Since the API data results are returned in JSON format, the JSON needs to be decoded. Thus, resulting in creating classes for the post API that have the same properties. Once these steps were completed, then the delta-speed was available to run through the algorithm.

3.4.2.1.2 Summary. Due to the help of the Purdue API server, the application can retrieve real-time data throughout the Indiana interstates. The app will call for new data every 2 seconds to get the latest updates. This team has developed an application in Swift to fetch real-time data and calculate potential back-of-queues for the driver route. Retrieving the data from the Purdue API server is an important element for the algorithms to determine the alerts.

3.4.3 Hardware/Software Implementation for Alerting System

3.4.3.1 Introduction. The goal of the software implementation was to develop the hardware/software required for the back-of-queue alerting system. Testing and evaluating were conducted on the following hardware/software's.

Test Numbers	Date	Time	Weather	Highway Name and Direction, mm	Queue Dynamics (Slow moving or Stopped)	Reasons of Queue (workzones, construction, accidents, etc.)	Warning Successful?	App open or exited?	Issues
1	3/25/21	6:02 PM	Cloudy	I-65 N near exit 114	Slow moving	Exit merge	Yes	Open	
2	3/25/21	6:06 PM	Cloudy	I-65 N mile 118.1	Slow moving	Stopped vehicle	Yes	Open	
3	3/25/21	6:15 PM	Cloudy	I-465 N mile 20.2	Slow moving	Exit merge	Yes	Open	
4	3/25/21	6:20 PM	Cloudy	96th Street	Neither	None	Yes	Open	False positive - usually crowds in this junction
4	3/26/21	2:27 PM	Cloudy	I-465 W near exit 20	Slow moving	Exit merge	Yes	Open	
5	3/26/21	2:32 PM	Cloudy	I-65 S mile 120.4	Slow moving	Exit merge	Yes	Open	
6	3/26/21	2:35 PM	Cloudy	I-65 S mile 114.6	Slow moving	Exit	Yes	Open	
7	3/26/21	6:07 PM	Cloudy	I-65 N near exit 115	Neither	None	Yes	Open	False positive - usually crowds in this junction
8	3/26/21	6:08 PM	Cloudy	I-65 N near exit 119	Neither	None	Yes	Open	False positive - usually crowds in this junction
9	3/26/21	6:11 PM	Cloudy	I-65 N near exit 121	Slow moving	Exit merge	Yes	Open	
10	3/26/21	6:13 PM	Cloudy	I-65 N near exit 123	Slow moving	Exit	Yes	Open	
11	3/26/21	6:14 PM	Cloudy	I-465 N near exit 21	Neither	None	Yes	Open	False positive - usually crowds in this junction
12	3/26/21	6:19 PM	Cloudy	I-465 E near exit 27	Slow moving	Exit	Yes	Open	
13	3/26/21	6:22 PM	Cloudy	421 N near 96th street signal light	Slow moving	Signal light	Yes	Open	
14	4/1/21	3:35 PM	Sunny	I-465 S (exit from 421 N)	Slow moving	Exit merge	Yes	Open	
15	4/1/21	4:10 PM	Sunny	I-65 S near exit 114	Slow moving	Unsure, cause was ahead of slow down area	Delayed	Open	The 5-minute delay from Purdue database caused warning to be delayed and also caused notification pile-up
16	4/1/21	6:08 PM	Sunny	I-465 N near exit 115	Neither	None	Yes	Open	False positive - usually crowds in this junction
17	4/5/21	4:53 PM	Sunny	US 31-S ramp from Main St. Carmel (465 exit ramp)	Neither	None	Yes	Open	False positive - usually crowds in this junction
18	4/5/21	5:01 PM	Sunny	I-465 E mm 31.6 (exit 35)	Slow moving	Exit	Yes	Open	
19	4/5/21	5:30 PM	Sunny	I-465 W near exit 33	Slow moving	Joining traffic	Yes	Open	
20	4/6/21	3:50 PM	Sunny	I-465 mm 24.3	Slow moving	Exit	Yes	Open	
21	4/6/21	3:59 PM	Sunny	I-465 E mm 118.5	Slow moving	Exit merge	Yes	Closed	
22	4/6/21	4:03 PM	Sunny	I-465 S near exit 114	Slow moving	Unsure, cause was ahead of slow down area	No	Open	Either traffic was recent and database was not updated or was too close to traffic when slow down occurred
23	4/6/21	6:07 PM	Sunny	I-465 N near exit 115	Slow moving	Cop car	Yes	Closed	
24	4/6/21	6:08 PM	Sunny	I-465 N near exit 116	Slow moving	Cop car	Yes	Closed	
25	4/6/21	6:14 PM	Sunny	I-465 N exit 123	Neither	None	Yes	Closed	False positive - usually crowds in this junction
26	4/6/21	6:22 PM	Sunny	I-465 E near exit 27	Slow moving	Exit merge	Yes	Closed	
27	4/7/21	1:01 PM	Sunny	I-465 S near exit 20 (to I-65 S)	Slow moving	Exit	Yes	Closed	App crashed after some time
28	4/7/21	3:13 PM	Sunny	I-65 N near exit 115	Slow moving	Exit merge	Yes	App open but not displayed	
29	4/7/21	3:16 PM	Sunny	I-65 N near exit 119	Slow moving	Exit	Yes	Closed	
30	4/8/21	6:06 PM	Sunny	I-65 S near exit 116	Neither	None	Yes	Closed	False positive - usually crowds in this junction
31	4/13/21	3:46 PM	Sunny	I-465 W mm 26.2	Slow moving	Exit ramp	Yes	Closed	
32	4/13/21	3:51 PM	Sunny	I-465 S near exit 20	Slow moving	Exit	Yes	Open and closed	
33	4/13/21	3:55 PM	Sunny	I-65 S mm 120.2	Slow moving	Exit merge	Yes	Closed	
34	4/13/21	3:58 PM	Sunny	I-65 S mm 117.1 (exit 117)	Neither	None	Yes	Closed	False positive - usually crowds in this junction
35	4/13/21	6:10 PM	Sunny	I-65 N near exit 115	Slow moving	Exit	Yes	Closed	
36	4/13/21	6:16 PM	Sunny	I-65 N near exit 123	Neither	None	Yes	Closed	False positive - usually crowds in this junction
37	4/13/21	6:22 PM	Sunny	I-465 E near exit 27	Slow moving	Exit	Yes	Closed	
38	4/14/21	1:00 PM	Sunny	I-465 ramp to I-465 S mm 24.7	Slow moving	Exit ramp	Yes	Closed + phone locked	
39	4/14/21	1:03 PM	Sunny	I-465 S near exit 21	Neither	None	Yes	Closed	False positive - usually crowds in this junction
40	4/14/21	1:05 PM	Sunny	I-465 S exit 20	Neither	None	Yes	Closed	False positive - usually crowds in this junction
41	4/14/21	1:09 PM	Sunny	I-65 S mm 119.1	Neither	None	Yes	Closed	False positive - usually crowds in this junction
42	4/15/21	3:16 PM	Cloudy/Sunny	I-465 S near exit 20	Slow moving	Exit merge	Yes	Closed	
43	4/15/21	3:22 PM	Cloudy/Sunny	I-65 S mm 118.5	Slow moving	Exit merge	Yes	Open	
44	4/15/21	3:24 PM	Cloudy/Sunny	I-65 S near exit 114	Slow moving	Exit	Yes	Open	
45	4/15/21	6:04 PM	Cloudy/Sunny	I-65 N near exit 116	Slow moving	Exit merge	Yes	Closed	
46	4/15/21	6:06 PM	Cloudy/Sunny	I-65 N near exit 119	Slow moving	Exit	Yes	Open	
47	4/15/21	6:09 PM	Cloudy/Sunny	I-65 N exit 123	Neither	None	Yes	Closed	False positive - usually crowds in this junction
48	4/15/21	6:14 PM	Cloudy/Sunny	I-465 N near exit 23	Slow moving	INDOT vehicle + stopped car	Yes	Closed	
49	4/20/21	3:55 PM	Cloudy/Rainy	I-465 S near exit 20	Slow moving	Exit	Yes	Closed	
50	4/20/21	4:01 PM	Cloudy/Rainy	I-65 S near exit 114	Slow moving	Exit	Yes	Closed	
51	4/20/21	6:08 PM	Snow	I-65 N near exit 115	Neither	None	Yes	Open	False positive - usually crowds in this junction
52	4/20/21	6:14 PM	Snow	I-65 N near exit 123	Slow moving	Exit	Yes	Open	
53	4/20/21	6:17 PM	Snow	I-465 N near exit 23	Slow moving	Exit	Yes	Open	
54	4/20/21	6:20 PM	Snow	I-465 E near exit 27	Slow moving	Exit	Yes	Open	
55	4/27/21	3:51 PM	Sunny	I-465 S near exit 21	Slow moving	Exit	Yes	Open	
56	4/27/21	3:58 PM	Sunny	I-65 S near exit 117	Slow moving	Exit	Yes	Open	
57	4/27/21	4:00 PM	Sunny	I-65 S near exit 114	Slow moving	Pile up beause of traffic pattern change	Yes	Open	
58	4/27/21	6:12 PM	Sunny	I-65 N near exit 121	Neither	None	Yes	Open	False positive- stopped by itself
59	4/27/21	6:15 PM	Sunny	I-65 N near exit 123	Slow moving	Lots of trucks	Yes	Open	
60	4/27/21	6:24 PM	Sunny	I-465 E near exit 27	Slow moving	Exit merge	Yes	Open	
61	4/29/21	4:03 PM	Cloudy	I-65 S near exit 117	Slow moving	Exit	Yes	Open	
62	4/29/21	4:04 PM	Cloudy	I-65 S near exit 115	Slow moving	Exit	Yes	Open	
63	4/29/21	4:05 PM	Cloudy	I-65 S past exit 114	Slow moving	Pile up beause of traffic pattern change	Yes	Open	
64	4/29/21	6:12 PM	Cloudy	I-65 N mm 118.8	Slow moving	Cop car	Yes	Open	
65	4/29/21	6:15 PM	Cloudy	I-65 N past exit 123	Neither	None	Yes	Open	False positive- stopped by itself
66	4/29/21	6:16 PM	Cloudy	I-465 N near exit 21	Neither	None	Yes	Open	False positive - usually crowds in this junction
67	5/7/21	11:00 AM	Sunny	I-65 S past exit 114	Slow moving	Pile up beause of traffic pattern change	Yes	Open	
68	5/7/21	12:30 PM	Sunny	I-65 N near exit 115	Slow moving	Pile up beause of traffic pattern change	Yes	Open	
69	5/7/21	12:39 PM	Sunny	I-465 N near exit 23	Slow moving	Exit	Yes	Open	
70	5/7/21	12:41 PM	Sunny	I-465 N near ramp to I-465 E	Neither	None	Yes	Open	False positive - usually crowds in this junction

Figure 3.12 On-road test results for Android-based smartphone application.

Figure 3.13 Test results on the ratio of success to failure.

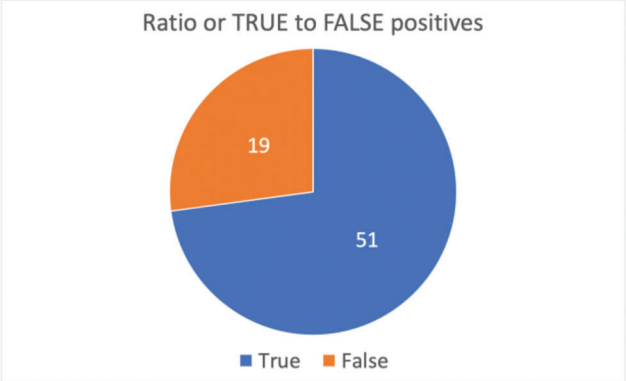


Figure 3.14 Test results on the true to false positives.

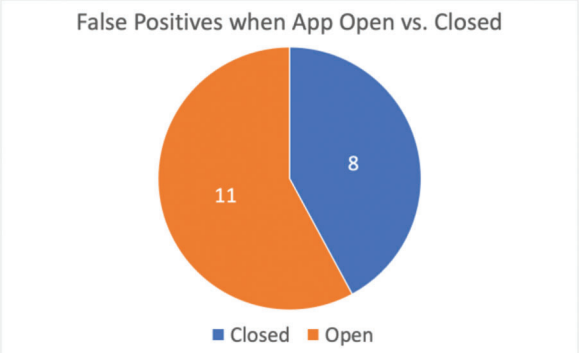


Figure 3.15 Test results on the false positives when the app is open or closed.

- Hardware: Simulated hardware environment on an iOS-based smartphone (iPhone 8).
- Software: Main functions of the iOS-based app application.
  - Accessing real-time traffic data of the speed events from Purdue API.
  - Obtaining the user location through GPS coordinate on the smartphone.
  - Determining the possibility of driving risks when the user approaches highway queues.
  - Delivering back-of-queue alerts to drivers via developed smartphone app.

3.4.3.2 Technical approach

3.4.3.2.1 Software–XCode

XCode

To create the iOS application, the software XCode was used. XCode is an integrated development environment (IDE) for iOS along with other apple products. This program supports the following programming languages: C, C++, Objective-C, Objective-C++, Java, Swift, Ruby, Python, and AppleScript. XCode requires a Mac/MacOS to run successfully. It potentially could run on a virtual machine which would result in violating Apple's terms. To implement the back-of-queue alerting systems on the iOS smartphone, a swift-based application is developed using Xcode application and an iPhone 6 or greater. A MacBook was used for the application to access XCode along with the programming language of Swift.

Map Features

Instead of using the apple map associated with the Xcode map view, there is a tile overlay of an open street map. An open street map is a free open source world-wide map. There are three buttons on the map that the user can use. There is zoom in, zoom out and zoom in and center on user location. Zoom in and zoom out buttons are restricted to a minimum center coordinate of 500 and a maximum center coordinate of 2,000. The zoom-in and zoom-out buttons have restrictions to prevent the app from crashing if the user decides to push the button multiple times. If these buttons are pressed, they do not follow the users' movement. Therefore, the zoom in and center on the user location button was created. This button, when pressed, will zoom in and center the user's location onto the screen along with adjusting the camera angle to be tilted at a pitch of 65 degrees. This gives the user better visuals of what is ahead.

Algorithm

In the subsequent section, the algorithm for the back-of-queue alerting system is described. The algorithm works off of obtaining the users' previous and current latitude and longitude points on the highway within 1 second of each other. This data is then sent to the Purdue API server to update the data. It is saved into a request body of the request. The parameters that are sent to Purdue API are users device ID, latitude and longitude one, latitude and longitude two, and having a set look ahead miles of two. Using this method with the data we receive back from the API, every 2 seconds we can calculate the delta speed events on the users' current location and notify them of a queue ahead. An algorithm is developed to find the speed events of the upcoming latitude and longitudes within the set amount of look-ahead variable value set via code. The following sections will go into specifics about each step. The flow chart of the algorithm is shown in Figure 3.19.

1. *Accessing delta-speed events*
   The first step in finding the delta speed is to retrieve the real-time traffic data from Purdue API. From the Purdue

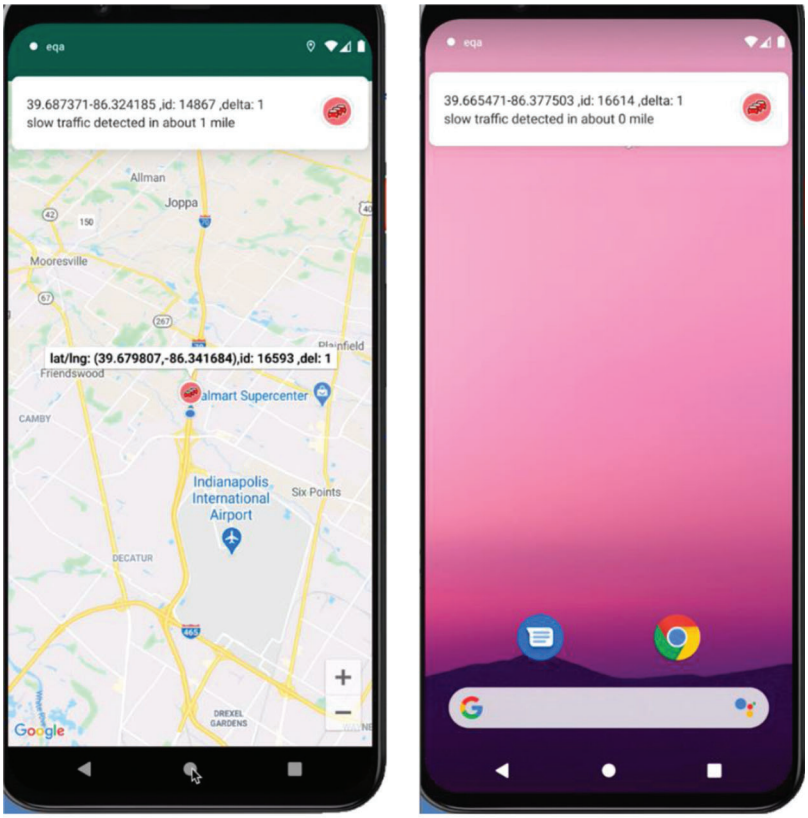


Figure 3.16 Two screenshots of the emulator testing.

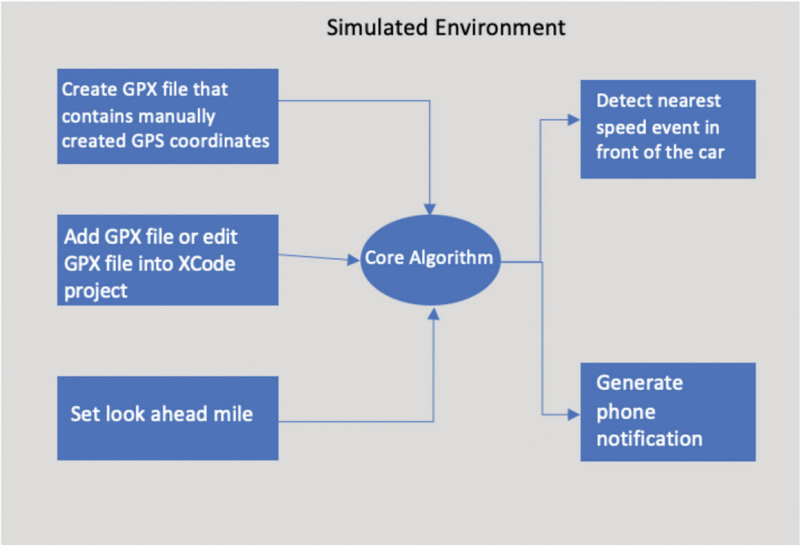


Figure 3.17 Project plan—simulated environment.

API, the output is a JSON file (see the sample shown in Figure 3.20). To get the JSON output file from the Purdue API server, we first declare parameters as a dictionary that contains string, doubles, and integers. Then create an URL Request with the URL to the Purdue API server. Set the HTTP request headers to include content-type as JSON and content-encoding as gzip. Afterward, create a data task to send the data to the server. Data received from the server will be decoded to contain specified data to do the delta-speed calculations. The significant properties that we used in the algorithm are summarized below.

- *Coordinates:* The coordinates contain latitude and longitude to specify the user location.

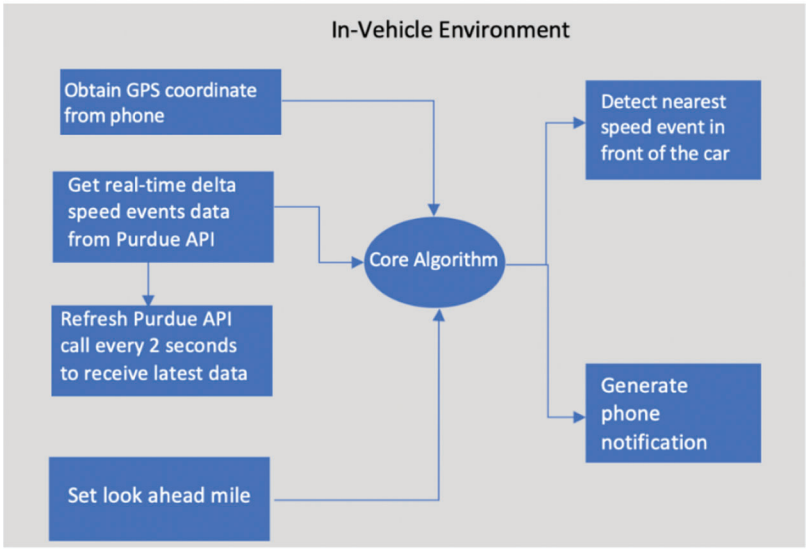


Figure 3.18 Project plan—in-vehicle environment.

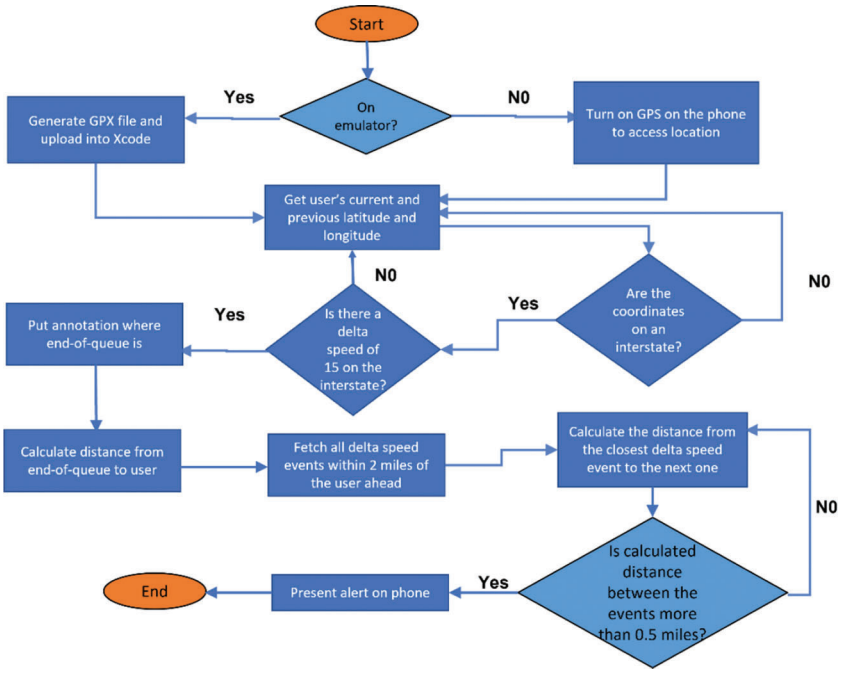


Figure 3.19 Flow chart of the algorithm.

- *Mile marker:* The mile marker of where the user is and where the event is.
- *Route heading:* This will state the direction that the user is heading.
- *Mile per hour:* States the speed to calculate the delta speed.

2. *Using GPS Coordinated for Location*
The main component of this application is the GPS coordinates. It is used throughout the application to capture the latitude and longitude of the user. As the user moves, the GPS coordinates will update. When the user first opens the app up, they will have the option to let the app use their location or not. If they choose no, then the app will direct the user to their settings to correct this. This pop-up will be present in both the simulation and on-road app testing. If the testing is taking place in the simulation, then a GPX file is needed. The GPX file will contain already preset coordinates of the user and their driving path. The GPX file determines the user direction, which results in the delta-speed events they may experience.
3. *User Coordinates*
The application updates the user location every 1 second. It gets the user's previous and current coordinates, which refresh throughout and are sent to the server every 2 seconds. The Purdue API will return a list of latitude and longitude points in front of the user's current location every time it is updated. The application will

```
{
    "waypoint_id": 152689,
    "waypoint_lat": 41.471257,
    "waypoint_lon": -87.327969,
    "waypoint_dist_to_point2": 94.47,
    "routename": "US-30 W",
    "mile_marker": 10.52,
    "veh_heading": 266,
    "speeds": [
        {
            "waypoint_id": 152689,
            "waypoint_lat": 41.471257,
            "waypoint_lon": -87.327969,
            "mm": 10.52,
            "route_heading": 266,
            "tstamp": "2021-06-24 00:16:10",
            "speed_mph": 33
        },
```


Figure 3.20 Sample JSON file.

continuously get the user's location if they are on the interstate or not. If the user is not on the interstate, the application will continue to make the coordinate updates and send them to the server.

4. *Determining Vehicle Direction*
   Once the current GPS is acquired, the user direction they are traveling will be determined. Using the multiple GPS coordinates of the user, it will determine the direction one is driving on the interstate. The Purdue API gets the user's previous and current coordinates. Which results in determining the direction that one is heading. Once this step is completed, it can be determined what potential delta-speed events are detected in front of the user.
5. *Calculating Delta-Speed Events*
   To calculate a delta-speed event a function was created. This function consisted of miles per hour 1 and 2, waypoint latitude and longitude 1, mile markers 0 and 1. Miles per hours 1 and 2 would take in the speed gathered from the API data results to subtract each one against one another to see if there was a delta-speed change of 15. If there was a delta-speed difference of 15 between two consecutive points, it would trigger the alert function. The alert function consists of an annotation showing the location of where the back-of-queue is. Followed by a local notification informing the user of information over the queue location relative to them.
6. *Calculating Distance*
   Once a delta-speed event is present, it would be helpful to know how far away it is from the user's current location. This is where the mile marker data is beneficial. Within the application, the absolute value of the users' location minus the nearest waypoint is calculated. That will result in the distance from the back-of-queue to the user location. Initially, there was no limitation on the rounding of the results, which was changed to have the distance round to the tenth.
7. *Issuing Alerts*
   Two different kinds of notifications can be given within this application, UIAlertController and local notification. A local notification is similar to that of push notifications but does not need a server or an apple entitlement to work. This kind of notification shows while the app is in the background and foreground. It is presented with an image on the right and the app's name and icon on the top of the alert. UIAlertController will show notifications from the bottom of the screen when the app is in the foreground. It will dismiss if the user presses okay or once the time delay runs out. The issue that occurred with this kind of notification was it would keep showing the same notification due to the number of times it had been called. Both of these alerts can be presented with similar properties such as sound, speech, and text. The local notifications are used in this application to show the alerts on the smartphone, which includes sound, text, and vibration features. An example of local notification, while the app is in the foreground, is shown in Figure 3.21.

3.4.4 Testing and Debugging

Throughout the app process, there were two ways of testing and debugging the application. Both of these methods helped to make the app improve.

1. *iOS 8 iPhone*
   Through using the smartphone, the application can be tested and debugged. The button and map functionality can be tested without the need for the tester to move. To test the other functionalities and algorithms of the app, it was needed to drive on the interstate. This application has been tested on different highways, and associated log files are created for debugging purposes. Throughout the process, the application was tested on Interstates 30, 65, 41, and 94.
2. *Simulate Location*
   A helpful tool that XCode provides is to simulate a location. There are preset ones already programmed in that

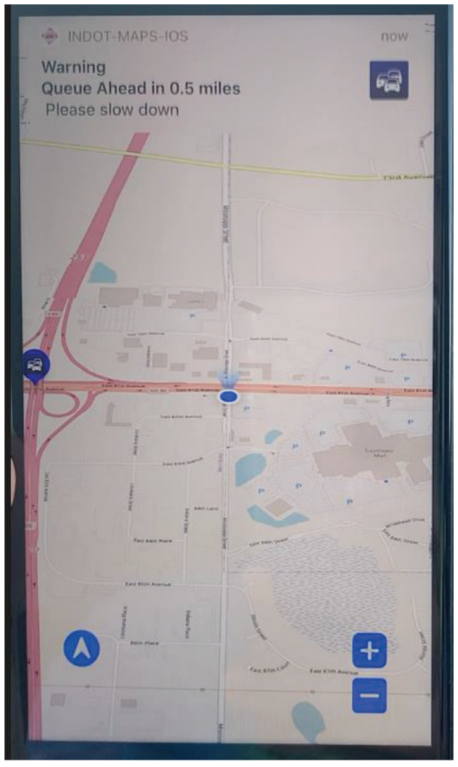


Figure 3.21 Notification on the iOS-based smartphone.

```
<?xml version="1.0"?>
<gpx version="1.1" creator="gpxgenerator.com">
<wpt lat="41.47126361044698" lon="-87.3250358591193">
    <ele>208.21</ele>
    <time>2021-03-02T22:10:43Z</time>
</wpt>
<wpt lat="41.47126361044698" lon="-87.3268383035773">
    <ele>208.43</ele>
    <time>2021-03-02T22:11:05Z</time>
</wpt>
<wpt lat="41.47121537650293" lon="-87.3290699014777">
    <ele>207.21</ele>
    <time>2021-03-02T22:11:32Z</time>
</wpt>
<wpt lat="41.47116714252294" lon="-87.33134441472232">
    <ele>204.40</ele>
    <time>2021-03-02T22:12:00Z</time>
</wpt>
<wpt lat="41.47110283049392" lon="-87.3340051660651">
    <ele>204.11</ele>
    <time>2021-03-02T22:12:33Z</time>
</wpt>
<wpt lat="41.47107067445545" lon="-87.33762584944279">
    <ele>206.50</ele>
    <time>2021-03-02T22:13:17Z</time>
</wpt>
<wpt lat="41.471054596430214" lon="-87.34131656904728">
    <ele>208.40</ele>
    <time>2021-03-02T22:14:02Z</time>
</wpt>
```

Figure 3.22 Example of a GPX file.

includes main cities that one could choose. Unfortunately, Indiana was not one of those pre-programmed locations. Hence a GPX file was created and imported into XCode. One can choose any road, spot, and speed within a GPX file. When one runs the app through XCode on their external phone, they will then click on the file that will simulate the location specified in the file. This can be useful to debug the program in a virtual environment with real-time data. Figure 3.22 shows an example of a GPX file setup.

3.4.3.1 CarPlay. The goal of using CarPlay is to issue alerts using a built-in vehicle hardware device. In this app, there are two ways the back-of-queue alert message is issued. The first one is to use a speech synthesizer to voice the alerts. The speech synthesizer tone, rate, pitch, and language can all be customized. Within this application, the speech rate was reduced. This app can run this feature without CarPlay. If CarPlay was implemented successfully, it would have been able to project this voice message to the user through the car speakers. Having the application be compatible with CarPlay could potentially reduce the distraction level to the driver.

For the app to be available on the CarPlay screen, one needs to apply for a CarPlay entitlement. Applying for the entitlement will let the application have access to all the capabilities that CarPlay has. The first step in this process is to have an apple developer account. While logged into the account, there will be a section to apply for the CarPlay entitlement. In this section, one will submit a short paragraph over the functionalities of the app, and then Apple will review the submission and give back feedback. The submission for the entitlement can be submitted more than once. For this project, we originally were denied the CarPlay entitlement based on the paragraph that we submitted. Certain CarPlay App Guidelines and requirements must be met, which can be found on the apple developer website.

After improvising the paragraph, we were then notified that to review the request, one needs to provide a link to the navigation app on the App Store and provide some information on the route guidance (turn-by-turn directions) and voice guidance features that are included in the navigation app. We then proceeded to implement route guidance and voice guidance with the app but realized that it took away from the main function, which is why we did not proceed further with the CarPlay navigation app.

There are short-term solutions around the CarPlay navigation app which is adding an entitlement file with com.apple.developer.playable-content as a Boolean in YES. This can be used in the development process but not in production for that will need Apple's entitlement granting.

Similar to that of Android Auto interaction with the app, the same output occurred with Carplay for iOS. The other short-term solution is jailbreaking the phone to see what it will look like on display. This method will display the app on CarPlay is through mirroring the phone and connecting it via a USB cord. Once the phone is jailbroken, then install Cydia and CarBridge. CarBridge takes any iPhone and can load it onto Apple CarPlay supported touchscreen head unit. Cydia allows users to install software that is not authorized by Apple onto jailbroken iPhones. After doing these steps, once connected to a touchscreen head-unit navigate to the CarBridge app, and once clicked on the screen should show in the unit. As seen in Figure 3.23, the screen in mirroring the iPhone. These two ways can be done in the meantime while Apple processes the CarPlay entitlement.

3.4.3.2 Important points. Important points when developing for CarPlay is that all voice interactions

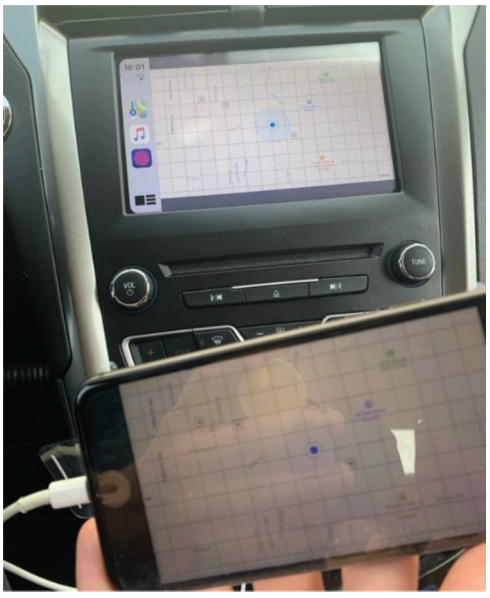


Figure 3.23 Mirroring iPhone onto the CarPlay unit.

must be handled using SiriKit. The iOS will manage the display of UI elements and handles the interface with the car. The application does not need to manage the layout of UI elements for different screen resolutions or support different hardware. One of the requirements for the navigation apps along with the route and route guidance is that the app needs to provide a way to enter panning mode. Some of the other requirements are not to instruct the user to perform tasks on iPhone. The CarPlay app icon must look similar to the app icon displayed on iPhone. The app is responsible for providing icons and image assets for CarPlay, including an app icon, navigation and map button icons, maneuver symbols, alert icons, and other UI elements.

3.4.3.3 Summary. In the previous section, the software development and hardware preparation were summarized for the back-of-queue alerting system. Functions were developed to be able to successfully talk to the Purdue API server and get data back. Algorithms were developed to calculate and issue back-of-queue alerts. Apple CarPlay and an iOS phone were looked into to be compatible with this application. Two methods for presenting the alerts were developed and tested. Along with the steps needed to be successful in obtaining a CarPlay entitlement and ways around it.

4. UPDATING OF DASHBOARDS

Over the past decade, a number of dashboards have been developed and are currently in use by operators at the TMC. As part of a deliverable of this project, existing TMC dashboards have been updated with the smart segment geodatabase to incorporate new and emerging data sources for greater data-actionable, targeted alerting and messaging to operators. In addition, three new dashboards and one new backend service for enabling a third-party interface have been developed that make use of new mapping schemes. Based on an inventory of all existing dashboards, development progress and assessment of usage, some were updated, sunset or replaced with new dashboards.

4.1 Inventory of Dashboards Prior to Project

In total, 14 dashboards developed from previous projects were inventoried (see Table 4.1). The list of categories and enumerated dashboard names are tabulated below.

Profile-Type Dashboards

1. Delta Speed Profile–ounts occurrences of speed differential events along miles of interstate and sums the amount over a time period.
2. Congestion Profile–counts hours of congestion along miles of interstate over a time period.
3. Speed Profile–counts the number of hours using different colors for each speed category, along miles of interstate over a time period.

Time-Space Heat Maps

4. Queueing Heat Map–time-space map of congestion using shading to indicate intensity.
5. Queue MOT–work zone milepost-based time-space map that indicates locations of speed differential and queue durations, with integrated ITS camera snapshots.

Tickers

6. Traffic Ticker–summary dashboard that shows total miles of interstate congestion over time, filterable by district and route.
7. Weather Ticker–summary dashboard of interstate congestion over time, filterable by snow route and district, with precipitation and solar flux data.

Geographic Maps

8. Delta Speed Map–segment-based dashboard that colorizes based on speeds on a map, with tabulated locations of interstate slowdowns.
9. Segment Ranking–ranked list of most improved and degraded segments statewide on a map interface.
10. Segment Travel Time–dashboard that contains cumulative distribution frequency diagrams that compares travel times of corridors from different time periods, by time-of-day and direction.

Year-Over-Year Charts

11. Congestion Hours–dashboard that plots interstate congestion-hours for multiple years, broken-out by month.
12. Distance-Weighted Congestion Hours–dashboard that plots interstate distance-weighted congestion-hours for multiple years, broken-out by month.
13. Delay45–dashboard that plots interstate delays below 45 mph for multiple years, broken-out by month.
14. Total Delay–dashboard that plots interstate total vehicle-hours of delay for multiple years, broken-out by month.

The inventory of dashboards developed prior to this project and their statuses are listed below. Each dashboard is listed as "new," "updated," or "sunset." "New" dashboards are applications that have been developed to replace or succeed the existing dashboard, integrating some or all of the old features with the addition of enhancements. "Updated" dashboards use the pre-existing user interface with updated smart segment data interfaces that generates the speeds information with the new geodatabase latitude and longitude points. "Sunset" dashboards are applications retired due to features having been migrated to other applications or lack-of-use and are no longer maintained.

4.2 New Dashboards

Three new dashboards were developed using the smart segment base map, using the Google Maps API, and include a combination of features from previous dashboards. The dashboards include new feature enhancements, metrics, and better user experience. Figure 4.1 shows the new splash page of the updated and new dashboards developed as part of the deliverables of this project. The site is accessible at https://tmc-dashboards.uc.r.appspot.com/.

TABLE 4.1
Traffic dashboards update status

Category	Old Dashboard Name	Status	Newly Featured In
Profiles	Delta Speed Profile	New	Heatmap
	Congestion Profile	Sunset	–
	Speed Profile	Updated	–
Heatmaps	Queueing Heat Map	New	Heatmap
	Queue MOT	New	Heatmap
Tickers	Traffic Ticker	Updated/New	Deltaspeed Version 2
	Weather Ticker	New	Heatmap
Geographic Maps	Segment Ranking	Sunset	Performance Measure Dashboard
	Segment Travel Time	New	Trajectory Heatmap
	Delta Speed Map	New	Delta Speed Version 2
Year-Over-Year	Congestion Hours	Sunset	–
	Distance-Weighted Congestion Hours	Sunset	–
	Delay45	Sunset	–
	Total Delay	Sunset	–

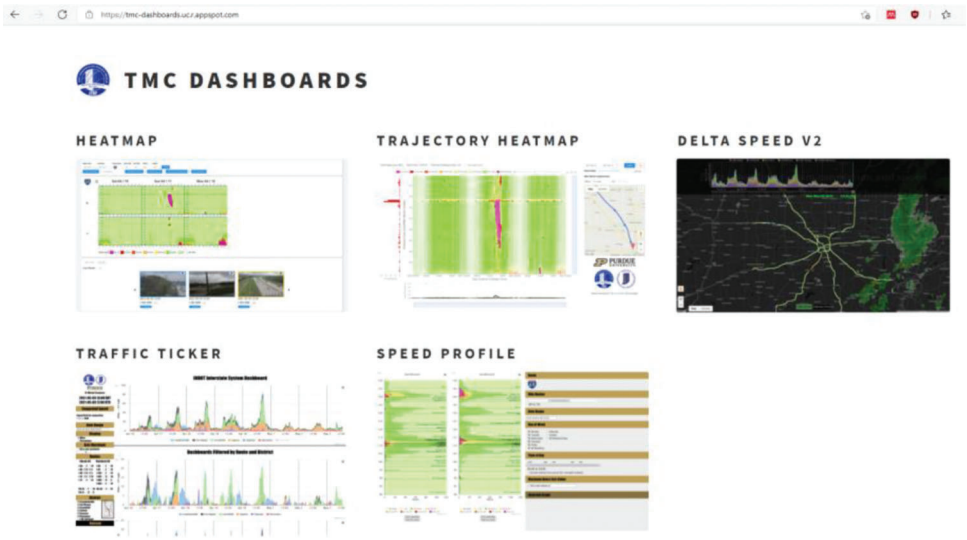


Figure 4.1 Splash page of new dashboards.

Additionally, a back-end application enabled Hoosier Helper position data to be retrieved and uploaded in real-time to a third-party emergency response vehicle tracking application. The application used the smart segment geodatabase to detect slowed emergency vehicles to trigger an event, and the vehicles and locations of the events on a real-time dashboard map. The application had allowed stakeholders to send emergency-response locations directly to Waze.

4.2.1 Heatmap

The heatmap application is a time-space-oriented dashboard that plots speed over miles of interstate across hours of the day using the smart segment geodatabase. The median speeds of each segment incremented at 5-minute intervals are colorized in the following seven categories: (1) 0 to 14 mph (pink); (2) 15 to 24 mph (red); (3) 25 to 34 mph (dark orange); (4) 35 to 44 mph (light orange); (5) 55 to 64 mph (dark green); and (6) 65 mph or greater (light green); and (7) no data (gray).

The user can select a route, milepost, and date range over one or more days to plot. Additional data integrated with the smart segment geodatabase and heatmap dashboard include the following: (1) images scraped from over 300 traffic cameras from the ITS camera network is provided at 3-minute intervals and can be viewed by selecting mileposts on the route; (2) NOAA High-Resolution Rapid-Refresh temperature and precipitation rate data at 1-hour, 3-kilometer intervals; (3) AVL GPS point and alert status data from emergency response vehicles; (4) statewide crash data; (5) crowdsourced CV position and speed data; (6) crowdsourced CV hard braking data; and (7) crowdsourced CV windshield wiper data.

The heatmap replaces the Delta Speed Profile, queuing heat map, queue MOT, and weather ticker. A view of the dashboard is shown in Figure 4.2.

4.2.2 Delta Speed Version 2

The Delta Speed version 2 application is an integrated map-based dashboard developed using the smart segment geodatabase map that shows interstate traffic speeds at 0.1-mile resolution, and Hoosier Helper (red circles), Haas Alert vehicle (orange circles), and snow truck (blue circle) locations from AVL sources. The sum of miles of interstate congestion statewide, colorized by INDOT district, is plotted in an area chart at the top of the dashboard. The user can access historical 7-days' data via a time slider.

Images from a vehicle's dash cam can be toggled by clicking on the circle position of the vehicle, where available, that shows additional information such as the vehicle name, dispatch garage, time stamp of reading, speed, and heading. Additionally, doppler data overlay is added to the map and colorized based on precipitation intensity and type and is accessible for 60-days. A screen shot of the dashboard during a summer thunderstorm in the Indianapolis area is shown in Figure 4.3.

During winter operations, locations of snow plows are tracked and plotted using the snow routes feature that interpolates and connects likely paths between 60-second pings of vehicles using the Google Maps Directions API. Additionally, the paths are colored from light blue to dark blue and purple based on the salt

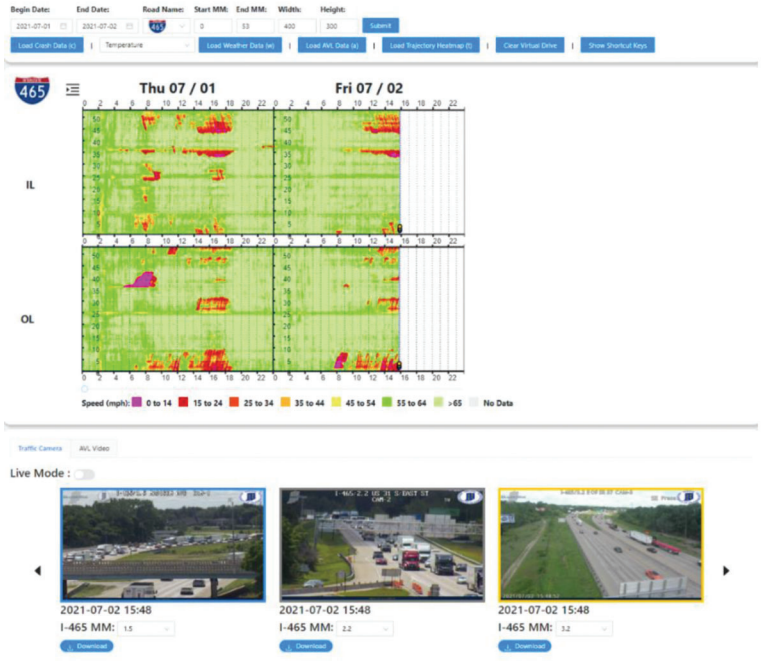


Figure 4.2 Heatmap dashboard.

Figure 4.3 Delta Speed version 2 dashboard during summer season.

application rate data from the vehicle. An example of the dashboard view in the winter is show in Figure 4.4.

4.2.3 Trajectory Heatmap

The Trajectory heatmap dashboard shows miles of interstate speeds over time using CV trajectory (GPS latitude, longitude, and time stamp) data for individual vehicle trips (see Figure 4.5). The trajectories are colorized by speed using the same color scale as heatmap. The user interface allows routes to be defined on-the-fly within the State of Indiana. Additionally, hard braking data plotted over the length of the defined route (left-vertical axis) sums the total number of hard braking events at each section of roadway over the time period. Travel time is plotted on the bottom horizontal axis as a scatter plot or 10-point moving average at each instance within the date range.

4.2.4 Haas Alert

A new backend service queries Hoosier Helper location data from a database in the TMC, and based on proximity to intestate routes, heading and speed, triggers an event to the Haas Alert API. The service uses the smart segment geodatabase to determine, if a Hoosier Helper vehicle is travelling within 250 feet of the route, below 2 mph and, in the direction, or exact opposite direction of travel (within 10 degrees tolerance), whether the vehicle is on the route and potentially assisting on the side of the roadway. If a vehicle does not match the speed, heading, or proximity criteria, the position of the vehicle is sent to the Haas Alert API without an alert. The service updates any available Hoosier Helper positions every 30 seconds.

4.3 Updated Dashboards

4.3.1 Speed Profile

The speed profile dashboard has been updated to use the smart segment geodatabase at 0.1-mile resolution. Using historic INRIX speed data from 2014 to 2019, the dashboard's segments have been repopulated and mapped to the updated segments for displaying sum of hours at which speeds a segment is operating over a time period.

4.3.2 Traffic Ticker

The Traffic Ticker dashboard has been updated to use the smart segment geodatabase at 0.1-mile

Figure 4.4 Delta Speed version 2 dashboard during winter operations.

Figure 4.5 Trajectory heatmap dashboard.

resolution. Using historic INRIX speed data from 2014 to 2019, the dashboard's segments have been repopulated and mapped to the updated segments for displaying the total number of hours of congestion over time plotted on an area chart by district or route.

4.4 Sunset Dashboards

4.4.1 Congestion Profile

The Congestion Profile dashboard has been sunset due to lack of use and will no longer be maintained. The speed profile has become a more popular and effective version to the application as it sums up the number of hours of when a section of roadway is operating based on speed rather than day-of-week or time period.

4.4.2 Segment Ranking

The Segment Ranking dashboard has been sunset and replaced by the Traffic Signal Performance Measure dashboard (Saldivar-Carranza et al., 2021). The new dashboard leverages CV data and intersection approach-based mapping. It includes features such as displaying on a gridded heatmap the arrivals-on-green, split failure, level of service, and downstream blockage at one or more intersections in the state.

4.4.3 Year-Over-Year

The following four year-over-year dashboards have been sunset due to lack of use and will no longer be maintained: (1) congestion hours, (2) distance-weighted congestion hours, (3) delay45, and (4) total delay. the dashboards have been replaced by historic review features in the updated speed profile, heatmap, trajectory heatmap, and Delta Speed version 2.

5. DEVELOPMENT AND EVALUATION OF USE CASES

5.1 Driving Simulator Study

To evaluate the effectiveness of the proposed back-of-queue alerting system, a driving simulator-based study was conducted and introduced by using a high-fidelity driving simulator at IUPUI and measuring driving speed changes, driver response time, and other driving safety measures. This task will mainly include the following sections: test scenarios design, data collection apparatus, participants and procedures, data pre-processing, and metrics selection.

5.1.1 Data Collection Apparatus

The obtained data from this study are mainly from two sources. One is a driving simulator that collects the vehicle dynamics data such as driving position, driving speed, and driving acceleration. The other source is the Driver State Sensing (DSS) which acquires the subjects' drowsiness data. Figure 5.1 demonstrates the DriveSafety DS-600c high-fidelity driving simulator in the Transportation and Autonomous Systems Institute (TASI) at Indiana University-Purdue University Indianapolis (IUPUI).

The driving simulator has rear-view and side-view mirrors and can accomplish longitudinal motion cues (5 inches) and variable pitch angle (+/-2.5 degrees). A partial vehicle cabin is instrumented with standard driver controls, full-width front interior, position/time triggering, and active instrumentation. The driving simulator is also available to set/assign different signs/visual signals at different roadside locations (Shen et al., 2021; Zhang et al., 2021). For the roadside BoQ alert, as can be seen in Figure 5.2, a roadside traffic sign with the text "Be Prepared to Stop" is designed in the driving simulator to mimic the actual roadside mobile signs implemented by INDOT, where a large message board is carried by a roadside assistance truck. The detailed introductions about the driving simulator can be seen in (Li et al., 2019). For evaluating the alerting system, an experimental platform was designed and implemented. The goal of designing this platform is to simulate a driving environment where the driver would be notified through the mobile application of the upcoming traffic hazards. The detailed design of the staged web server between the driving simulator and the mobile application can also be referenced (Li et al., 2019).

5.1.2 Driving Scenario Design

Three components are integrated into the driving simulator, namely, HyperDrive, Vection, and Dashboard, respectively. HyperDrive is the software for system preparation and scenario creation. Vection is the software for simulating creating test scenarios. The Dashboard is the tool that interfaces with the HyperDrive and Vection. The overall structure of the driving simulator can be found in (Li et al., 2019). To simulate the highway BoQ alerting system with real driving conditions on Interstate highway I-465, several test scenarios were created in the driving simulator, where we can build the test environment with roadway networks and controlled entities. Entities were also used to determine objects in the environment, such as vehicles or road signs. As described in (Li et al., 2019), the simulated I-465 is a ring road of Indianapolis with a length of around 52.79 miles (84.46 km). The test environment of both the internal view and external view of DS-600 is demonstrated in Figure 5.3. Furthermore, several key variables that are collected after every simulation are depicted in Table 5.1.

To eliminate the learning effects from the subjects and the predictability of the future scenarios, three different road segments were designed and assigned from the overall I-465 route to subjects randomly with the lengths of 5 miles, 6 miles, and 7 miles, respectively. To make sure that all test scenarios are similar besides several controlled variables (alert types, driver states,

Figure 5.1 The DriveSafety high-fidelity driving simulator.

Figure 5.2 Roadside traffic sign alert in the driving simulator.

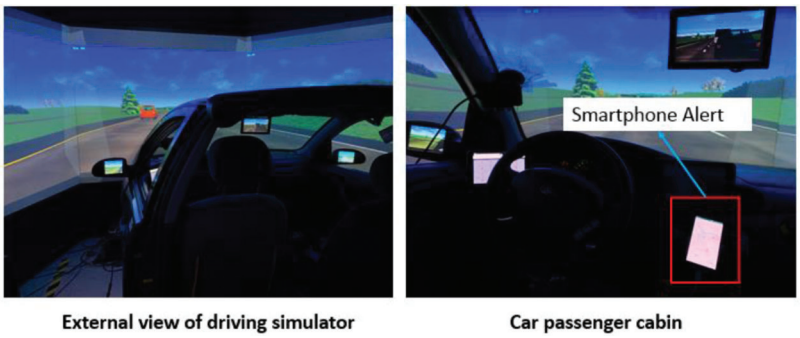


Figure 5.3 Actual test conditions using the driving simulator.

and weather conditions), these test road segments were established based on the following rules.

1. All road segments are on the highway suburban areas.
2. There are three lanes along the driving direction and three lanes on the opposite way.
3. The curves included in the simulation are relatively large ($\geq$ 1,000 meters).
4. Subjects have a good view range and enough time when approaching the queue at the end of each test.
5. All queues are static at the end of each scenario.
6. The visibility in the simulated foggy condition is the same in three different road segments.

Meanwhile, both roadside sign alert and the in-vehicle auditory alert are placed and triggered at about 1 mile upstream of the BoQ. The in-vehicle auditory alert is delivered through the smartphone installed on the cluster of the driving simulator vehicle cabin (shown on the right of Figure 5.3). Note that subjects may get increasingly familiar with the scenarios and slow down the vehicle when encountering slower traffic. To avoid such learning effects, we placed one or two waves of the slow traffic (not slow enough to be at the BoQ) randomly during each test scenario. Thus, it is much more difficult for the subjects to estimate the final traffic queue positions. To better demonstrate and summarize the test scenarios during experiments, some key designs were shown in Figure 5.4.

TABLE 5.1
Key variables collected after simulation

Key Variable Name	Unit
Longitudinal Acceleration	m/s^2
Crash or Not	0 or 1(binary)
Crash Speed	m/s
Brake Pressure	0%–100%
Steering Angle	degree
Vehicle Location	m

5.1.3 Participants and Procedures

There is a total of 40 (24 male, 16 female) recruited subjects. During the experiments, the subjects need to drive in three different states, as described earlier in the driving scenario design. The detailed descriptions of the driver states are as follows (Shen et al., 2021; Zhang et al., 2021).

1. *Normal:* During this test, subjects are asked to drive in the simulator for about 10 miles on a replica of highway I-465 with moderate traffic. At the end of the 10-mile drive, a traffic queue will appear. Drivers will not have any impaired driving states during the tests. This test takes about 10 minutes.
2. *Distracted:* During this test, subjects are also asked to drive on the simulator for about 5–7 miles on a replica of highway I-465 with moderate traffic. At the end of the scenario, a traffic queue will appear. We used the N-back task to control the cognitive workload of subjects (Kirchner et al., 1958). To perform the N-back task, a sequence of random numbers is broadcast to the drivers at a certain speed, and the drivers are required to instantly recall and repeat the number "N" places before the current number. In this study, the subject will engage with a 1-back task (where the drivers need to repeat the number one place before the current number) since the random location of driving from the beginning until the end of the test.
3. *Drowsy:* During this test, subjects are asked to drive for about 20 miles on a replica of highway I-465 with moderate traffic. The subject should drive as they normally would, but they should be drowsy when they perform this test based on the KSS feedbacks from subjects every 5–6 mins. To help ensure drowsiness, this test is the last environment tested in both categories. Additionally, the time this test is performed is planned around when subjects will naturally get tired (i.e., in the afternoon after lunchtime). This test takes about 20 minutes.

5.1.4 Data Pre-Processing

To take advantage of the data from both the driving simulator and the DSS, the combined data need to be

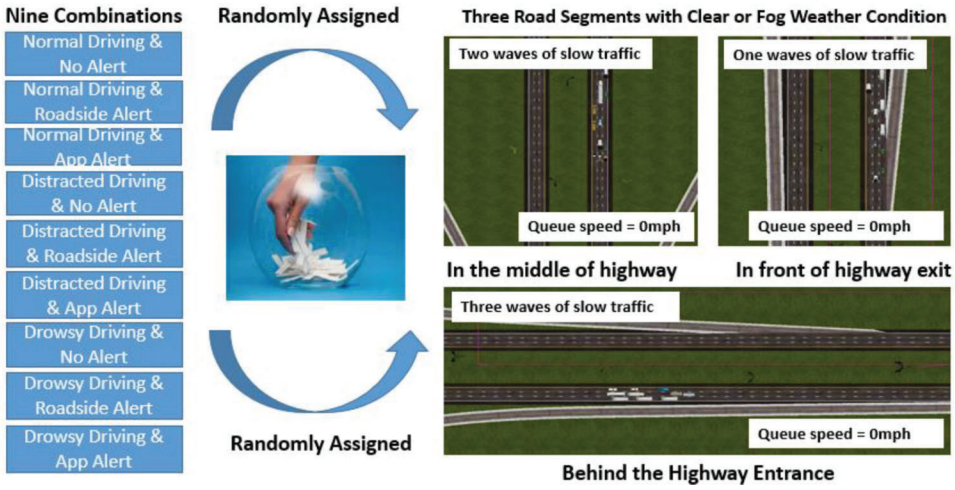


Figure 5.4 Some key scenario designs in the driving simulator (Zhang et al., 2021)

preprocessed before being analyzed. Since two sets of data were captured from different computers, the most important step of data pre-processing is to synchronize them. The need for time synchronization arises from the need to process sequential data collected from different sensors at different sampling rates. To match the driving simulator data to the DSS data, each frame of the DSS data was time-stamped at the rate of 40, and the UNIX timestamp was recorded as part of the saved data. Similarly, each frame of driving simulator data was also timestamped with a rate of 60. We developed software programs to synchronize the data such that the list of the indices in the driving simulator data can be generated and used for extracting the synchronized data with the DSS data. Finally, the pre-processed data has nine key variables for each test scenario, including time, left eye open status, right eye open status, velocity, steering wheel angle, acceleration, braking ratio, position X, and position Y (Zhang et al., 2021).

5.1.5 Metric Selection

We selected three widely-used metrics as the indicator of safety, which are the minimum time-to-collision (mTTC), maximum braking (mBraking), and maximum steering angle (mSteer). mTTC measures the time to collision under the current speed, and either a high speed of the vehicle or a short distance away from the incident might cause a small mTTC, which is considered dangerous. Maximum braking is very similar to maximum deceleration, which causes driving discomfort for the driver and may lead to safety concerns on surrounding vehicles. The maximum steering angle is an indicator of vehicle stabilization. In addition to the above-mentioned metrics, we also considered the remaining distance (RD) to the traffic queues when mTTC is triggered. Therefore, we have a remaining distance when the driver attains the maximum time-to-collision. A Larger RD indicates an earlier engagement to prevent the crash. Note the mTTC and mBraking are calculated for the last 1 mile of the speed profile (after the alert is issued). The mSteering is calculated for the last half mile where the highway is straight (Zhang et al., 2021).

5.1.6 Summary

In this section, we mainly implemented an in-vehicle highway BoQ alerting system based on an Android-based smartphone app. Then a mixed design (containing both between-subjects and within-subjects designs) to examine the effects of driver states, type of alerts, and visibility conditions were devised for assessing the effectiveness of the proposed alerting system. The experiment design and hypotheses were illustrated. The background and the functionality of the driving simulator are also demonstrated along with its embedded software and hardware. Several test scenarios designed in DS-600 were also demonstrated. Forty subjects were recruited to perform driving tests on the driving simulator as well as the testing procedure. The required data has been saved and pre-processed for future analysis for proving the proposed hypotheses. Three widely-used metrics as the indicator of safety were selected to evaluate the system performance.

6. TESTING AND VERIFICATION OF MESSAGE DELIVERY SYSTEM

6.1 HMI Evaluation—Preliminary Evaluation

6.1.1 Design of Experiment

The goal of this section is to examine the effectiveness of different types of alerts (no alert, roadside traffic sign alert, and in-vehicle auditory alert) in the BoQ situation while the driver is in different conditions (normal or distracted). Based on the current development of the BoQ alerting system, we have developed six test scenarios via a full-factorial design to evaluate the effectiveness of the traffic queue warning functionality. The controlled variables are shown in Table 6.1. The six test scenarios are categorized as no alert, roadside traffic sign alert, and in-vehicle auditory alert. Under each warning type, there are two driving states: normal and distraction. In this experiment design, the no alert warning type will act as the baseline to be compared with the other two warning categories. We applied a within-subject design where each subject completes all test scenarios. The order of six test scenarios was randomly shuffled for each subject to reduce the learning effects.

Both in-vehicle auditory alert and roadside traffic sign alert are triggered or placed 1 mile upstream of the final traffic queue. Similarly, to ensure that subjects could observe the roadside traffic sign for approximately 20 seconds at a speed of 65 mph, there is no curved road designed before where the sign is placed. The in-vehicle auditory alert is delivered through the smartphone installed on the cluster of the driving simulator vehicle cabin. The content of the auditory alert is "slow traffic ahead. Please slow down." with a young female voice.

This experiment mainly focuses on the following four hypotheses.

- Hypothesis 1: Roadside BoQ alert and in-vehicle BoQ alert can improve driving performance compared to no alert.
- Hypothesis 2: In-vehicle alert is more efficient than roadside alert to improve driving safety.
- Hypothesis 3: Distracted driving can result in more risks when approaching the BoQ during highway cruising.

TABLE 6.1
Experimental design

Factor	Level		
Driver State	Normal	Distracted	
Type of Alert	No Alert	Roadside Sign	In-cabin Alert

- Hypothesis 4: Driver distraction can affect the efficiency of different BoQ alerts compared to the normal driving situation.

6.1.2 Data Collection

A total of ten participants aged 22–34 (mean +/- SD = 29.2 +/- 4.61) were recruited in this testing. All participants possessed a valid driving license and have driving experiences in the U.S. The data collection was approved by the Institutional Review Board (IRB) of Indiana University.

Firstly, the subjects made an appointment with the researcher through a phone call. The subjects are required to maintain a sober state on the experiment day. After they came to the lab, they were verbally informed of the purpose and procedures of the experiment. They would need to read and sign a consent form before starting the experiment. Then the subjects are asked to perform a 5-minute and a 2-minute practice on the normal driving task and 1-back distracted task, respectively. They can ask for more practice if they need it.

The subjects started the experiments on the driving simulator after all the training sessions. There are a total of six scenarios, each of which took an average of 6 minutes to complete. There is a 1-minute break after each trial, and the subjects can require more time to rest if they feel tired or have motion sickness. The researchers will terminate the experiment if the subjects exhibit severe motion sickness symptoms. The whole experiment took about 60 minutes to complete.

The subjects need to drive in two different states during the experiments, as described earlier in the experiment design. The detailed descriptions of the driver states are the following.

- *Normal:* During this test, subjects are asked to drive on the simulator for about 5–7 miles on a replica of highway I-465 with moderate traffic. At the end of the scenario, a traffic queue will appear. Drivers will not have any impaired driving states during the tests.
- *Distracted:* During this test, subjects are also asked to drive on the simulator for about 5–7 miles on a replica of highway I-465 with moderate traffic. At the end of the scenario, a traffic queue will appear. We used the N-back task to control the cognitive workload of subjects (Kirchner, 1958). For the N-back task, a sequence of random numbers is broadcast to the drivers at a certain speed, and the drivers are required to instantly recall and repeat the number "N" places before the current number. In this testing, the subject will engage with a 1-back task (where the drivers need to repeat the number one place before the current number) for 45-second of driving from the beginning until the end of the test. A smartphone is placed behind the driving simulator playing a pre-recorded series of single-digit numbers at 1/3 Hz (20 numbers per minute). The difficulty of the 1-back task is relatively easy, and we observed an averagely of 95% accuracy during the tests among the subjects when repeating the required numbers. This 1-back task aims to mimic certain distracting behaviors during driving (e.g., talking on the phone and talking with passengers).

All subjects are instructed to drive on the driving simulator around 65 mph if applicable. Subjects must follow all driving laws and rules observed in the real world. Lane changing is not allowed in this experiment, and all subjects are expected to drive in the middle lane. They need to slow down or completely stop when encountering slow traffic or traffic queues.

6.1.3 Metrics

To better evaluate the usefulness of different warning types in improving driving performance, it is important to study both the driver inputs (brakes and accelerations) and vehicle dynamics (vehicle lateral acceleration) when approaching the back of traffic queues. A list of measures needs to be compared between the scenarios when the alert is delivered or not for normal and distracted drivers. This study focuses on three main metrics: the minimum time-to-collision (TTC), the maximum deceleration, and the maximum lateral acceleration for all test scenarios.

For the first measurement, the TTC is defined as the time that remains until a collision between two vehicles would have occurred if the collision course and velocity difference are maintained (Hayward, 1972). TTC is a significant quantity for measuring driving safety. The larger the minimum TTC is, the better the driving safety is. The second and third measurements are variables related to vehicle dynamics and are very important for capturing evasive driving behaviors when approaching the BoQ. They can be obtained directly from the driving simulator. The smaller values of these two parameters indicate better and safer driving performance.

6.1.4 Data Analysis Method

A total of 60 tests are completed and recorded in this study (6 scenarios for each of the ten subjects). For each of the tests, the minimum TTC and the maximum deceleration are calculated for the last mile of the speed profile (after the alert is provided). In addition, maximum lateral acceleration is calculated for the last 0.4 miles (where the driving highway is straight).

We use OLS regression, two-way ANOVA, and Tukey's HSD test to analyze the data concerning each parameter. Note that we do not add the interaction terms into the regression model, for which we study in the ANOVA. For Tukey's HSD test, we set the significant level to 0.05 due to the limited sample size. The statistical models are used to examine the effects of driver states, types of alerts, and their interactions, on driving performances measured by the three metrics. Standard assumptions for the models are used and checked.

6.1.5 Results

6.1.5.1 Effects of BoG alerts on the driving performance. We first studied the effects of two types of BoQ alerts on driving performance. For each alert

type (no alert, roadside alert, and in-vehicle alert), there are 20 tests completed in two different driving states. At this stage of the analysis, the data from the two driver states (normal and distracted) are combined to study the overall alerting effects in real conditions, where some drivers may get distracted while others are not.

In the model where mTTC is the dependent variable, we find that the effect of "alert types" ($F = 4.32$, p-val $= 0.02$) are significant from the ANOVA. The average mTTC and standard errors for different types of alerts is shown in Figure 6.1(a). The regression model (R-squared = 0.13) also indicates that the coefficients for "no warning" (coef = -1.24, t = -2.70, p-val = 0.009) are significant while "in-vehicle auditory alert" is merged in the intercept (coef = 5.40, t = 14.42, $P < 0.001$). Tukey's HSD tests suggest that mTTC with "in-vehicle auditory alert" is higher than mTTC with "no warning." However, it fails to reject the hypothesizes that mTTC with "roadside traffic sign" is different from it with "no warning" or "in-vehicle auditory alert." This indicates "in-vehicle auditory alert" can significantly increase mTTC so that making the driving performance safer. Considering the relatively small sample size, we think this difference is pretty prominent to be detected. The data (Figure 6.1(a)) also show a clear trend that "roadside traffic sign" can increase mTTC to make the BoQ situation safer, which may not be as prominent as the "in-vehicle auditory alert" thus will need a larger sample size to detect.

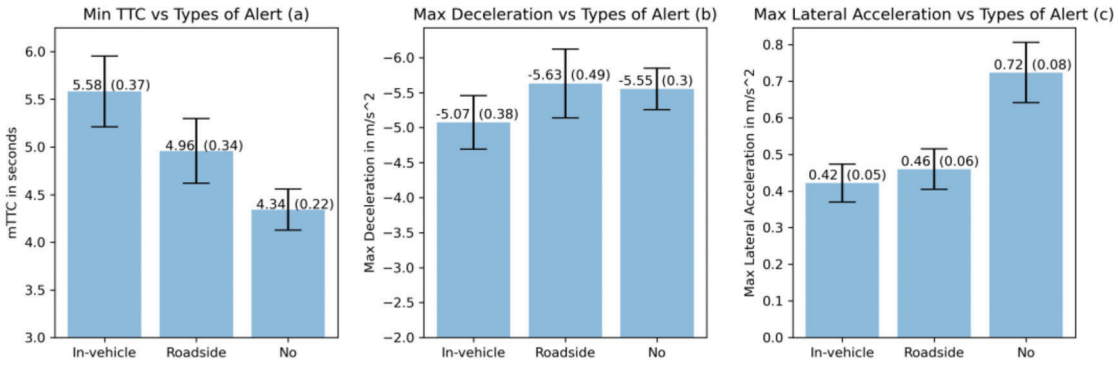
In the model where the maximum deceleration is the dependent variable, no significant results of types of alerts are found from the ANOVA and regression model, which suggest that the effect of alert types is not statistically significant. The average maximum deceleration with respect to different alert types is shown in Figure 6.1(b). We can notice that the mean max deceleration of different alert types is close to each other (with overlapping confidence intervals). These results suggest that BoQ alerts do not show significant effects in reducing the maximum deceleration in general.

In the model where the maximum lateral acceleration is the dependent variable, the results are similar to the results of the mTTC analysis. The effect of types of alerts ($F = 6.00$, p-val $= 0.004$) is significant from the ANOVA. The only significant coefficient (except for the intercept) in the regression model is for "no warning" (coef = 0.30, t = 3.20, p-val = 0.002) and "in-vehicle auditory alert" is included in the intercept (coef = 0.45, t = 5.79, p-val < 0.001). Tukey's HSD tests infer that the maximum lateral acceleration under the "no warning" condition is higher than the other two conditions. The average maximum lateral acceleration for different types of alerts is shown in Figure 6.1(c). These results show that both BoQ alerts can smooth the driving by reducing maximum lateral accelerations similarly.

To summarize all the analysis results in this stage, we can conclude that the first two hypotheses are approved. In addition, the results show that roadside BoQ alert and in-vehicle BoQ alert can increase mTTC and reduce the maximum lateral acceleration when approaching the BoQ. Also, mTTC analysis results suggest that in-vehicle alerts are more efficient than roadside alerts to improve driving safety further.

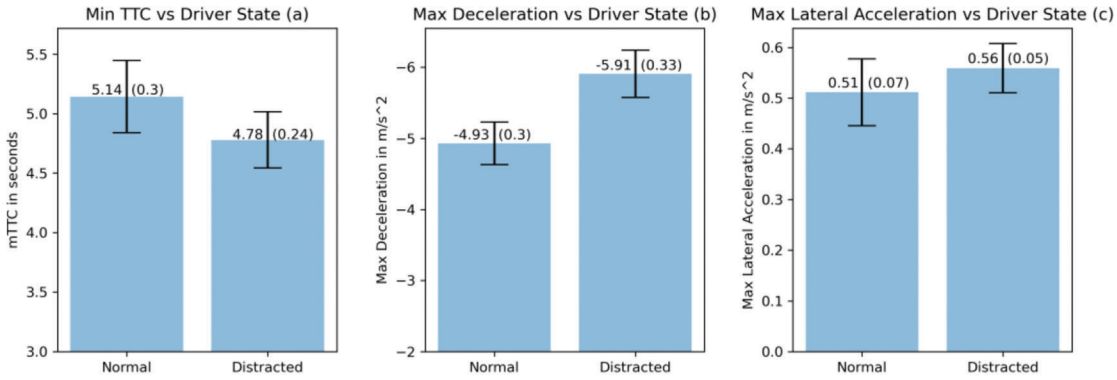
6.1.5.2 Effects of driver distraction on the driving performance. In the second part of the data analysis, we focus on the effects of driver distraction on driving performance, combining the different alert types. This analysis tries to understand when different alerts (or no alert) are provided when approaching the BoQ, how normal and distracted drivers will behave in terms of the three measurements. The data were reorganized so that each driver state will have 30 tests (10 from each alert type) for the following analyses.

We conducted a similar analysis process as the previous section for studying the effects of alert types. In the model where mTTC is the dependent variable, no significant results of driver states are found from any of the statistical tests. Figure 6.2(a) shows the average mTTC under different driver states. The two mTTC average values are close to each other with overlapped confidence intervals. This indicates that, in general, drivers with different states will have similar mTTC, when combining the three alert types for the analysis.



Note: The number in the parenthesis is the standard error of the mean.

Figure 6.1 Average mTTC (left), average maximum deceleration (middle), and maximum lateral acceleration (right) for different types of alerts.



Note: The number in the parenthesis is the standard error of the mean.

Figure 6.2 Average mTTC (left), average maximum deceleration (middle), and maximum lateral acceleration (right) for different driver states.

The ANOVA concludes that the driver states ($F = 4.66$, p-val = 0.04) is a significant factor for the maximal deceleration. The coefficient of the "normal state" (coef = 0.98, t = 2.14, p-val = 0.037) is significant in the regression model (R-squared = 0.09) where the "distracted state" is merged in the intercept (coef = -5.56, t = -12.14, p-val < 0.001). Also, we conclude that the maximum deceleration with "distracted state" is higher than it with "normal state" according to Tukey's HSD test. The average maximum deceleration of different driver states (as shown in Figure 6.2(b)) is considerably different, consistent with Tukey's HSD test results. This result approves that distracted drivers tend to have more emergent brakes across different alert types when approaching the BoQ during highway driving.

In the model where the maximum lateral acceleration is the dependent variable, the results again show that the effect of driver state is not a significant factor from all tests. The average maximum lateral acceleration for different driver states is shown in Figure 6.2(c). It is noticeable that the two mean values are close to each other with overlapped confidence intervals, which also confirm that drivers with different states have similar maximum lateral acceleration values in our experiments across all three alert types.

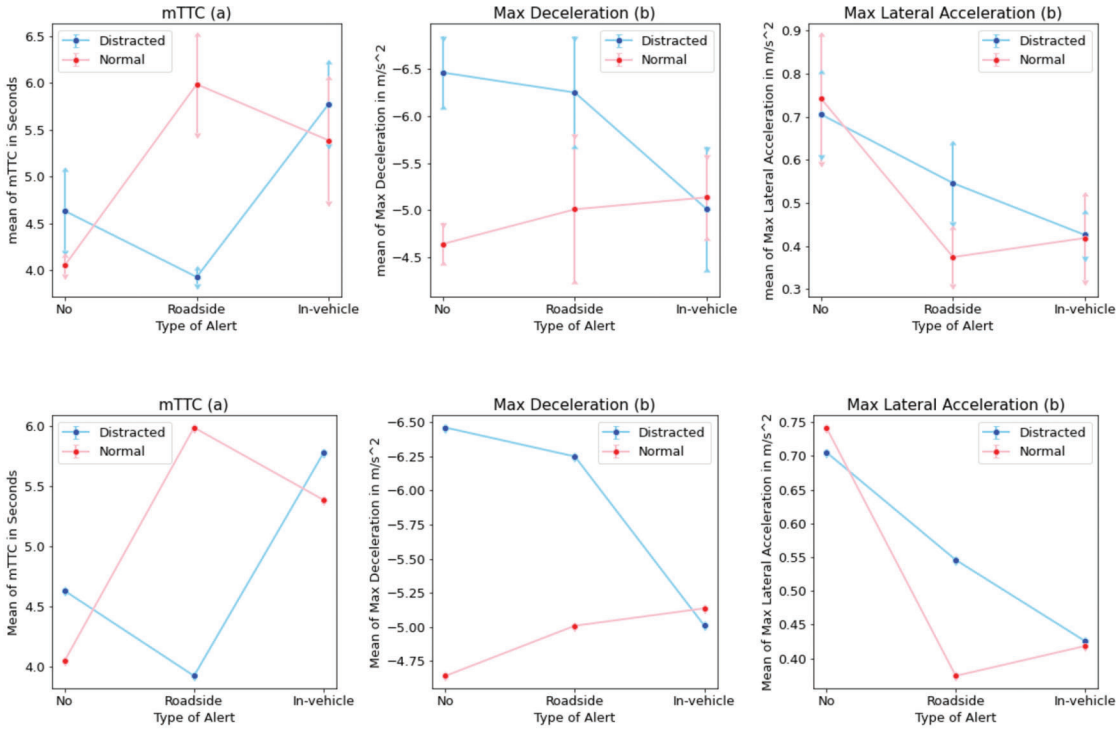
In general, we can conclude that our analysis results show that distracted driving can result in more risks when approaching the BoQ during highway cruising, which is mainly reflected by the maximum deceleration. Thus, Hypothesis 3 is approved in the study.

6.1.5.3 Effects of driver distraction on performance. Lastly, we focus on the effects of driver distraction on BoQ alert efficiency, or in other words, the interactive effects of the two scenario factors. We have ten tests (N = 10) for each scenario in the combination of driver states and alert types. All interaction plots are shown in Figure 6.3, with mean values and standard errors reported. This figure shows how the performance changes with different alert types when the drivers are in normal or distracted conditions.

When the driver is in normal conditions, we can notice that the driving performance is significantly improved in terms of mTTC (larger values) and maximum lateral acceleration (smaller values) when either alert type is applied. The effects of the two types of alerts are similar. Standard errors can confirm the above statements statistically. On the contrary, the maximum deceleration measurement is not much affected by neither types of BoQ alert, compared to the no-alert situation (no statistical differences are shown as all three means of normal driving in Figure 6.3(b) are close to each other with overlapped confidence internals). All of these results can be explained relatively easily. When drivers are in normal conditions, they will pay full attention to the road and thus will notice and read the roadside signs as well as the in-vehicle alerts. Also, they will recognize the queue and respond earlier enough to smooth the deceleration process, which in turn will reduce the benefits of either alert.

The result is quite different when the driver is in distracted conditions, as illustrated by the blue lines in Figure 6.3. It is noticeable that in this situation, there are more differences between the two alert types than the differences between no-alert and roadside alert scenarios for almost all three driving performance measurements. For mTTC, roadside alert actually shows negative effects (reduced mTTC) for distracted drivers, while in-vehicle alert can significantly improve the performance back to the similar level of normal driving with the same alert. Similar trends can be seen for the measurements of the maximum deceleration and the maximum lateral acceleration. For these two variables, the roadside alert can slightly improve the performance by reducing the values (no statistical differences can be approved compared to the no-alert tests), while the in-vehicle alert can significantly improve the performance. These results confirm the benefits of in-vehicle alerts compared to the roadside alert for distracted drivers. When the roadside alert is easy to miss without full attention, in-vehicle alerts can get the drivers' attention more easily and provide more chances for them to read the contents.

It is also noticeable that for roadside alerts, normal driving and distracted driving show large differences in



Note: The arrows represent the standard errors of the mean.

Figure 6.3 Interaction plots for mTTC (left), average maximum deceleration (middle), and average maximum lateral acceleration (right) for two scenario factors (alert types and driver states).

all three performance measurements (statistically different for the mTTC and the maximum lateral acceleration). However, with an in-vehicle alert, the three measurements are consistently improved to a similar level (no statistical differences among the two driver states when an in-vehicle alert is applied) from the distracted driving with roadside alert situations or the baseline situation any alerts.

To conclude, the results have approved the last hypothesis by showing that driver distraction can affect the efficiency of different types of BoQ alerts significantly. Compared to the roadside alert, in-vehicle alert shows much better results for distracted driving but similar results for normal driving.

6.1.1 Summary and Conclusion

Most of the driving simulator study results are reasonable and in accordance with intuition, while some remain interesting. For example, the presence of an in-vehicle alert improves the driver's safety under the BoQ situation by significantly increasing the mTTC (30% longer than that of "no warning") and decreasing the maximum lateral acceleration (60% less than that of "no warning"). We expect that driver states would also significantly affect the "distracted state" that might undermine the driver's safety compared with the "normal state." Although the "in-vehicle auditory alert" dominated the other types of alerts, the results concluded that the "roadside traffic sign alert" performance is similar to that of "no alert" when the driver is distracted.

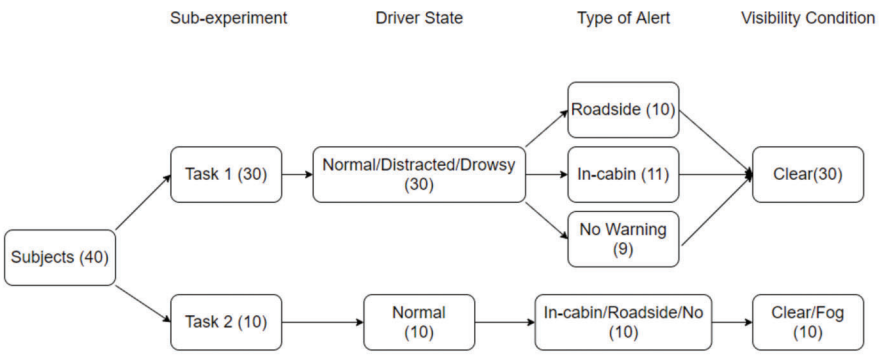
Moreover, all metrics used can only measure the safety of the driver in the BoQ situation. To evaluate the overall performance of different types of alerts, we might need to consider another important measure of the performance such as driving comfort. In future research, we will consider incorporating the metric for this parameter. In summary, this study concludes that the "in-vehicle auditory alert" is the best alerting type among three different alert types for BoQ situations, which suggests its good potential for implementation to improve driving safety.

6.2 HMI Evaluation–Comprehensive Evaluation

6.2.1 Design of Experiment

Though the second driving simulator test is similar to the first test, we improve the test by adding more critical independent variables and increasing the sample size, further examining the effectiveness of the in-cabin auditory alert in BoQ scenarios. This test consists of two independent experiments investigating different combinations of factors.

We recruited 40 subjects to participate in two different sub-experiments (Figure 6.4). In the first experiment, we



Note: The number in parenthesis represents the number of participants in the corresponding test condition.

Figure 6.4 Design of experiment.

used a mixed design (containing both between-subjects and within-subjects designs) to examine the effects of driver states (drowsy, distracted, and normal), type of alerts (in-vehicle auditory alert, roadside traffic sign alert, and no alert), and their interaction. A subject in Task 1 experienced all three different driver states and one type of alert. We used the between-subjects design on the driver states because of the difficulty for a subject to be drowsy multiple times during an experiment session. In Task 2, we applied the within-subjects design to both types of alert and visibility conditions (fog and clear).

Again, the cabin interior set-up is the same to Test 1. The in-vehicle auditory alert is a female-voiced warning, "Slow down! Traffic queue is ahead," followed by a hazardous beeping. We mounted a smartphone in the front of the co-pilot seat to deliver the in-vehicle auditory alert. Both in-vehicle auditory alert and the roadside traffic sign alert are triggered or placed exactly 1 mile away from the traffic queue. Also, we showed subjects both alerts to make them familiar with the alerts. To control the driver states, we adopted two strategies for distracted and drowsy states, respectively. To induce a distracted state, we used a 1-back task. One researcher plays a pre-recorded sequence of numbers (20 Hz), and the subject needs to repeat the previous number at the time they listen to the current one. We did not require any specific accuracy, but we ask them to do their best. Overall, most of the subjects had over 90% of accuracy. We induced the drowsy state naturally and measured the drowsiness through the Karolinska sleepiness scale (KSS) (Kaida et al., 2006). Before starting the drowsy test, we assigned the subject to a preparation test, where the simulator track is an endless circle with no buildings. We showed the subject the KSS and gained the initial score and asked the subject to be sleepy as best as one can while driving. We measured the subject's KSS score every 6 minutes until they achieved the seventh sleepiness score. Once the subject attains the level of sleepiness, we start the drowsy state simulator test immediately. Due to the uncertainty of side effects of drowsiness and inducing time, we arrange the drowsy test as the last test.

Same to Test 1, since each subject experiences multiple times of BoQ scenarios, we designed a pseud-queue to reduce the learning effect. There are none to two waves of slow vehicles placed on the track during each test based on the track's length. When the subject's vehicle arrives at certain checkpoints, the driving simulator generates a wave of slow vehicles ahead to simulate the potential BoQ situation. When the subject approaches the slow vehicles, the slow vehicles in the same lane as the subject's vehicle will change lanes to allow the subject to overtake. In addition to the pseud-queue, we prepared three different length tracks with similar configurations (including driveway, roadside building, etc.). Even though the length of the track is different, we restrict the tracks after alerts (last 1 mile) to be the same to make tracks comparable.

We designed the following general driving guidelines to control the macroscopic driving behavior. (1) Subjects need to maintain in the middle lane for the entire test. (2) Subjects cannot overtake the front vehicle by switching lanes. (3) Subjects should maintain at 65 miles per hour if applicable. (4) Subjects should place their safety as the first priority when driving. The first three guidelines control their general driving behavior, while the last one allows them to demonstrate their true driving behavior when encountering BoQ scenarios. Driving scenarios in the simulator is exactly the same as Test 1.

For the above experiment design, the ultimate goal is to test the hypotheses below.

1. Both in-vehicle auditory alert and roadside traffic alert can improve the safety of the ego vehicle compared with no alert.
2. Drowsy state jeopardizes the ego vehicle's safety the most, distracted states jeopardize the safety less than the drowsy state, and the normal state's safety is the baseline.
3. There are interaction effects of driver states and types of alerts, such as the higher effect of in-vehicle auditory alerts when the driver is drowsy.
4. Worse visibility condition (e.g., foggy) decreases driver safety compared with the normal visibility condition (e.g., clear day).

5. There are interaction effects of visibility conditions and types of alerts, such as the higher effect of in-vehicle auditory alerts when the visibility condition is foggy.

6.2.2 Data Collection

There is a total of 40 (24 male, 16 female) recruited subjects. Most subjects (33) aged in the range of 20–30, where the average age of all subjects was 25.51 (STD = 7.19). Thirty subjects were assigned to Task 1 and 10 subjects were assigned to Task 2. All recruited subjects were screened initially to satisfy the research needs, such as having normal hearing, no virtual reality sickness, and holding a valid driver's license. In addition, we asked the subjects in Task 2 not to consume any product with caffeine within 24 hours before the actual testing. Before subject testing, we explained the use of the driving simulator and 1-back task, respectively. There is a 2-minute break between each test, and a more extended break is rendered if request. Due to the uncertainty of inducing sleepiness, there is no uniform completion time, but the average completion time for test scenarios is about one and a half hours.

6.2.3 Metrics Selection

We selected three widely-used metrics as the indicator of safety, which are the minimum time-to-collision (mTTC), maximum braking (mBraking), and maximum steering angle (mSteer). mTTC measures the time to collision under the current speed, and either a high speed of the vehicle or a short distance away from the incident might cause a small mTTC, which is considered dangerous. Maximum braking is very similar to maximum deceleration, which causes driving discomfort for the driver and may lead to safety concerns on surrounding vehicles. The maximum steering angle is an indicator of vehicle stabilization. In addition to the abovementioned metrics, we also considered the remaining distance (RD) to the traffic queues when mTTC is triggered. Therefore, we have a remaining distance when the driver attains the maximum time-to-collision. A larger RD indicates an earlier engagement to prevent the crash. Note the mTTC and mBraking are calculated for the last 1 mile of the speed profile (after the alert is issued). The mSteering is calculated for the last half mile (where the highway is straight).

6.2.4 Results

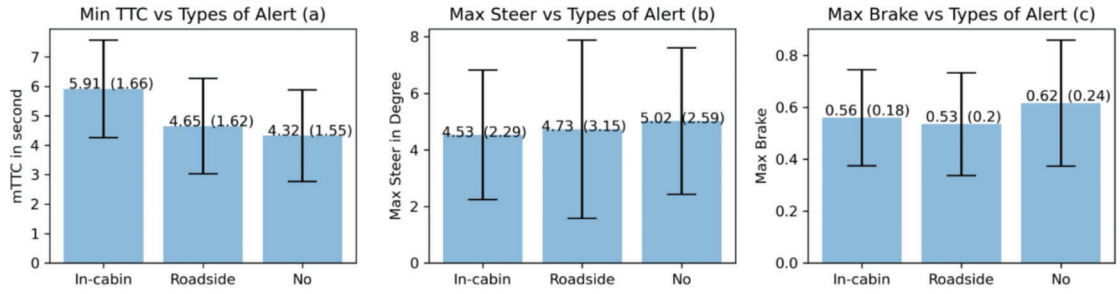
We first performed the analysis of variance (ANOVA) to examine whether the effect of a potential independent variable is statistically significant ($\alpha < .05$). If it is, we then use Tukey's HSD test to make pairwise comparisons between each level of the independent variable. Finally, standard assumptions for the models are used and checked. This section is composed of the results of two tasks, respectively.

6.2.4.1 Driver states and type of alerts. In Task 1, the control variables are driver states and the type of alerts. For each alert type (no alert, roadside traffic sign alert, and in-vehicle alert), there are 11, 10, and 9 subjects with three different driver states, respectively. Therefore, there are 90 data points. We first studied the effects of two types of BoQ alerts on driving safety. When mTTC is the dependent variable, ANOVA results indicate the effect of this type of alert is significant ($F = 9.78$, p-value $< .001$), which means that there is a different within-group variation of mTTC between each group. We further applied post-hoc test. The mean comparisons of in-vehicle and roadside alert (mean difference = -1.3251, p-value = .001) and in-vehicle and no alert (mean difference = -1.6516, p-value $< .006$) are significant, which means the in-vehicle auditory alert improves the mTTC by 1.3 and 1.7 seconds when compared with roadside alert and no warning, respectively. In other words, drivers with in-vehicle auditory alerts averagely have more than 1 second to perform maneuvers than roadside traffic signs and no alerts (shown in Figure 6.5).

In the model with either mSteer ($F = 0.344$, p-value = .71) or mBrake ($F = 1.09$, p-value = .34) as the independent variable, the effect of alert types is not significant. Therefore, the type of alert does not change the vehicles' stabilization and maximum braking. Note that mBrake is not necessarily the same as the maximum deceleration because mBrake does not measure the time length of braking. Pressing the braking pedal a longer time and a shorter time with the same force will result in completely different deceleration but the same mBrake. Figure 6.5 shows that there is little difference in the average mBrake and mSteer across the groups.

Figure 6.6 demonstrates the last metric, remaining distance when attaining mTTC. The average RD for the in-cabin auditory alert is 94.09 meters (STD = 70.76), which is 20 meters longer than roadside and no alert. Moreover, we can see the majority of green (no warning) and orange(roadside) points are located at the bottom-right corner, which is considered the most dangerous situation (low TTC and remaining distance). On the other hand, the blue (in-vehicle) points are located across the top region, which indicated a relatively safe TTC with uniformly distributed RDs.

Then we studied the effect of driver state. When the dependent variable is mTTC, the ANOVA results indicate that the effect of driver state is significant ($F = 8.11$, p-value $< .001$). In addition, the mean difference between drowsy and normal states (mean difference = 1.62, p-value = 0.001) is significant, while the other two comparisons are not. In other words, we found the drowsy state reduces the mTTC by 1.6 seconds when compared with the normal state (shown in Figure 6.7). Again, when either the mBrake or mSteer is the dependent variable, the driver state's effect is not significant. Figure 6.8 shows similar results, where the green (drowsy) and orange (distracted) points are located at the bottom right corner.



Note: The number in the parenthesis is the standard deviation of the samples.

Figure 6.5 Average mTTC (left), average mSteer (middle), and mBrake (right) for different types of alerts for Task 1.

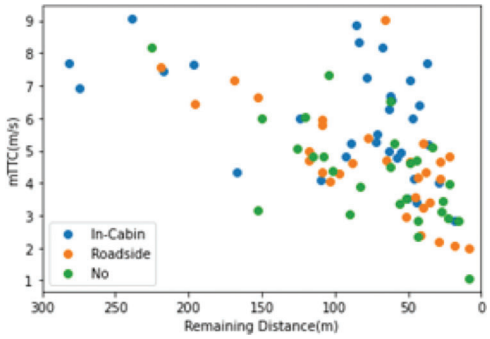
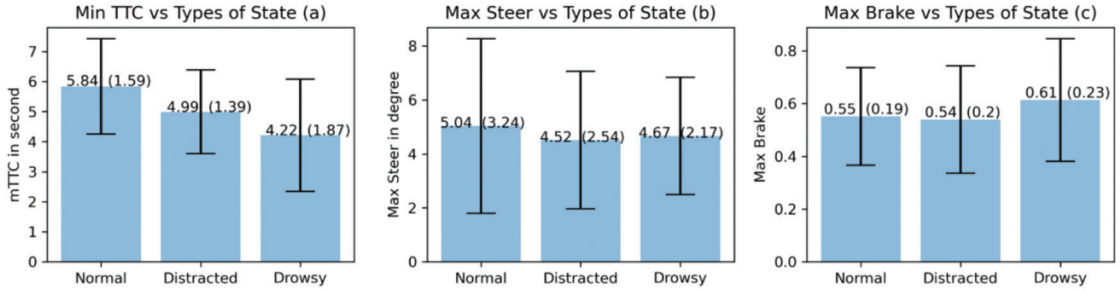


Figure 6.6 Scatter plot of alert types.



Note: The number in the parenthesis is the standard deviation of the samples.

Figure 6.7 Average mTTC (left), average mSteer (middle), and average mBrake (right) for different driver states.

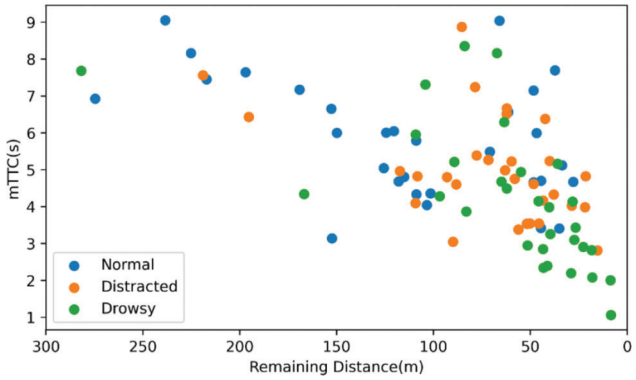


Figure 6.8 Scatter plot of driver states.

Moreover, the effects of the interaction between driver states and type of alerts in all three models are not significant (mTTC: F = 1.06, p-value = .38; mBrake: F = 0.39, p-value = .88; mSteer: F = 0.74, p-value = .56). We conclude that there is no effect of the interaction between driver states and type of alerts.

6.2.4.2 Type of alerts and visibility conditions. In Task 2, we control the visibility conditions and type of alerts. Since this task utilized a between-subjects design (three types of alerts and two visibility conditions), there are 60 data points for 10 subjects. Again, we first examine the effect of the type of alerts. When mTTC is the dependent variable, the ANOVA results indicate the effect of the type of alert is significant (F = 5.05,

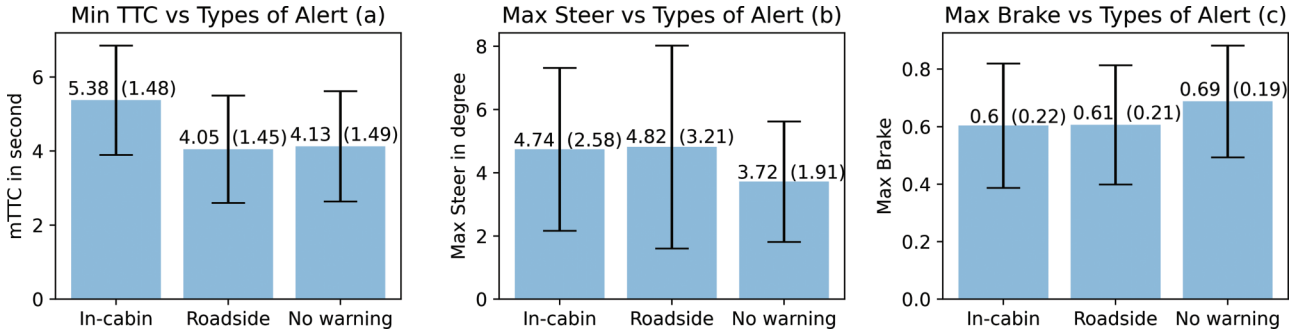
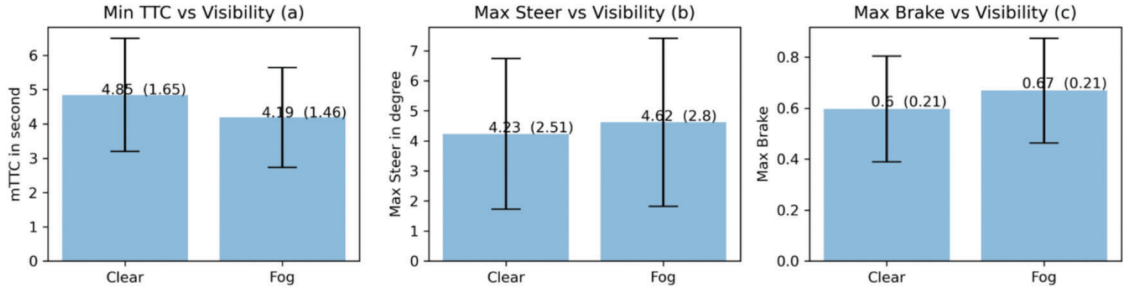


Figure 6.9 Average mTTC (left), average mSteer (middle), and mBrake (right) for different types of alerts for Task 2.



Note: The number in the parenthesis is the standard deviation of the sample.

Figure 6.10 Average mTTC (left), average mSteer (middle), and mBrake (right) for different visibility conditions.

p-value = .009), which means different types of alerts have different mTTCs. Tukey HSD test found the mean differences between in-vehicle and no alert (mean difference = -1.25, p-value = 0.03) and in-vehicle and roadside alert (mean difference = -1.33, p-value = 0.02) are statistically significant, which is consistent with the results from Task 1. In other words, in-vehicle auditory alerts increase the mTTC by more than 1 second when compared with the roadside alert and no alert. When either mBrake (F = 1.05, p-value = 0.36) or mSteer (F = 1.00, p-value = 0.37) is the dependent variable, the effect of type of alerts is not significant. Although the effect of alert types is not significant on mBrake, Figure 6.9 shows that the average mBrake of no alert is slightly higher than the roadside and in-vehicle alert. If the sample size is larger, the effect might be significant.

Then, we examine the effect of the visibility condition. We found the effect of visibility condition is close to significant in the model of mTTC (F = 2.97, p-value = 0.09), but it is far from significant in the other models (mBrake: F = 1.81, p-value = 0.18; mSteer: F = 0.29, p-value = 0.60). The average scores of each metrics are showed in Figure 6.10. In addition, Figure 6.11 shows the orange points (foggy condition) are more concentrated in the bottom right corner, which means the distances attaining mTTC under foggy condition are shorter than the clear condition.

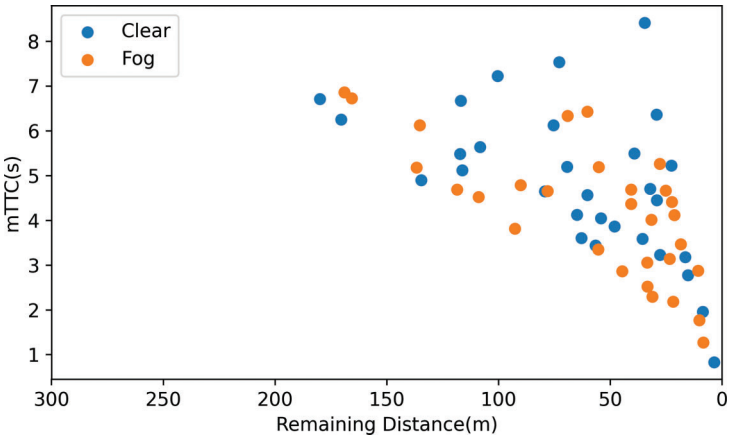


Figure 6.11 Scatter plot of different visibility conditions.

6.2.5 Post Survey

As mentioned earlier, we did a short interview each time the subject finished one trial of the driving simulator test. There are at most three questions for each interview.

1. How do you feel about your braking timing in the previous driving? Choose one option on the following scale based on the corresponding driving simulator test. 1 = very early; 2 = moderately early; 3 = slightly early; 4 = right on time; 5 = slightly late; 6 = moderately late; 7 = very late.
2. Choose one option on the following scale to indicate the satisfaction to the in-vehicle auditory alert based on the corresponding driving simulator test. 1 = completely satisfied; 2 = mostly satisfied; 3 = somewhat satisfied; 4 = neither satisfied nor dissatisfied; 5 = somewhat dissatisfied; 6 = mostly dissatisfied; 7 = completely dissatisfied.
3. Tell us about the whole experiment experience (optional).

We asked the second question if the subject experienced the in-vehicle alert in the corresponding trial. Unfortunately, due to the misoperations, we delete three subject's survey results. Therefore, there are in total 37 collected surveys.

Since the second question aimed to collect subjects' feedback on the proposed in-vehicle alert, we calculate the average score to indicate its satisfaction. The average score for the second question is 3.31 (51 trails with in-vehicle alert). The score suggests slightly positive feedback on the in-vehicle alert. Some subjects gave some explanations for low scores (dissatisfied) on the satisfaction in question 3. We summarized the most common complaints below.

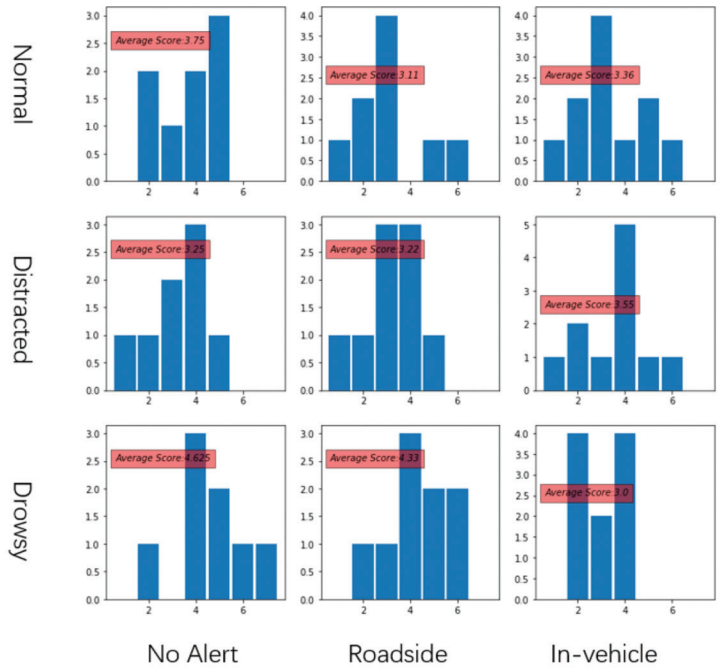
1. The in-vehicle alert is too loud and startling.
2. The in-vehicle alert is triggered too early.
3. The driving simulator is not comfortable to drive.

We could see the first two comments pointing to the in-vehicle alert, while the last one is about the driving simulator. The first two comments are quite straightforward. The first comment shows the inability of the alert customization, while the second one shows that not everyone prefers the soft alert. Adding customization features on both the alert sound and trigger condition (distance before traffic queue) might alleviate the problem. Note, even though the abovementioned comments are common in the surveys, many subjects (especially whose are satisfied with the alert) did not answer question 3. Therefore, the alert is still valid and satisfying according to the average score.

For the first question, the analysis is much more complicated than the second one. Since the scale in question 1 is bi-direction (neutral attitude is in the middle), the average score does not necessarily indicate a moderate attitude on their experience of encountering traffic queues. Therefore, we graphed the distributions for each test environment in Figure 6.12, which demonstrates that the subjects agree with an early to the right-on-time braking when in-vehicle alert presence. In addition, the subjects think they break late when they are drowsy and no in-vehicle alert presence.

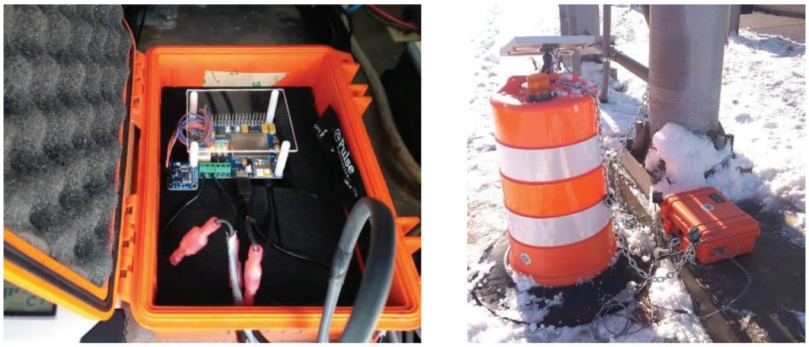
6.2.6 Summary and Conclusion

This test implemented an in-vehicle highway BoQ alerting system based on an Android-based smartphone app. Then a mixed design (containing both between-subjects and within-subjects designs) to examine the effects of driver states, type of alerts, and visibility conditions was devised for assessing the effectiveness of the proposed alerting system. The data analysis on the collected data indicated the effectiveness of the in-vehicle auditory alert (Hypothesis 1). Hypothesis 2 was also supported by the results, where drivers in drowsy states perform the worst in the BoQ scenarios. Although the effect of visibility condition is not statistically significant, we found that the average difference between clear and foggy conditions is close to significant (Hypothesis 4). Both hypotheses of the interaction effect are not supported by the results, so we conclude



Note: The x-axis for each subplot is the score, and the y-axis is the number of person-time.

Figure 6.12 Distribution plots for each test environment.



(a) Embedded computer (b) Device in field

Figure 6.13 Field device for testing message delivery system.

that there is no interaction effect between the type of alerts and driver states/visibility conditions.

6.3 Field Device

A field device was developed to test and verify the speed retrieval API at the Purdue traffic lab. A small, cellular-enabled credit card-sized embedded computer automatically detected its own location and orientation via sensors, and using that data formulated a request to the API once a minute. Speed data at and downstream of the device in the direction of travel was returned and based on the pre-set thresholds, the device logic decided whether or not to trigger a flashing strobe alert.

6.3.1 Hardware

The embedded computer consisted of a Raspberry Pi Zero device, a cellular IoT application shield, and a compass, and operated at 5-volts. The application shield was connected to the Raspberry Pi via a 40-pin GPIO header. The application shield had built-in LTE cellular capability, GNSS module, a three-axis accelerometer, and a relay that can control up to 60 watts. External antennae were required for the LTE cellular module and GNSS. An additional compass module was connected to the application shield via GPIO to determine heading orientation of the device.

The device was powered by an 88-watt-hour lithium-ion battery pack. The battery was replenished by a 28-watt solar panel. For a test beacon, a small 6-watt LED flasher was used. The entire system was then housed in a protective waterproof case. Figure 6.13a shows the working configured device.

6.3.2 Test Deployment

In November 2019, the field device was deployed along I-70 eastbound 1.1 miles east of Holt Road. A traffic barrel was deployed with the solar panel and flashing beacon mounted, with the device in a separate case. The setup was placed at the base of the Dynamic Message Sign at that location where the flashing beacon would be hidden from view of traffic. An additional time-lapse camera was installed 40-feet away to monitor the status of the beacon as traffic conditions changed throughout the day. Figure 6.13b shows a picture of the deployment.

7. SUMMARY

An integrated critical information delivery platform for smart segment dissemination to road users was developed. A statewide baseline milepost geodatabase was created at 0.1-mile resolution along with tools, protocols, and interfaces that allow other data sources to be efficiently utilized. A variety of data sources (e.g., INRIX, CARS, Doppler, camera images, connected vehicle data, automated vehicle location) were integrated into existing and new dashboards for stakeholders to monitor roadway conditions and after-action reviews. Additionally, based on these data sources, algorithms were developed and an API was created to identify hazardous road conditions when the location of the end-user mobile device was given. Message delivery schemes were successfully implemented to issue alerts to drivers, which were integrated with two in-vehicle smartphone applications. The performance of the integrated platform was evaluated using both the driving simulator and a number of simulated and on-road tests. The results demonstrated the system was able to disseminate data in real-time using the developed platform.

REFERENCES

Bentley, J. L. (1975, September). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*(9), 509–517. https://doi.org/10.1145/361002.361007

Desai, J., Li, H., Mathew, J. K., Cheng, Y.-T., Habib, A., & Bullock, D. M. (2021). Correlating hard-braking activity with crash occurrences on interstate construction projects in Indiana. *Journal of Big Data Analytics in Transportation*, *3*(12), 27–41.

Desktop Doppler. (n.d.). *NWS NEXRAD* [Webpage]. Retrieved July 1, 2021, from https://web.archive.org/web/20160113151652/http://www.desktopdoppler.com/help/nws-nexrad.htm#rainfall rates

GitHub. (n.d.). *GitHub docs* [Webpage]. Retrieved June 30, 2021, from https://guides.github.com

Hayward, J. C. (1972, January 17–21). *Near-miss determination through use of a scale of danger* [Paper presentation]. 51st Annual Meeting of the Highway Research Board, Washington D.C., United States.

INDOT. (n.d.a). *Protect the queue* [Webpage]. Retrieved July 15, 2021, from https://www.in.gov/indot/4076.htm

INDOT. (n.d.b). *TrafficWise* [Webpage]. Retrieved June 30, 2021, from https://indot.carsprogram.org/#roadReports?timeFrame=TODAY&layers=roadReports%2CwinterDriving%2CweatherWarnings%2Cflooding%2CallReports

Kaida, K., Takahashi, M. Akerstedt, T. Nakata, A., Otsuka, Y., Haratani, T., & Fukasawa, K. (2006, July). Validation of the Karolinska sleepiness scale against performance and EEG variables. *Clinical Neurophysiology*, *117*(7), 1574–1581.

Kamiński, B., Jakubczyk, M., & Szufel, P. (2017). A framework for sensitivity analysis of decision trees. *Central European Journal of Operations Research*, *26*(1), 135–159.

Kirchner, W. K. (1958, April). Age differences in short-term retention of rapidly changing information. *Journal of Experimental Psychology*, *55*(4), 352–358.

Li, H., Day, C. M., & Bullock, D. M. (2016, November 1–4). *Virtual detection at intersections using connected vehicle trajectory data* [Conference session]. 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil. https://doi.org/10.1109/ITSC.2016.7795969

Li, H., Remias, S. M., Day, C. M., Mekker, M. M., Sturdevant, J. R., & Bullock, D. M. (2015). Shock wave boundary identification using cloud-based probe data. *Transportation Research Record: Journal of the Transportation Research Board*, *2526*(1), 51–60. https://doi.org/10.3141/2526-06

Li, H., Wolf, J. C., Mathew, J. K., Navali, N., Zehr, S. D., Hardin, B. L., & Bullock, D. M. (2020). Leveraging connected vehicles to provide enhanced roadway condition information. *Journal of Transportation Engineering, Part A: Systems*, *146*(8), 04020073.

Li, L., Chen, Y., Tian, R., & Li, F. (2019). *Back of queue warning and critical information delivery to motorists* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2019/24). West Lafayette, IN: Purdue University. https://doi.org/10.5703/1288284317102

Libpysal. (n.d.). *Source code for libpysal.cg.kdtree*. Retrieved July 1, 2021, from https://pysal.org/libpysal/_modules/libpysal/cg/kdtree.html

Mekker, M. M., Remias, S. M., McNamara, M. L., & Bullock, D. M. (2020). *Characterizing interstate crash rates based on traffic congestion using probe vehicle data* (JTRP Affiliated Reports, Paper 31). https://doi.org/10.5703/1288284317119

NSSL. (n.d.a). *NSSL projects: Multi-radar/multi-sensor system (MRMS)* [Webpage]. NOAA National Severe Storms Laboratory. Retrieved June 30, 2021, from https://www.nssl.noaa.gov/projects/mrms/

NSSL. (n.d.b). *Operational MRMS GRIB2 tables* [Webpage]. NOAA National Severe Storms Laboratory. Retrieved July 1, 2021, from https://www.nssl.noaa.gov/projects/mrms/operational/tables.php

INRIX. (n.d.). *Overview–INRIX documentation* [Webpage]. Retrieved June 30, 2021, from http://docs.inrix.com/iq/IQ_Product_Description/

INRIX. (2013, October 29). *New INRIX XD™ traffic covers more roads with greater precision than any other service*. PR Newswire. Retrieved June 30, 2021, from https://www.prnewswire.co.uk/news-releases/new-inrix-xd-traffic-covers-more-roads-with-greater-precision-than-any-other-service-229643971.html

Saldivar-Carranza, E. D., Li, H., Mathew, J., Hunter, M., Sturdevant, J., & Bullock, D. (2021). Deriving operational traffic signal performance measures from vehicle trajectory data. *Transportation Research Record: Journal of the Transportation Research Board*, *2675*, 1250–1264. https://doi.org/10.1177/03611981211006725

Shen, D., Zhang, Z., Ruan, K., Tian, R., Li, L., Li, F., Chen, Y., Sturdevant, J., & Cox, E. (2021, January 24–28). *Assessing the effectiveness of in-vehicle highway back-of-queue alerting system* (TRBAM-21-03879). 2021 Transportation Research Board 100th Annual Meeting, Washington D.C., United States.

Zhang, Z., Shen, D., Tian, R., Li, L., Chen, Y., Sturdevant, J., & Cox, E. (2021, September 19–22). *Implementation and performance evaluation of in-vehicle highway back-of-queue alerting system using the driving simulator* [Conference session]. 2021 IEEE Intelligent Transportation Systems Conference, Indianapolis, IN.

About the Joint Transportation Research Program (JTRP)

On March 11, 1937, the Indiana Legislature passed an act which authorized the Indiana State Highway Commission to cooperate with and assist Purdue University in developing the best methods of improving and maintaining the highways of the state and the respective counties thereof. That collaborative effort was called the Joint Highway Research Project (JHRP). In 1997 the collaborative venture was renamed as the Joint Transportation Research Program (JTRP) to reflect the state and national efforts to integrate the management and operation of various transportation modes.

The first studies of JHRP were concerned with Test Road No. 1—evaluation of the weathering characteristics of stabilized materials. After World War II, the JHRP program grew substantially and was regularly producing technical reports. Over 1,600 technical reports are now available, published as part of the JHRP and subsequently JTRP collaborative venture between Purdue University and what is now the Indiana Department of Transportation.

Free online access to all reports is provided through a unique collaboration between JTRP and Purdue Libraries. These are available at http://docs.lib.purdue.edu/jtrp.

Further information about JTRP and its current research program is available at http://www.purdue.edu/jtrp.

About This Report

An open access version of this publication is available online. See the URL in the citation below.

Li, L., Chen, Y., Tian, R., Li, F., Li, H., & Sturdevant, J. R. (2021). *An integrated critical information delivery platform for smart segment dissemination to road users* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2021/34). West Lafayette, IN: Purdue University. https://doi.org/10.5703/1288284317440