TECHNICAL REPORT STANDARD PAGE

1. Report No. FHWA/LA-	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Development of a Congestion Management System Using GPS Technology	5. Report Date April 1997 6. Performing Organization Code	
7. Author(s) Darcy Bullock and Cesar A. Quiroga	8. Performing Organization Report No. 299	
9. Performing Organization Name and Address Remote Sensing and Image Processing Laboratory Louisiana State University Baton Rouge, LA 70803	10. Work Unit No. 11. Contract or Grant No.	
12. Sponsoring Agency Name and Address Louisiana Transportation Research Center 4101 Gourrier Avenue Baton Rouge, LA 70808	13. Type of Report and Period Covered Final Report January 1995 - November 1996 14. Sponsoring Agency Code	

15. Supplementary Notes

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

16. Abstract

This report describes the results of a study undertaken to demonstrate the feasibility of using global positioning system (GPS) and geographic information system (GIS) technologies to measure travel time and speed data on urban highways. Compared to more traditional approaches for conducting travel time studies, which require a significant amount of manual field work and are prone to recording errors, the methodology described here dramatically increased productivity, as well as virtually eliminated data collection and data reduction errors.

The methodology described in this report includes data collection, data reduction, and data reporting procedures. The data collection procedure is based on the use of GPS receivers to automatically collect time, local coordinates, and speed every one second. This way, an accurate depiction of vehicle location and speed is obtained. The data reduction procedure is based on the aggregation of GPS data using highway segments which are normally 0.2 miles (mi) in length. However, the model is sufficiently general so that other segment sizes can be easily accommodated. The data reporting procedure uses a GIS-based management information system to define queries, tabular reports, and color coded maps to document travel time data along the corridor segments. Examples of such data includes travel times, average speeds, minimum speeds, and delays. Color coded maps show the spatial variation of items such as speed and travel time, and are particularly suitable for explaining travel time delays and congestion issues at public meetings. Tabular reports offer a very compact way of archiving travel time and speed data along highway segments. This makes them suitable for archival and analytical purposes. The procedure to produce these tabular reports has been automated, therefore increasing the usefulness of such an approach. Reporting procedures using world wide web (WWW) resources are also implemented. These procedures allow any user with access to the Internet to select highway segments and retrieve all records associated with these segments, and to retrieve real-time travel time data between checkpoints located on a highway corridor.

The methodology described here was used to obtain travel time and speed data needed for developing congestion management systems (CMSs) in Baton Rouge, Shreveport, and New Orleans. In Baton Rouge, 25,000 mi of travel runs were made on a 151 mi highway network. From a total of 428 GPS data files and 2.5 million GPS points recorded, use of the methodology resulted in 155,300 segment records. In Shreveport, 844 mi of travel runs were made on a 93 mi highway network. From a total of 100 GPS data files and 85,000 GPS points recorded, use of the methodology resulted in 5,048 segment records. In New Orleans, 3,805 mi of travel runs were made on a 86 mi highway network. From a total of 68 GPS data files and 322,000 GPS points recorded, use of the methodology resulted in 22,613 segment records.

17. Key Words Congestion Management, GPS, GIS, travel time studies, data collection	18. Distribution Statement Unrestricted. This document is available through the National Technical Information Service, Springfield, VA 21161.

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

Form DOT F1700.7 (1-92)

DEVELOPMENT OF A CONGESTION MANAGEMENT SYSTEM USING GPS TECHNOLOGY

VOLUME I

By

Darcy Bullock
Assistant Professor

Cesar A. Quiroga
Research Associate

Civil and Environmental Engineering Department
Remote Sensing and Image Processing Laboratory
Louisiana State University
Baton Rouge, La 70803

LTRC RESEARCH PROJECT 95-7SS
STATE PROJECT 736-99-0235

Conducted for

LOUISIANA DEPARTMENT OF TRANSPORTATION AND DEVELOPMENT
LOUISIANA TRANSPORTATION RESEARCH CENTER

April 1997

ABSTRACT

This report describes the results of a study undertaken to demonstrate the feasibility of using global positioning system (GPS) and geographic information system (GIS) technologies to measure travel time and speed data on urban highways. Compared to more traditional approaches for conducting travel time studies, which require a significant amount of manual field work and are prone to recording errors, the methodology described here provides consistency, automation, finer levels of resolution, and better accuracy in measuring travel time and speed.

The methodology described in this report includes data collection, data reduction, and data reporting procedures. The data collection procedure is based on the use of GPS receivers to automatically collect time, local coordinates, and speed every one second. This way, an accurate depiction of vehicle location and speed is obtained. The data reduction procedure is based on the aggregation of GPS data using highway segments which are normally 0.2 miles (mi) in length. However, the model is sufficiently general so that other segment sizes can be easily accommodated. The data reporting procedure uses a GIS-based management information system to define queries, tabular reports, and color coded maps to document travel time data along the corridor segments. Examples of such data includes travel times, average speeds, minimum speeds, and delays. Color coded maps show the spatial variation of items such as speed and travel time, and are particularly suitable for explaining travel time delays and congestion issues at public meetings. Tabular reports offer a very compact way of archiving travel time and speed data along highway segments. This makes them suitable for archival and analytical purposes. The procedure to produce these tabular reports has been automated, therefore increasing the usefulness of such an approach. Reporting procedures using world wide web (WWW) resources are also implemented. These procedures allow any user with access to the Internet to select highway segments and retrieve all records associated with these segments, and to retrieve real-time travel time data between checkpoints located on a highway corridor.

The methodology described here was used to obtain travel time and speed data needed for developing congestion management systems (CMSs) in Baton Rouge, Shreveport, and New Orleans. In Baton Rouge, 25,000 mi of travel time runs were made on a 151 mi highway network. From a total of 428 GPS data files and 2.5 million GPS points recorded, use of the methodology resulted in 155,300 segment records. In Shreveport, 844 mi of travel runs were made on a 93 mi highway network. From a total of 100 GPS data files and 85,000 GPS points recorded, use of the methodology resulted in 5,048 segment records. In New Orleans, 3,805 mi of travel runs were made on an 86 mi highway network. From a total of 68 GPS data files and 322,000 GPS points recorded, use of the methodology resulted in 22,613 segment records.

ACKNOWLEDGMENTS

This project was supported by the Louisiana Department of Transportation and Development (LADOTD), through the Louisiana Transportation Research Center (LTRC Research Project 95-7SS; State Project 736-99-0235). The help of Jim Joffrion, Art Rogers, Wanda Vick, Huey Dugas, Jay Naidu, Chris Petro, Nitin Kamath, Chris Schwehm, and Hong-Lie Qiu is gratefully acknowledged. Also recognized is the work done by all undergraduate students who drove the probe vehicles and helped in data processing.

IMPLEMENTATION ISSUES

Hardware and software requirements

Considerations for this project included hardware and GIS software already available at DOTD and the Baton Rouge, Shreveport, and New Orleans metropolitan planning organizations (MPOs): Capital Region Planning Commission (CRPC) in Baton Rouge; Northwest Louisiana Council of Governments (NLCOG) in Shreveport; and Regional Planning Commission (RPC) in New Orleans. Because compatibility was essential to ensure an efficient use of resources and computing power, as well as a smooth transition from one agency to another, Intergraph's Modular GIS Environment (MGE) with Oracle v.7 as the underlying database package was chosen for the project. Both MGE and Oracle v.7 were run on a Windows NT computer.

Microsoft's Access v.7 was used for handling most non-spatial queries and for developing archival tabular reports. The connectivity between Access and Oracle was made through the use of an Open Database Connectivity (ODBC) driver. This driver allows any inexpensive personal computer (PC) to communicate directly with the Oracle databases used by Intergraph workstations. The archival tabular reports are produced with a series of concatenated macros that generate all the queries needed to populate all fields in the report template. Because of the amount of information generated, it was decided to design such a template assuming an 11"x17" layout.

In Baton Rouge and New Orleans, Trimble Placer GPS 400 receivers were used to collect travel time and speed data. On-board differential correction was made using real-time data collected with Differential Corrections Inc's. (DCI's) Remote Data System (RDS) 3000 receivers via FM subcarriers. The actual differential correction took place inside the GPS 400 receivers by simultaneously processing the uncorrected satellite data and the FM correction data at the same time. The differentially corrected GPS (DGPS) data was then stored in laptop computers.

In Shreveport, a Pathfinder ProXL Trimble unit with recording capabilities was used to collect travel time and speed data. Differential correction was made by post-processing this raw GPS data with GPS data from a 4000 SSE geodetic surveyor base station receiver. Shreveport used GPS equipment of much higher specifications than Baton Rouge or New Orleans did because it already had that equipment for surveying purposes. Strictly speaking, however, any GPS equipment capable of providing a horizontal positional accuracy between one and five meters spherical error probability (SEP) using differential correction is more than adequate. More stringent is the requirement on speed data. Whatever GPS brand or unit is chosen, its velocity computation algorithm must be based on both pseudorange (distance from satellite to receiver) and pseudorange rate data. Velocity computation based on coordinates of adjacent position fixes is not acceptable. Likewise, it is important that the GPS receiver used have the capability to output data in ASCII format.

Electronic deliverables

To assist DOTD in the process of implementing the system described in this report, the following files are also delivered on an accompanying CD:

1. Three MGE project files (cmsbtr.mpd, cmsshv.mpd, and cmsmsy.mpd). These files are MGE export files (.mpd extension) that contain all maps, including the segmented network, intersecting streets, master file for production of color-coded maps, and ancillary data. They also contain all database tables needed to make queries by corridor, date, time of day, and other qualifiers. Each file is the basic file needed by the corresponding MPO to generate a copy of the project in its Intergraph workstation.
2. Three Access v.7 database files (cmsbtr.mdb, cmsshv.mdb, and cmsmsy.mdb). These files are .mdb files that contain the tables, queries, and macros needed to produce archival tabular reports. An accompanying readme.txt file provides the instructions to generate and print report pages.
3. Three sets of GPS point data files (one for each MPO). These files have an .ndc extension and contain the original GPS position fixes, time, and speed. These ASCII files are input files to the data reduction utility.
4. Two conversion files (cms-cvt.exe and shv-cvt.exe). These files are executable files that convert the original raw GPS data files into GPS data files with an .ndc extension. One conversion file is for data collected with the Trimble Placer GPS 400 receivers. The other conversion file is for data collected with the Pathfinder ProXL Trimble unit.
5. One Microstation Developing Language (MDL) file (cms2.ma). This file is an executable file that contains the data reduction utility written to aggregate GPS point data. In order to load the application efficiently within Microstation, file cms2.ma should be located in the following directory: c:\ingr\ustation\mdlapps\.

In order to accelerate project implementation at each MPO, copies of the MGE project files (item 1) and archival tabular reports (appendixes F, G, and H in volumes II, III, and IV[1]), which are the output from item 2, have already been sent to the Baton Rouge, Shreveport, and New Orleans MPOs.

In general, project implementation is not expected to require significant changes on the files submitted. The MGE project files, the MDL executable file, and the Access v.7 database files are standard software application files which do not require any level of software customization. The conversion utilities are executable files that can be run from the DOS prompt. The only customization that will likely be required is modification of the conversion utility to accommodate new GPS receivers that may be used for data collection.

[1] Volumes II, III, and IV are available at LTRC upon request.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Page

INTRODUCTION

One of the mandates of the Intermodal Surface Transportation Efficiency Act (ISTEA) of 1991 was the development of Congestion Management Systems (CMSs) *[1]*. Also included in the mandate was the development of statewide highway traffic monitoring systems. Based upon air quality attainment status, defined in the Clean Air Act Amendment (CAAA) of 1990 *[2]*, Metropolitan Planning Organizations (MPOs) were authorized by the Federal Government to draw funding from the Congestion Mitigation and Air Quality Program (CMAQ) to develop and implement CMSs for non-attainment areas. The federal mandate of 1991 also established deadlines for developing CMSs. For areas rated non-attainment for mobile emissions, the deadline for an operational CMS was October 1, 1995. The deadline for the remaining areas was October 1, 1996.

An important component in the development of CMSs is the capability to measure congestion accurately and reliably. This is particularly critical at a time when the construction of new facilities nationwide has virtually stopped and, as a result, most of the interest is now in the implementation of more localized solutions to mitigate congestion. However, most techniques currently in use to measure congestion were developed at a time when massive and long construction projects were the norm and, consequently, there was little regard for documenting localized problems. What is needed, then, is the development of techniques that will satisfy the needs for documenting these localized problems.

Most methods used to measure congestion involve collecting travel time data in one way or another. However, current techniques for collecting travel time data are labor intensive, tend to be expensive, and are prone to frequent errors both in the field and in the office. As a result, only a few runs are usually made and tend to be plagued with inaccuracy problems. Therefore, it would be of interest to develop a methodology which would substantially alleviate the problems of current techniques and which would allow MPOs to conduct the necessary travel time studies they need for the implementation of their CMSs.

OBJECTIVES

The goal of this research study is to demonstrate the feasibility of using a global positioning system (GPS) - geographic information system (GIS) approach to measure travel time and speed data needed for the development of CMSs in major metropolitan areas in Louisiana. Its main contribution is the development of a cost-effective methodology including data collection, data reduction, and data reporting procedures. Specific objectives are summarized as follows:

- Development of an efficient travel time data collection methodology.
- Development of an efficient procedure for producing GPS-based highway network maps suitable for travel time studies. These directional centerline network maps allow for an accurate mapping of GPS-derived travel time data to highway segments.
- Development of a procedure and accompanying software to link GPS-derived travel time and speed data to highway segments. This procedure allows for the reduction of GPS point data into segment-based traffic data.
- Development of a procedure for reporting segment travel time and speed data both in graphical and tabular formats. Such a procedure also includes dissemination of travel time information using Internet resources.

SCOPE

This research study is concerned with the development of a methodology for conducting travel time studies using GPS and GIS. More specifically, the project focuses on the development of a suitable spatial model, a data collection procedure, a relational database model, a data reduction procedure, and a data reporting procedure. The methodology is general in the sense that it can be applied to a variety of urban scenarios where travel time and delay are used as primary measures of effectiveness of highway system performance. This is the case of the three metropolitan areas in Louisiana (Baton Rouge, Shreveport, and New Orleans) described in this report.

This research study focuses on travel time and delay only, and not on other variables like flow rates, traffic composition, availability of alternative modes of transportation, and characteristics of the highway network, all of which play an important role in the process of assessing congestion in urban areas. This means that the results obtained with this research project must be viewed as a foundation block rather than the final composition of a congestion management system.

The GPS-GIS methodology described in this report can be applied to a variety of levels of resolution, both in space and time. Because short highway segments, say about 0.2 mi, can be used, the methodology can be applied to detect and analyze fairly localized congestion problems. However, the methodology was designed to measure delay along corridors rather than delay at specific points. As a result, issues such as the sensitivity of the methodology to positional errors associated with individual GPS points were not deemed critical and, consequently, were not addressed. It is acknowledged, however, that for the methodology to be applied at the micro level, such issues would have to be considered.

This research project addresses the needs of metropolitan areas like Baton Rouge, Shreveport, and New Orleans for developing their congestion management systems. However, it does not attempt to define specific implementation procedures like ranking of congested sections, design and implementation of congestion monitoring plans, or coordination with other transportation management systems.

METHODOLOGY

Data Collection Methodology

Traditional approach

Two techniques have traditionally been used to measure travel time: the license plate technique, and the floating car technique *[3], [4]*. With the automation provided by computers and other electronic devices, additional techniques have emerged over the years, including automatic vehicle identification (AVI), automatic vehicle location (AVL), cellular phone tracking, and video imaging *[5], [6]*. For comparison purposes, these techniques can be grouped into two categories: roadside techniques and vehicle techniques.

Roadside techniques

These techniques are based on the use of detecting devices physically located along the study routes at pre-specified intervals. They obtain travel time information from vehicles traversing the route by recording passing times at predefined checkpoints. License plate matching and AVI are examples of these techniques. License plate matching is based on recording the license plate number of individual vehicles and the corresponding time stamps as they pass control points. Travel times are determined as the difference in time between control points. An assumption of this technique is that each individual vehicle does not make intermediate stops. This may be limiting, particularly if there are intersections, on-ramps, off-ramps, or interchanges between control points. Computer vision systems are now being developed to automate this technique. They reduce the amount of field work, but still do not provide an accurate measurement of behavior between control points. In addition, these automated systems tend to be quite expensive. For example, the video recording equipment for a single observation station could cost nearly $3,000 *[5], [6]*. The corresponding data processing is estimated to cost around $50 per hour of video *[6]*.

AVI is a technology that is now being used in several metropolitan areas both in the United States and abroad. AVI systems consist of in-vehicle transponders (or tags), roadside reading units, a communication network, and a central computer system. The roadside reading units detect individual vehicles equipped with transponders as they pass nearby and transmit the corresponding transponder data to the central computer system. Travel times between consecutive checkpoints are computed in a similar manner as with the license plate technique, except that transponder identification numbers are used to compare time stamps instead of vehicle license plate numbers. AVI systems are typically very expensive. For example, each roadside reader unit costs about $30,000 *[6]*. Because of this, distances between consecutive checkpoints tend to be large (1-5 mi) and, consequently, detecting localized problems becomes much more difficult. One advantage of AVI technology is that area-wide real-time travel time data collection and dissemination are possible. With Internet tools, for example, cities like Houston, Chicago, and Seattle are using AVI technology to communicate up-to-date geo-referenced information regarding speeds and travel times to the traveling public.

Vehicle techniques

These techniques are based on the use of detection devices carried inside the vehicle. Examples of these techniques include the traditional floating car technique, AVL, and cellular phone tracking. In the floating car technique, a single probe vehicle is driven with the traffic flow, i.e., passing as many cars as cars pass the probe vehicle. Travel time and passage of specific landmarks are recorded along the route. In the traditional approach, two people, usually technicians, are required in the car: one of them to drive the vehicle, and the other one to manually record the location and time of individual checkpoints. Such a system is prone to errors, particularly if the distance between contiguous checkpoints is short and the vehicle is moving relatively fast. Nowadays, distance measuring instruments (DMIs) can be attached to the vehicle's transmission to automatically record distance, time and speed. In this case, only one technician is needed in the probe vehicle. However, their use is not error free *[7]*. For example, the actual location of all checkpoints still has to be made manually, which makes the system dependent on the technician's ability to locate checkpoints exactly in the same place run after run. Even on systems that allow users to store checkpoint locations in memory, care must be exercised to ensure that the vehicle starts exactly from the same spot and that the entire route is driven exactly the same from beginning to end. In addition to this, DMIs tend to be relatively expensive. Including a notebook computer for data storage, each unit costs about $2,000.

The floating car technique is one of several techniques grouped under the general names of test vehicle techniques or moving observer techniques *[3]*, *[8]*. In the general case, a probe vehicle is used to measure both travel time and other data such as flow rates, vehicle-miles, and time spent on queues. Flow rates are used, among other things, to refine observed travel time values when the number of vehicles passed by the probe vehicle is not the same as the number of vehicles that pass the probe vehicle. To improve accuracy in estimating flow rates, a second vehicle driving in the opposite direction can be used to count the number of vehicles traveling in the same direction as the first probe vehicle. Notice, however, that using a second vehicle adds to the total cost of the travel time study. Having only one vehicle drive both directions of travel could result in some savings but, unfortunately, the trade off is a decrease in accuracy because counts on both directions of travel are not made simultaneously *[8]*. Also, counting vehicles traveling in the opposite direction while driving is feasible only for low to moderate volumes. This is clearly not the case of most urban congested areas. Furthermore, all portions of the opposing segment must be visible. For this reason, probe vehicles are generally used to measure travel time only. In general, an effort is made to pass as many cars as cars pass the probe vehicle. Under congested conditions, this is relatively easy to accomplish as speed variation by lane is usually very low. During non-congested conditions, speed variation by lane may be quite significant. In this case, approximately average speeds can be obtained by driving on the middle lane *[3]*.

AVL is a generic term that groups several techniques that use receivers or transmitters on-board to determine vehicle location (in latitude and longitude) and speed. Examples of these techniques are ground-based radio navigational systems and GPS. GPS is particularly advantageous because it does not need receiving towers on the ground as traditional radio navigational systems do. One of the significant advantages of AVL compared to other techniques is that traffic monitoring is network and driver independent. This makes AVL suitable for many applications, including tracking the motion of special-purpose probe vehicles and entire fleets. When used with single probe vehicles, AVL systems are usually configured so that data is collected and stored on board, and then post-processed in the office. When used with entire fleets, AVL systems are usually configured so that data is collected and transmitted via radio or cellular phone to a central location where it is immediately processed. AVL costs vary widely depending on the particular application, but in general, they range from $1,000 to $4,500 per vehicle *[6]*.

Cellular phone tracking is an experimental technique that involves locating cellular phones that are being used by motorists on the road. Geolocation is done using cellular phone tower triangulation methods and techniques involving differences in signal time of arrival to those phone towers. Cellular phone tracking systems are increasingly being used because the use of cellular phones among motorists is also increasing *[6]*. Total costs are expected to be relatively low because private motorists can more easily absorb the cost of using their cellular phones. However, system operation is highly dependent on the motorists' willingness to use their cellular phones when they cross pre-specified checkpoints. Overall, cellular phone systems appear to be more feasible for reporting traffic incidents.

Comparison of techniques

Table 1 is a summary of characteristics and applicability of the travel time data collection techniques described previously. Roadside techniques are obviously infrastructure dependent, as opposed to vehicle techniques. Roadside techniques have lower levels of resolution and accuracy than vehicle techniques. However, vehicle techniques, specifically those based on DMIs and AVL, are generally based on a limited number of probe vehicles, which means that area wide coverage is limited. This makes roadside techniques (specifically AVI) better suited for daily or real-time monitoring. In contrast, vehicle techniques are best for determining initial conditions and for annual monitoring.

In practice, most travel time studies are made using vehicle techniques. Many government agencies use DMIs. However, as mentioned before, DMIs are not exactly error free. For this reason and because of budgetary constraints, many other agencies still conduct their travel time studies with the traditional two-people clipboard and stopwatch approach. Not surprisingly, runs made during most travel time studies tend to be extremely low in number and subject to significant accuracy problems *[9] [10]*. Consequently, there is a need to develop automated procedures that increase productivity and reduce both data collection and data reduction errors. Such procedures would result in an increased number of runs in travel time studies so that a statistically significant amount of data could be obtained with little effort *[9]*.

Travel time studies with GPS

GPS is a positional and navigational system developed and operated by the US Department of Defense. Its main component is a constellation of 24 satellites that broadcast signals providing data on their position and trajectory. For strategic military reasons, those signals may be subject to a random intentional degradation process known as selective availability (SA) *[11]*. When SA is disabled, horizontal positional accuracy on the ground may vary from a few millimeters to tens of meters, depending on the GPS receiver used. When SA is enabled, an additional random error is added to the satellite signal which results in the horizontal positional accuracy on the ground to degrade to about 100 meters (m) (two sigmas) *[11]*. In practice, this limitation can be bypassed by operating two GPS receivers simultaneously (one mobile and the other one stationary) and by using the data collected with the stationary unit to differentially correct the data collected with the mobile unit. This correction process is termed differential because differences between the true coordinates of the stationary unit location (determined previously following established surveying and/or geodetic standards) and the coordinates read with the stationary unit are used to correct the coordinates read with the mobile unit. The differentially corrected GPS points are then called DGPS points.

Table 1
Comparison of travel time data collection techniques (adapted from *[5]*, *[6]*)

Criteria	Travel Time Collection Technique					
	Roadside techniques			Vehicle techniques		
	License plate matching		AVI	DMI	AVL	Cellular phone
	Traditional	Video				
Characteristics:						
Infrastructure dependent	yes	yes	yes	no	no	no
Travel time/speed resolution	low	low	low	high	high	unknown
Travel time/speed accuracy	good	good	good	good	very good	unknown
Area wide coverage	low	low	very good	low	low	unknown
Technology status	proven	being tested	proven	proven	proven[1]	being tested
Capital costs	low	high	high	low	low to mod.	mod. to high
Operating costs per unit	moderate	moderate	low	high	low to mod.	low to mod.
Applicability:						
Annual monitoring	yes	yes	yes	yes	yes	yes
Daily monitoring	limited	limited	yes	limited	limited	limited
Real-time travel information	limited	limited	yes	no	yes	limited
Incident detection	limited	limited	yes	limited	limited	yes

[1] GPS is a proven technology. However, its applicability to travel time studies has been limited until recently.

Obviously, as accuracy increases so does cost, as shown in figure 1. With the floating car technique, only one probe vehicle is needed to characterize traffic flow along a specific direction of travel. As a result, it is sufficient to use GPS receivers having a positional accuracy of around 2-3 m. Including differential correction and a laptop computer for data storage, such receivers now cost less than $2,000. This makes GPS data collection price competitive with other techniques such as those based on the use of DMIs. Actually, GPS receivers are much more versatile than DMIs because they can also be used for purposes other than just data collection. Examples of GPS applications include sign inventories, fleet tracking, and origin-destination (O-D) studies. In contrast, DMIs are good for just one thing: measuring distances.

However, a data management problem still remains because of the huge number of records typically found in GPS data files. For example, collecting GPS data every one second means 3,600 position and speed records per hour. This number must then be multiplied by the number of hours a particular run takes to complete and also by the number of runs made on all routes considered. One approach to deal with this situation is simply to take the physical discontinuity attribute data recorded in the field with the GPS receiver and manually compute travel time information between specific checkpoints along the corridor. This approach is valid only when total travel time data is of interest and small-scale variations in traffic behavior along the corridor of interest are neglected. Obviously, only a handful of GPS data points are actually used in the analysis. Thus, it is reasonable to expect that results from this approach would closely resemble those from the traditional use of the floating car technique.

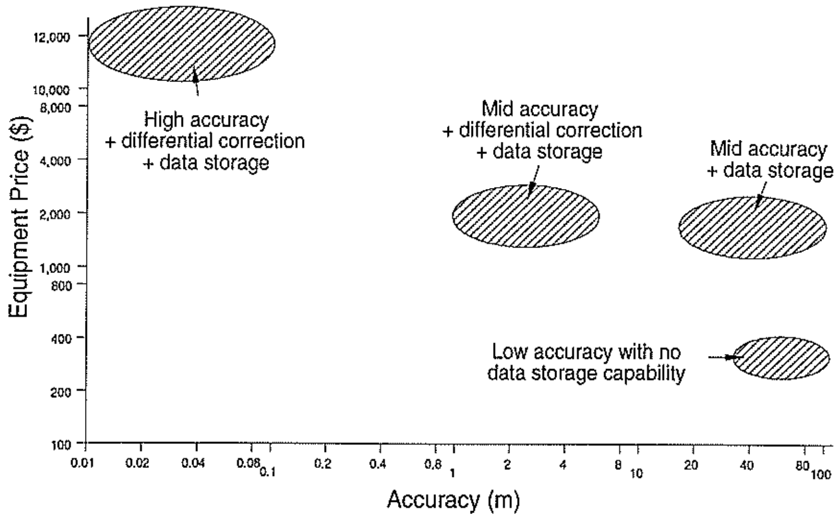


Figure 1
Relationship between GPS equipment cost and spatial accuracy (adapted from *[12]*)

A second approach involves keeping all GPS points and using a spreadsheet or a similar tool to provide a linear reference to all GPS points based on the coordinates of specific checkpoints *[13]*, *[14]*, *[15]*. This allows the construction of quite detailed speed-time or speed-distance profiles along corridors (figure 2). For visualization purposes, the procedure is usually augmented with the display of GPS points on maps containing corridors and other geographic features. In general, the argument is made that keeping all GPS points is important because a very detailed and rich picture of the traffic situation is readily available *[13]*. This would make the second approach suitable for detecting small-scale variations in traffic behavior. Such an argument is certainly powerful. However, it overlooks the fact that traffic may vary greatly from one day to another both in space and time. For example, at specific locations and time periods in the I-10, I-12 corridor in Baton Rouge it is not unusual to observe speed differences of 30 miles per hour (mph) or more from one day to another. This means that keeping all GPS data for analysis purposes may actually become counterproductive and that some kind of aggregation may be both desirable and necessary.

Base map preparation procedure

A good base vector map with linkages to a database is essential for conducting travel time studies using GPS and GIS. This base vector map could be obtained by using digitized quad maps or topologically integrating geographic encoding and referencing (TIGER) files. However, such maps only provide a very crude representation of the corridors and their surroundings. A simpler and much more accurate approach is to drive probe vehicles over all study routes in both directions collecting GPS data at regular time intervals, say every one second. During this field work drivers also survey on-ramps, off-ramps, interchanges, and signalized streets so that these discontinuities are included on the base map (figure 3a). The GPS data is then imported into a GIS map to create a directional centerline network map (figure 3b). By constructing this base map directly from GPS data, it can be guaranteed that the GPS data collected during future travel time studies will match the vector base map.

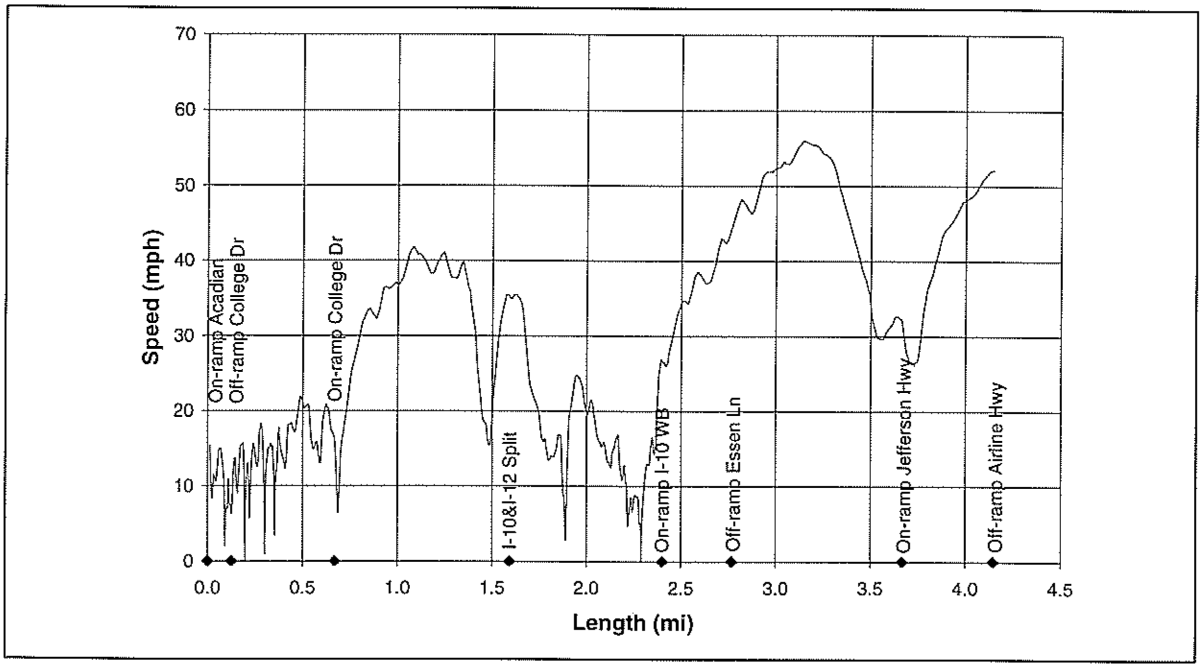


Figure 2
Speed-time profile using GPS data on the I-10, I-12 corridor east bound (EB) in Baton Rouge, Louisiana

Traditional travel time studies record travel time and average speed between checkpoints along the study route. Checkpoints in the GIS map can be formalized by using two simple rules. First, a checkpoint is established at all physical discontinuities such as signalized intersections, significant unsignalized intersections, lane drops, on-ramps, off-ramps, and other geometric discontinuities (figure 3c). Second, the section of road between physical discontinuity checkpoints is segmented so that there are nominally n checkpoints every mi. Figure 3d illustrates how the relatively large distances between exit and entrance ramps are segmented to create intermediate checkpoints. Finally, a procedure is followed to link each of the discrete segments to a relational database. This is done by assigning unique integer identifier numbers to each segment. For example, segment 12444 in figure 3e always represents the section of I-10 east bound (EB) immediately before the I-10, I-12 Split. Similarly, segment 12478 in figure 3e always represents the segment of I-12 EB immediately after the interchange from I-10 west bound (WB). By creating these unique identifiers, each segment can have fixed data such as number of lanes and posted speed limit associated with it.

It can also be used to index travel time data from travel time studies performed on different dates and times. These travel time studies would be conducted by driving a probe vehicle equipped with a GPS receiver and traversing the study routes during the morning peak , afternoon peak, and off peak periods.

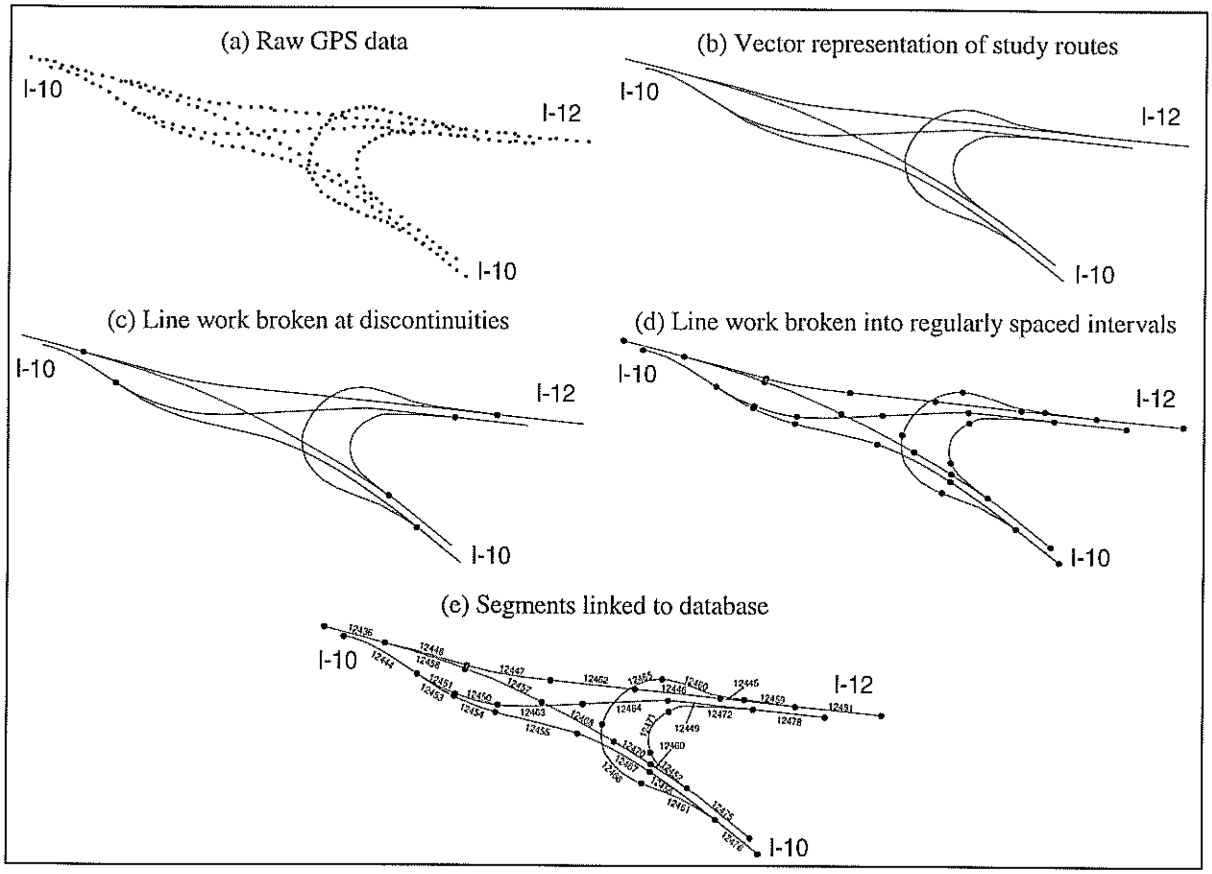


Figure 3
Sample network map geocoding and segmentation of the I-10, I-12 Split in Baton Rouge, Louisiana

Theoretically, the number of checkpoints per mi, n, can be set at any value. In practice, however, a careful balance must be established between the need for accuracy, the need to derive meaningful information from GPS-derived travel time data, and the need to develop a consistent procedure. In this study, two criteria were used for segmenting sections of road between physical discontinuity checkpoints:

- During the data reduction process, GPS travel time and speed data are converted into segment-wise travel time and speed information. To minimize the error associated with the new aggregated values, the original GPS travel times and speeds should be as uniform as possible. One way to accomplish this is by limiting the number of GPS points that can be linked to any particular segment. At the same time, however, the number of GPS samples per segment should also be large enough to ensure statistical significance. As shown in table 2, there is a trade off between segment length, probe vehicle speed, and number of GPS points that can associated with a segment. For the purposes of this study, 0.2 mi segments were used. This is equivalent to having five checkpoints per mi. Having 0.2 mi segments means that for a probe vehicle traveling at, say 35 mph, while collecting GPS data every one second, around 20 GPS travel time/speed records could be aggregated into one segment travel time/speed record. Obviously, other segment lengths are also possible. This issue is described in greater detail in appendix C.

Table 2
GPS points associated with a segment, assuming GPS data every one second

Segment length		Probe vehicle speed (mph)				
(mi)	(ft)	25	35	45	55	65
5.0	26,400	720	514	400	327	276
2.0	10,560	288	205	160	130	110
1.0	5,280	144	102	80	65	55
0.5	2,640	72	51	40	32	27
0.2	1,560	28	20	16	13	11
0.1	528	14	10	8	6	5
0.05	264	7	5	4	3	2
0.02	105.6	2	2	1	1	1
0.01	52.8	1	1	-	-	-

- Segment lengths should be as uniform as possible. This is particularly important as segments approach physical discontinuities. In this study, all segmentations began at a physical discontinuity and proceeded against the traffic flow using 0.2 mi segments until the next upstream physical discontinuity was encountered. In almost all cases, the total length between physical discontinuities was not an exact multiple of 0.2 mi. As a result, the last segment before the upstream physical discontinuity was less than 0.2 mi in length. To prevent the risk of not having any GPS data associated with segments that were too short (table 2), a minimum 0.1 mi length for the most upstream segment was specified. This required some changes in the segmentation procedure for the last one or two most upstream segments. Table 3 shows a few examples of the heuristics used in these cases.

In this study, all maps were generated using Intergraph's Modular GIS Environment (MGE) package. Appendix A describes the actual procedure used to create and segment the line work in MGE.

Travel time and speed

Once the directional centerline base map is developed, a set of runs is scheduled to measure travel time, speed and delay. The criteria to schedule runs may vary from case to case. For example, in Baton Rouge there is a clear distinction in traffic behavior between summer and academic year. Therefore, it would be necessary to schedule runs both in the summer and while schools are in session to make comparisons. In general, runs are made during the AM peak, PM peak, off peak, and other specific traffic conditions. For each time period, the number of runs (or sample size) must comply with acceptable error tolerance specifications. In this study, minimum sample sizes were estimated using the guidelines included in the Institute of Transportation Engineers (ITE) Manual of Transportation Engineering Studies *[4]*. The following assumptions were made:

- Confidence level: 95%
- Error tolerance (for planning studies): 5 mph
- Average range in running speed: 10 mph.

The resulting minimum sample size per time period was three.

Table 3
Rules for segmenting network into segments with a nominal length of 0.2 mi

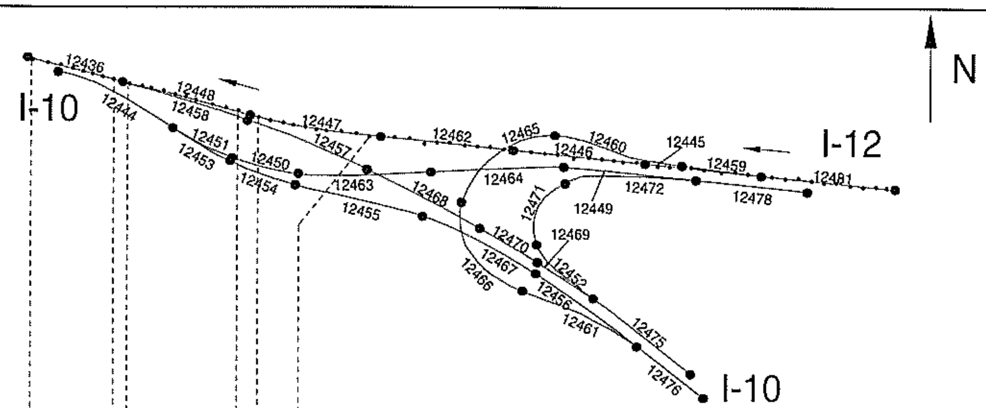
Total length to be segmented (mi)	Number of segments	Configuration (arrow indicates traffic flow direction)
0.0 ≤ L ≤ 0.2	1	L
0.2 < L < 0.3	2	0.5L, 0.5L
0.3 ≤ L ≤ 0.4	2	L-0.2, 0.2
0.4 < L < 0.5	3	$\frac{L-0.2}{2}$, $\frac{L-0.2}{2}$, 0.2
0.5 ≤ L ≤ 0.6	3	L-0.4, 0.2, 0.2
. .	. .	. .
. .	. .	. .
0.2(n-1) < L < 0.2(n-0.5)	n	$\frac{L-0.2(n-2)}{2}$, $\frac{L-0.2(n-2)}{2}$, 0.2 … 0.2 (n-2 segments)
0.2(n-0.5) ≤ L ≤ 0.2n	n	L-0.2(n-1), 0.2, 0.2 … 0.2 (n-1 segments)

For the past 20 years, many travel time studies have been planned and executed following the ITE guidelines. However, a recent evaluation of existing methodologies to estimate required sample sizes has demonstrated that the ITE guidelines seriously underestimates required sample sizes. An updated methodology was developed, and a paper was prepared and submitted to the ITE Journal for review and possible publication *[16]*. With the updated methodology, using the same numbers as above, the required sample size per time period would be six. In practice, this means that using a sample size lower than six should result in a confidence level lower than 95 percent. For example, if the actual sample size is 3, assuming the same error tolerance and average range values as above, the resulting confidence level would be around 75 percent.

Regardless of number of runs and scheduling specifications, standard procedures must be followed to ensure that the GPS data is consistent, that GPS files are properly handled and processed, and that ancillary data is adequately collected and processed. In the case of Baton Rouge, ancillary data was the incident information taped with the microcassette recorder. Obviously, specific data collection procedures are hardware dependent. As an example, however, appendix A lists the data collection procedure followed in Baton Rouge.

Data Reduction Methodology

Figure 4 shows an enlarged diagram of figure 3e with example GPS point data overlaid along a section of I-12 WB that merges with I-10 WB. The GPS Point Data table shows that the vehicle entered segment 12447 at 8:30:01 am, then segment 12448 at 8:30:11 am, and finally segment 12436 at 8:30:21 am. The Segment Aggregated Data table shows net travel times through segments 12447, 12448, and 12436. That table also tabulates the corresponding average vehicle speed values. It is these two values: travel time and average speed that are the important summary statistics.



GPS Point Data

Time (local)	Latitude (hh:mm:ss)	Longitude (hh:mm:ss)	Speed (mph)
8:30:01	30:25:09	-91:06:57	50.5
:	:	:	:
8:30:10	30:25:11	-91:07:09	48.5
8:30:11	30:25:11	-91:07:10	49.5
:	:	:	:
8:30:20	30:25:14	-91:07:20	52.7
8:30:21	30:25:14	-91:07:21	52.7
:	:	:	:
8:30:28	30:25:16	-91:07:28	54.5

⇒

Segment Aggregated Data

Segment	Travel time (s)	Average speed (mph)
12447	10	49.2
12448	10	51.4
12436	8	54.0

Figure 4
GPS data mapping onto highway segments

To efficiently transform the GPS point data into segment travel times and average speeds, a procedure was developed to filter and reduce this data so that only summary information could be written to the database. First, short filtering programs in C++ were written to filter out data that was not strictly geographic coordinates, time, speed, and differential correction status. Examples of data filtered out included satellite navigational data, communication status, and other messages. The source codes of these programs are included in appendix B. Second, an application was written in Microstation Developing Language (MDL) to interactively link each segment with the filtered GPS data (figure 5). The source code of this application is also included in appendix B. As shown in figure 5a, when a segment is selected, the corresponding GPS points are displayed on the screen. Since the GPS location was recorded every one second, each point shown on the map is spaced one second from the previous point. The form shown in figure 5b displays basic segment information including segment code, name, and length (in mi), and travel information including starting time, ending time, travel time, and speed. Time is given in seconds, and speed is given in mph. The starting time is the time the probe vehicle enters the segment. The ending time is the time the probe vehicle leaves the segment. The difference between the two times is the travel time. Two values of speed are given: the average speed results from dividing the segment length by the travel time. The GPS speed results from averaging the instantaneous speed values given for all GPS points associated with the segment. The ratio between the two speed values computed is an indication of how uniform speed is kept within the segment. Details of the mathematical model used for these computations are included in appendix C.

Operators of the interactive data reduction application were responsible for loading the GPS data files and for clicking on segments (figure 5a) so the MDL application could determine segment entrance time, segment exit time, and average segment speed. For example, when the user clicked on segment 12444, the corresponding travel time and average speed were computed and shown in the user interface (figure 5b). The application also displayed a number of points ahead so that the user knew that the next segment to click on was segment 12451. Successive segments were selected by the operator in a similar fashion. Operators were typically trained in one to two hours and were proficient within 10 hours. They typically reduced data about six to eight times faster than when it was collected in the probe vehicle. For example, if two hours of GPS data were collected (for a total of 7,200 records), they reduced this data to a table of segment identifiers, travel time, and average speed in about 15-20 minutes. More automation of the reduction application procedure could reduce processing times even further. Eventually, the application might be able to select the segments automatically without any assistance from the operator in which case the data reduction process would be expected to take only a few seconds.

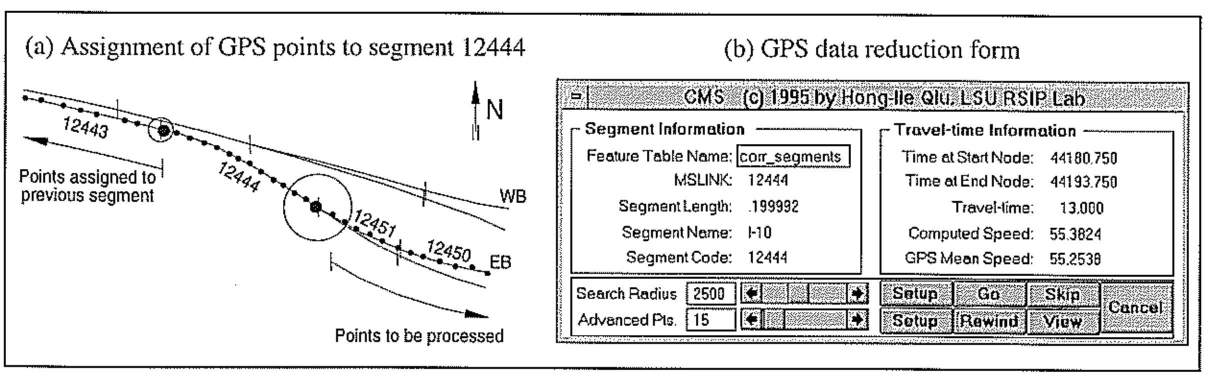


Figure 5
Graphical interface used for data reduction

Geographic database schema

A geographic database was developed to store and process GPS-derived data. All maps were processed using Intergraph's MGE on a Windows NT workstation, with Oracle v.7 as the underlying relational database package. The database schema is composed of five attribute tables, as shown in figure 6: CORR_SEGMENTS, CORR_NAMES, FUNCT_TYPES, SEG_TYPES and SEG_ TRAVEL_TIME. A summary of the associated attributes is shown in table 4. A sample of records is shown in figure 7.

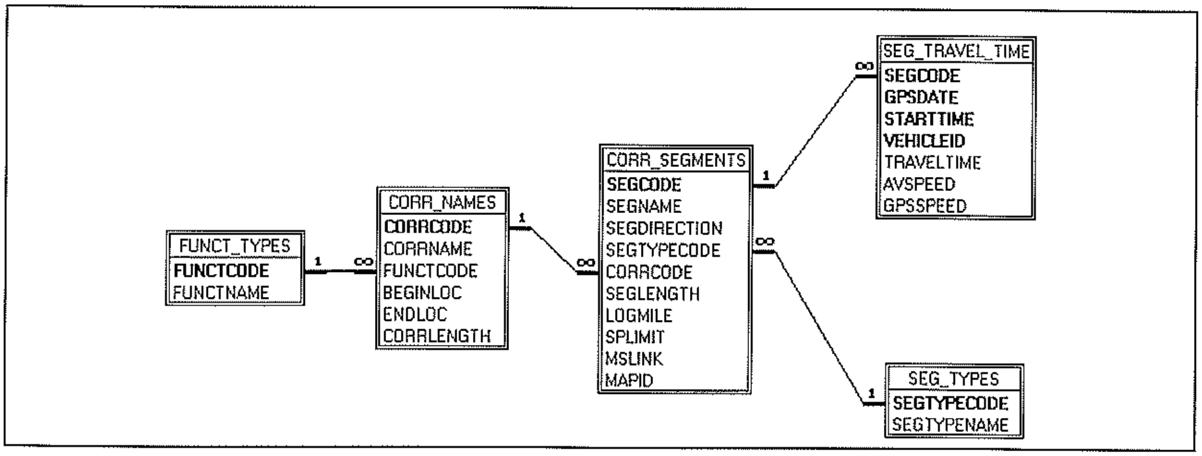


Figure 6
Geographic database schema

A short description of the tables shown in figure 6 follows:

1. CORR_SEGMENTS: This table contains basic data about each segment, including a unique segment code, name, direction, type, length, posted speed limit, and internal linkage to the database. Note that there are two references to the corridor: in the segment name field, and in the corridor code field. In most cases, the segment has the same name as the corridor. However, some roads that change name but are part of a defacto corridor can be studied as a group using the corridor code field.
2. SEG_TYPES: This table is a lookup table that contains the linkage between segment type codes included in table CORR_SEGMENTS and the corresponding segment type names. Seven segment types have been defined: main, interchange, on-ramp, off-ramp, service road, short link, and other.
3. CORR_NAMES: This table contains basic information about the corridors, including corridor name, beginning and ending points, and length. It also contains a field used to define the corridor functional class. Length is actually a derived field based on the average of the cumulative lengths of all segments on both directions of travel along the corridor.
4. FUNCT_TYPES: This table is a lookup table that contains the linkage between the corridor functional class codes included in table CORR_NAMES and the corresponding functional classes. Two functional classes have been defined: Interstate, and Principal Arterial.

Table 4
Description of geographic database table fields

Table	Attribute	Key	Description
CORR_SEGMENTS	SEGCODE	X	Segment code: has the same value as MSLINK
	SEGNAME		Segment name
	SEGDIRECTION		Segment direction: EB, WB, NB, SB
	SEGTYPECODE		Segment type code: 1, 2, 3, 4, 5, 6, or 7
	CORRCODE		Corridor code to which segment belongs
	SEGLENGTH		Length of segment, measured graphically with GIS
	LOGMILE		Cumulative length along corridor
	SPLIMIT		Posted speed limit (mph)
	MSLINK		Linkage between graphical element and attribute table; created by MGE
	MAPID		Map identification number: created automatically by MGE
SEG_TYPES	SEGTYPECODE	X	Segment type code: 1, 2, 3, 4, 5, 6, or 7
	SEGTYPENAME		Segment type name equivalent to SEGTYPECODE
CORR_NAMES	CORRCODE	X	Corridor code
	CORRNAME		Corridor name
	FUNCTCODE		Corridor function class code: 1 or 2
	BEGINLOC		Location where corridor begins
	ENDLOC		Location where corridor ends
	CORRLENGTH		Corridor length
FUNCT_TYPES	FUNCTCODE	X	Corridor functional class code: 1 or 2
	FUNCTNAME		Corridor functional name equivalent to FUNCTCODE:
SEG_TRAVEL_TIME	SEGCODE	X	Segment code
	GPSDATE	X	Date GPS data was collected
	STARTTIME	X	GMT time associated with beginning of segment
	VEHICLEID	X	Probe vehicle ID number
	TRAVELTIME		Time in seconds probe vehicle takes to survey segment length
	AVSPEED		Average speed of probe vehicle (segment length/travel time): in mph
	GPSSPEED		Speed that results from averaging GPS point instantaneous speed values

Table CORR_SEGMENTS

SEG CODE	SEG NAME	SEGDI RECTION	SEGTYPE CODE	CORR CODE	SEG LENGTH	LOG MILE	SP LIMIT	MSLINK	MAPID
12444	I-10	EB	1	1	0.200	5.00	55	12444	21
12451	I-12	EB	1	3	0.104	0.00	55	12451	21
12450	I-12	EB	1	3	0.104	0.10	55	12450	21

Table SEG_TYPES

SEGTYPE CODE	SEGTYPE NAME
1	main
2	interchange
3	on-ramp

Table CORR_NAMES

CORR CODE	CORR NAME	FUNCT CODE	BEGINLOC	ENDLOC	CORR LENGTH
1	I-10	1	W. study boundary	Ascension Parish boundary line	18.21
2	I-110	1	I-10 intersection	Scenic Hwy	8.84
3	I-12	1	I-10 I-12 Split	E. study boundary	17.51

Table FUNCT_TYPES

FUNCT CODE	FUNCTNAME
1	Interstate
2	Principal arterial

Table SEG_TRAVEL_TIME

SEGCODE	GPSDATE	STARTTIME	VEHICLEID	TRAVELTIME	AVSPEED	GPSSPEED
12444	25-JUL-95	49556.75	2	11.5	62.61	56.50
12451	25-JUL-95	49568.25	2	7.0	53.38	57.30
12450	25-JUL-95	49574.75	2	7.0	53.38	57.30
12463	25-JUL-95	49581.75	2	12.5	57.60	58.78

Figure 7
Sample of records from the database

5. SEG_TRAVEL_TIME: This table contains summarized GPS-derived traffic related data. For each segment code, date, time, and vehicle ID, it stores travel time and average speed. For convenience, time in this table is expressed in seconds Universal Coordinated Time (UTC). Note that this is the table containing the bulk of the data: 155,300 records for Baton Rouge; 5,048 records for Shreveport; and 22,613 records for New Orleans.

Database queries

Two types of queries can be made: spatial and non-spatial queries. Spatial queries depend on segment location and require GIS tools for creating and executing the queries. Examples of spatial queries are some of the queries needed to generate color coded maps. In contrast, non-spatial queries do not depend on the actual location of a segment and, as a result, they can be created and executed outside the GIS environment. Examples of non-spatial queries are queries used to determine number of records per segment, or average speed values per time period.

While the number of queries that can be generated from the database is enormous, a few number of queries seem to be needed quite frequently. Some of the queries that fall into this category are the following (see appendix D for a detailed description):

- Selection of records associated with a specific segment
- Computation of segment minimum, average, and maximum speed and travel time per date range and per time period
- Computation of segment median speed per date range and per time period
- Determination of free flow speeds
- Computation of segment travel time delay
- Computation of speed and travel time at the corridor level

Audio recorder log schema

A database was developed in Microsoft's Access to store data collected with the microcassette recorder. Figure 8 shows the data entry form used to transcribe the tapes. The database contains summarized audio recorded information, including vehicle ID, date, time, route name, direction, location, and specific comment description. For statistical purposes, comments were classified into seven types: weather, lighting, traffic conditions, route incident, other incident, probe vehicle, and other.

Data Reporting Methodology

Once the data is reduced to the relational database format shown in figure 7, it becomes important to develop efficient reporting procedures. One obvious approach is to draw speed-distance or speed-time profiles along the corridor of interest, following the traditional practice of travel time studies. However, use of such charts can be considerably augmented by the use of GIS and database querying and reporting tools. Three reporting options are considered here: Color coded maps; archival tabular reports; and World Wide Web (WWW) reports.

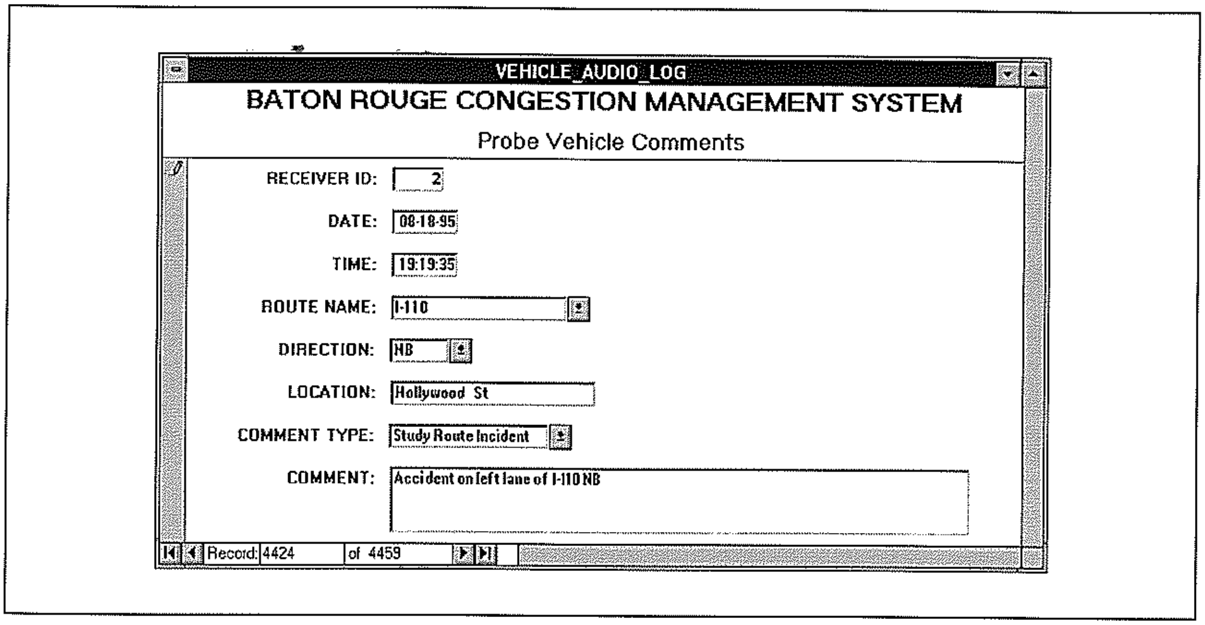


Figure 8
Audio recorder log data entry form

Color coded maps

Figure 9 shows an example gray scale shaded map corresponding to the average observed speed from 4:30 pm to 5:30 pm during the 1995-1996 academic year in Baton Rouge. Obviously, the query used to generate the map could be modified to include different time periods or show other speed values such as median speeds, minimum speeds, and free flow speeds. For illustration purposes, figure 9 shows only a very small portion of the network. However, in practice, large format color plots covering a 10x10 mi^2 area are typically plotted. These maps are particularly effective for explaining congestion problems at public meetings. The detailed procedure to produce the maps is included in appendix E.

Archival tabular reports

For archival and analytical purposes, a second reporting format similar to the one used to document highway features such as sign posts and culverts was implemented. These reports are produced on 11x17 paper and cover 20 highway segments in each direction of travel. A sample report page is shown in figure 10. The data tabulated above and below the "strip map" are average speeds and average cumulative travel time for both directions of travel during the AM peak, off peak, and PM peak periods. The shading next to the average segment speed provides visual indication of the ratio of observed speed to posted speed limit so that problem areas are readily apparent. The procedure to produce these pages has been automated by a series of Microsoft's Access macros that execute the necessary queries on the travel time table and format the report page. A detailed description of the structure and query schema used for generating these report pages is included in appendix E. In a typical application, the database operator would be asked to select a range of dates, the report page(s) of interest, and the printing date. The application automatically formats the report page and sends the results to the printer.

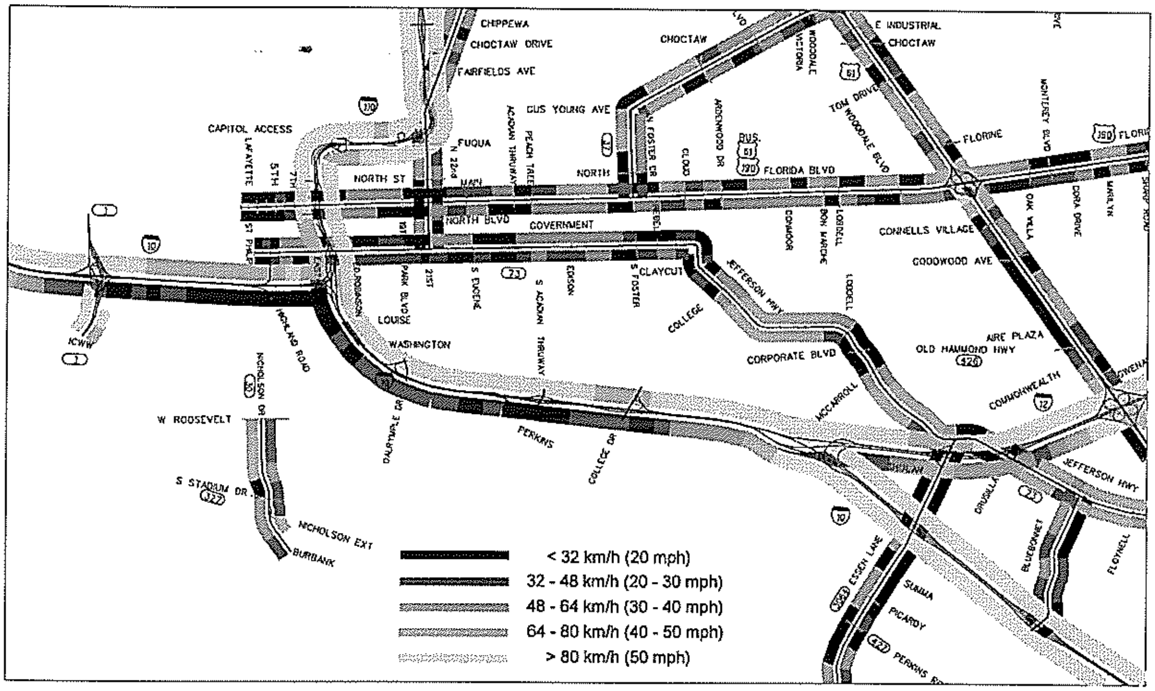
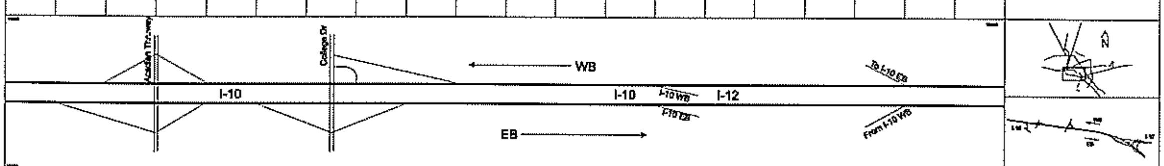


Figure 9

PM peak (4:30-5:30 pm) average speeds during the 1995-1996 academic year in Baton Rouge

BATON ROUGE — CORRIDOR No. 1 – I-10 — 9/1/95 to 5/3/96

																				PM PEAK
53	51	54	54	50	47	48	48	55	55	54	55	55	57	56	57	56	55	52	47	Avg. Travel Speed (mph)
233	224	216	203	191	177	162	153	144	135	122	108	95	86	74	61	48	35	24	10	Cum. Travel Time (sec)
																				OFF PEAK
53	52	56	56	50	54	51	52	55	55	55	55	52	42	60	59	59	59	57	57	Avg. Travel Speed (mph)
227	219	211	198	187	172	159	151	142	134	120	107	94	84	67	55	43	31	20	8	Cum. Travel Time (sec)
																				AM PEAK
43	40	49	50	44	43	39	38	36	37	38	39	40	35	37	39	47	49	41	28	Avg. Travel Speed (mph)
306	296	285	271	258	242	225	215	202	190	170	151	132	119	98	79	61	46	32	15	Cum. Travel Time (sec)
55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	Posted Speed Limit (mph)
0.12	0.12	0.20	0.18	0.20	0.20	0.11	0.13	0.13	0.20	0.20	0.20	0.15	0.20	0.20	0.20	0.20	0.18	0.20	0.12	Segment Length (mi)
14262	12412	12421	12422	12423	12428	12423	12435	12434	12439	12438	12437	12436	12448	12447	12462	12446	12445	12481	12480	Segment ID



12139	12414	12420	12415	11810	12426	12429	12427	12440	12441	12442	12443	12444	12451	12450	12463	12464	12449	12478	12479	Segment ID
0.15	0.14	0.20	0.20	0.12	0.14	0.20	0.20	0.12	0.20	0.20	0.20	0.20	0.10	0.10	0.20	0.20	0.20	0.17	0.20	Segment Length (mi)
55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	Posted Speed Limit (mph)
																				AM PEAK
51	52	53	52	51	53	52	53	52	53	53	53	53	52	53	54	54	53	54	55	Avg. Travel Speed (mph)
10	19	33	47	56	66	79	93	102	115	129	143	156	163	170	184	197	210	222	235	Cum. Travel Time (sec)
																				OFF PEAK
51	56	55	53	50	54	52	52	50	53	54	54	55	57	58	62	63	63	61	61	Avg. Travel Speed (mph)
10	19	33	47	55	65	79	93	101	115	128	142	155	161	167	179	190	202	213	224	Cum. Travel Time (sec)
																				PM PEAK
21	14	12	13	22	23	22	20	23	33	37	39	41	31	17	20	21	16	22	35	Avg. Travel Speed (mph)
26	61	118	173	193	215	249	284	304	326	345	364	382	393	415	451	486	531	559	580	Cum. Travel Time (sec)

09/17/96	Remote Sensing and Image Processing Laboratory	Avg Travel Speed/Posted Speed Limit Ratio: >0.9, 0.8-0.9, 0.7-0.8, 0.5-0.7, 0.3-0.5, <0.3	Section 11

Figure 10

Example report page for the I-10, I-12 Corridor during the 1995-1996 academic year in Baton Rouge

WWW reports

Several web pages were developed at the Remote Sensing and Image Processing Laboratory (RSIP) at Louisiana State University (LSU) to make travel time data available using Internet resources. A short description of the structure of the main web pages and a corresponding example follows.

CMS segment travel time data

The WWW web page that allows users to make queries related to the Baton Rouge CMS is located at http://www.rsip.lsu.edu/cms/cmsbtr/cms-query.html. A network database link was implemented so that users could execute queries by segment and download the corresponding data into their computers. Currently, the system allows users to select a segment either by clicking on a set of sensitive maps until finding the specific segment of interest (figure 11) or by typing in the segment code in a text field. The system then displays all existing records associated with that segment and produces summary speed data on an hourly basis.

As shown in figure 12, there are two computer systems and two software components that comprise the CMS-WWW query interface. The Windows NT based system running Oracle v.7 acts as the database server. The Sun workstation is the server for WWW accesses. When the user on a remote computer clicks on a segment (for example segment 12444 in figure 11c), the WWW server executes a Practical Extraction and Report Language (PERL) script *[17]*. This script calls a C program to query the Structured Query Language (SQL) database, formats the SQL query results in Hypertext Markup Language (HTML) format, and passes this data back to the WWW browser. The C program uses routines from the Oracle server query library to send SQL queries across the network to the Oracle server and retrieve the query results. For example, if the user clicked on segment 12444 (figure 11c), the WWW server would pass the segment code 12444 to the PERL script which would, in turn, pass 12444 to the C program. This program would then construct the following query:

```
SELECT    SEGCODE, GPSDATE, STARTTIME, GPSSPEED
FROM      SEG_TRAVEL_TIME
WHERE     SEGCODE = 12444;
```

and transmit that query across the network to the Oracle database. When the query results are returned, the PERL script would insert the necessary HTML commands so the results are presented in a color coded tabular format similar to the data shown in figure 13. The source codes of the PERL script and the C program are included in appendix E.

A second example of an SQL command which is executed to determine the total number of records in the database is:

```
SELECT    COUNT (*)
FROM      SEG_TRAVEL_TIME;
```

In this case, no arguments are required since this is simply gathering summary information about the database and is not collecting information about a particular subgroup of records.

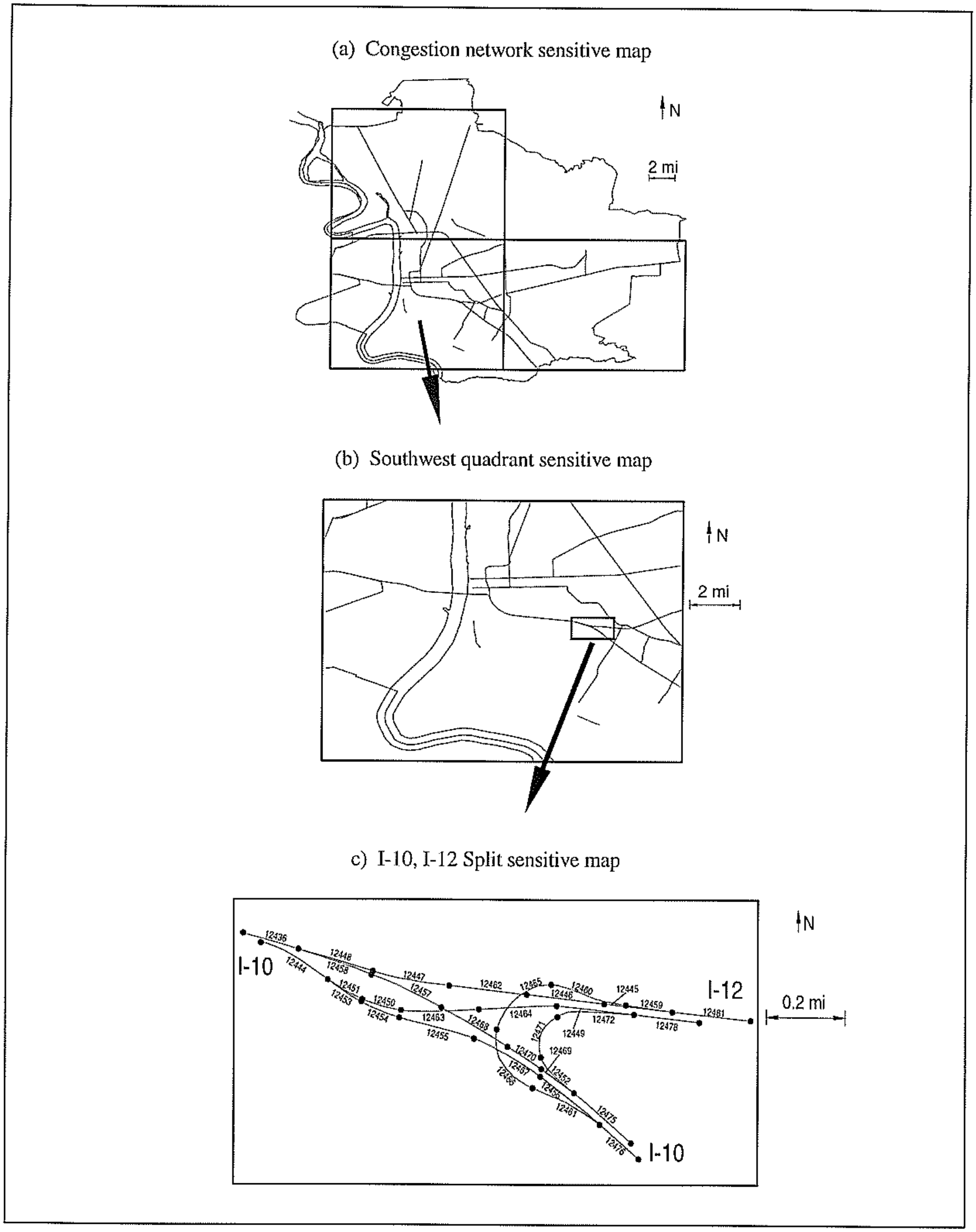


Figure 11
Congestion network and segmentation at the I-10, I-12 Split in Baton Rouge, Louisiana

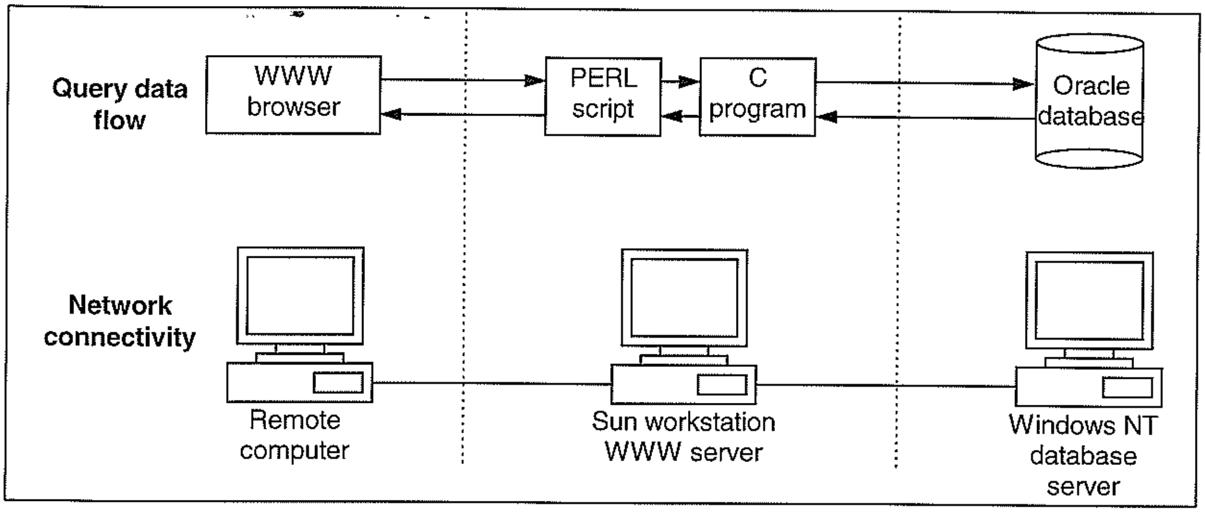


Figure 12
Computer and software components of the CMS-WWW query interface

Sample of query results

SEGCODE	GPSDATE	STARTTIME	GPSSPEED
...	...	...	...
12444	15-AUG-95	6:23:22.25	49.2
12444	15-AUG-95	7:08:33.75	55.6
12444	15-AUG-95	7:56:45.25	57.2
...	...	...	...
12444	27-NOV-95	15:38:19.25	56.5
12444	27-NOV-95	16:10:25.75	61.0
12444	27-NOV-95	16:41:45.25	49.5
...	...	...	...
12444	20-DEC-95	10:24:51.41	58.4
12444	20-DEC-95	10:46:31.37	55.5

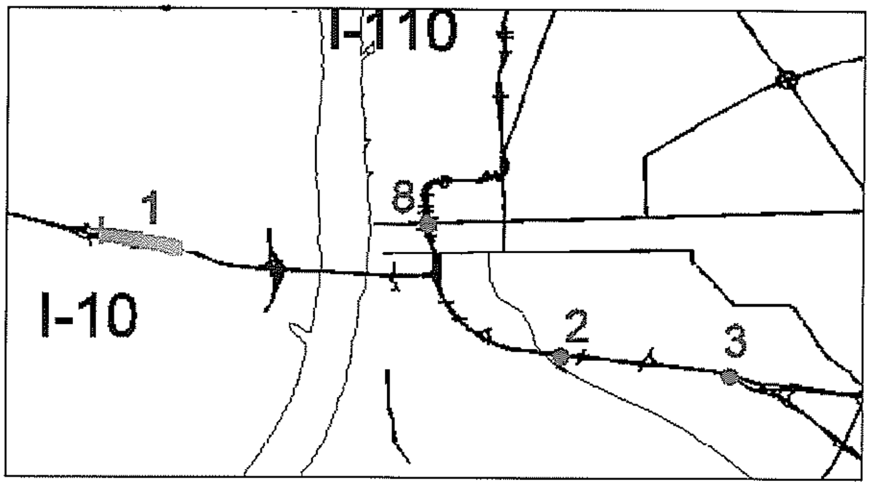
Query summary

Time interval (h)	Average speed (mph)		# records
5-6	58.0		1
6-7	54.4		69
7-8	54.0		46
8-9	53.3		22
9-10	53.6		10
10-11	55.2		5
11-12	55.3		6
12-13	55.8		4
13-14	51.7		12
14-15	50.2		20
15-16	49.1		22
16-17	40.3		39
17-18	35.9		23

Figure 13
Sample of query results and query summary from the CMS web page

Real-time travel time data

A WWW web page was developed to display real-time travel time data at specific checkpoints along the I-10, I-12 corridor. The objective of this web page was to demonstrate the feasibility of using real-time GPS data for monitoring traffic conditions on the Interstate highways in Baton Rouge. This web page (located at http://www.rsip.lsu.edu/cms/cmsbtr/ hwyckpoints.html) displayed a map of the I-10, I-12 corridor with dots representing the last 10 vehicle locations retrieved from the field via radio and a table showing the corresponding time stamp and probe vehicle speed values (figure 14). The map was updated every two minutes. When the probe vehicle crossed a checkpoint, the system determined the time stamp associated with it and calculated interval and cumulative travel time data. For comparison purposes, it also displayed the travel time history for the previous run along the same corridor.



Current probe vehicle speed

Date	Time (CDT)	Speed (mph)
7/12/96	17:12:27	28.2
7/12/96	17:12:32	23.1
7/12/96	17:12:42	30.9
7/12/96	17:12:53	36.9
7/12/96	17:13:04	37.0
7/12/96	17:13:17	39.1
7/12/96	17:13:23	42.0
7/12/96	17:13:32	42.9
7/12/96	17:13:42	40.7
7/12/96	17:13:52	38.1

Current trip travel time history

Checkpoint	6WB	5WB	3WB	2WB	1WB
Time (CDT)	16:39:33	16:46:13	16:50:03	16:53:17	17:13:32
Interval travel time	0	6:40	3:50	3:14	20:15
Cum. travel time	0	6:40	10:30	13:44	33:59

Previous trip travel time history

Checkpoint	6WB	5WB	3WB	2WB	1WB
Time (CDT)	8:41:41	8:48:21	8:50:15	8:52:40	9:00:05
Interval travel time	0	6:40	1:54	2:25	7:25
Cum. travel time	0	6:40	8:34	10:59	18:24

Figure 14
Real-time travel time data on the I-10, I-12 corridor in Baton Rouge

The real-time travel time data system had two components. The first component was a radio monitoring system (figure 15). This component consisted of a program that ran on a PC and queried each vehicle for position information every 10 seconds. The second component was a WWW information update system (figure 15). This component consisted of a C program that was executed at fixed intervals to process the radio information and update the tables and graphics for the web pages. This program was executed every two minutes. The program performed several tasks. First, the program updated the table containing information about the latest 10 vehicle locations. The map which showed these locations was also updated (figure 14). Second, the program determined if any checkpoints had been crossed and updated the table containing this checkpoint information. Finally, the program calculated the most recent segment travel times for a selected set of segments and produced a table containing this information (figure 14). The source code of the C program is included in appendix E.

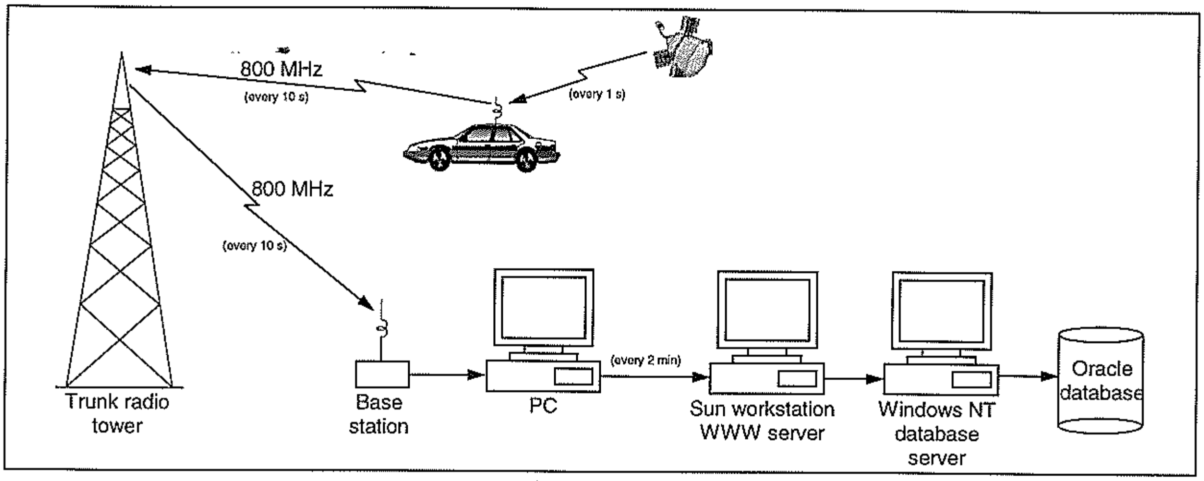


Figure 15
Real-time travel time data transmission and storage

Two additional web pages were developed to display real-time travel time data. These web pages were used to track the operation of a city bus circulating in the vicinity of LSU in Baton Rouge. The web pages are located at http://www.rsip.lsu.edu/cms/bus/bus.shtml and http://www. rsip.lsu.edu/cms/bus/busphoto/busphoto.shtml. The first web page displayed a map around the LSU campus in Baton Rouge with dots representing the latest GPS points retrieved from the field via radio, and also contained a table showing the corresponding time stamp and instantaneous speed values. The second web page also showed dots representing the latest GPS points, but instead of displaying a map on the background, it displayed an aerial photograph. The source code of the programs used to update these home pages is almost identical. For simplicity, only the source code for web page http://www.rsip.lsu.edu/cms/bus/ bus.shtml is included in appendix E.

DISCUSSION OF RESULTS

This section describes the implementation of the data collection, data reduction, and data reporting procedures described previously to three case studies in Louisiana: Baton Rouge, Shreveport, and New Orleans.

Baton Rouge

The Baton Rouge MPO, the Capital Region Planning Commission (CRPC), defined a congestion corridor network composed of 22 corridors covering 151 mi (figure 16). As shown in table 5, of the 22 corridors, three were located on the Interstate highway system and covered 45 mi. The remaining 19 corridors were located on principal arterials and covered 106 mi. The RSIP Laboratory was responsible for all activities including data collection, development of base vector map, data reduction, and data reporting. A description of each of these activities is provided in the following paragraphs.

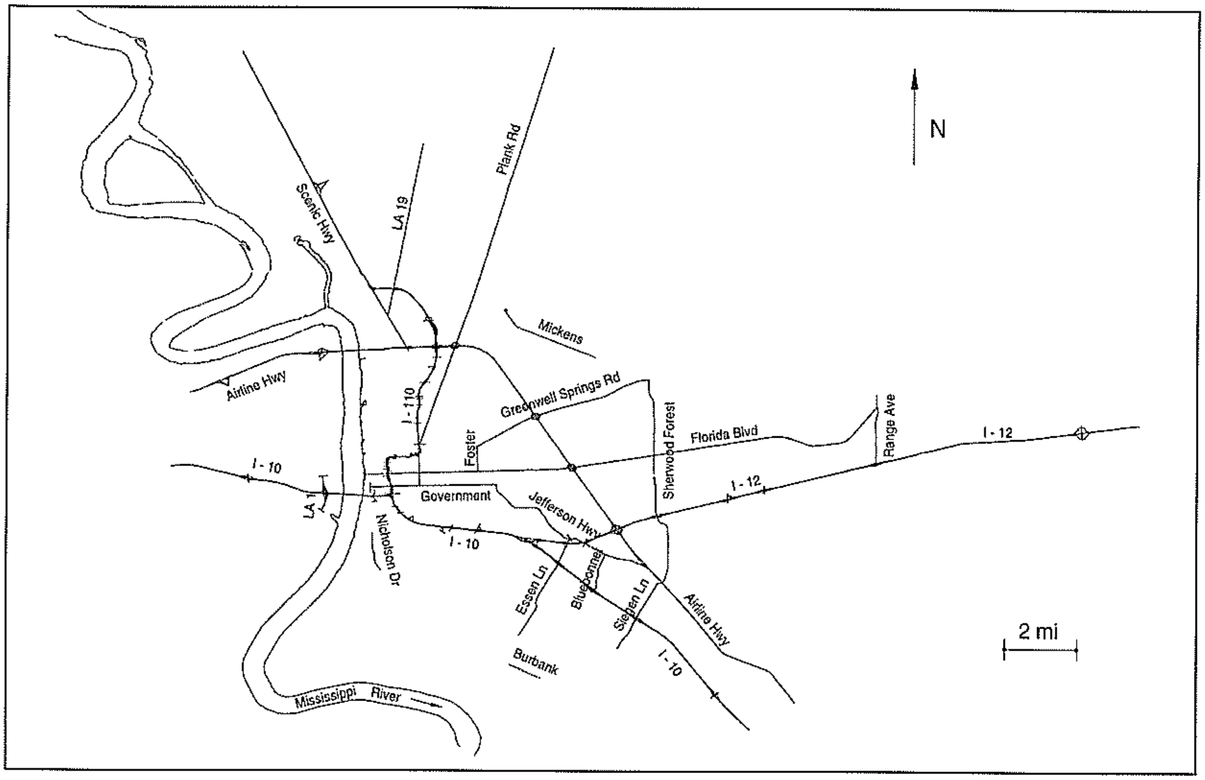


Figure 16
Congestion corridor network in Baton Rouge

Data collection

Equipment

Figure 17 illustrates the equipment used to monitor traffic conditions and the connectivity among components. The GPS receiver was a Trimble Placer GPS 400 unit. This unit collected data from the constellation of satellites and determined coordinates (in latitude, longitude), time (in Universal Coordinated Time - UTC), speed (in mph), and other ancillary data. To bypass the intentional signal degradation produced by selective availability (SA) *[11]*, an RDS 3000 unit was also installed. This unit collected differential correction data provided through a commercial FM subcarrier. The actual differential correction took place inside the GPS 400 receiver by processing the uncorrected satellite data and the FM correction data at the same time. The differentially corrected GPS (DGPS) data was then stored in an ASCII file in the laptop computer. The resulting positional accuracy of the DGPS data was two to five meters spherical error probability (SEP) *[18]*.

Table 5
Congestion corridor network in Baton Rouge

ID	Name	From	To	Dir.	Functional class	Length (mi)	
1	I-10	2 mi West Of LA 415	Ascension Parish line	E-W	Interstate	19.06	
2	I-110	I-10, I-110 Split	Scenic Hwy	N-S	Interstate	8.71	
3	I-12	I-10 &-12 Split	1.5 mi East of LA 447	E-W	Interstate	17.59	
4	La 19	Scenic Hwy	Wimbush Lane	N-S	Princ. Arterial		5.62
5	Plank Rd	Government St	LA 64	N-S	Princ. Arterial		14.64
6	Airline Hwy	LA 1145	2 mi East of Highland Rd	N-S	Princ. Arterial		22.87
7	Florida Blvd	River Front	Range Ave	E-W	Princ. Arterial		14.69
8	Mickens Rd	Hooper Rd	Joor Rd	E-W	Princ. Arterial		3.01
9	Sherwood Forest Blvd	Greenwell Springs Rd	Airline Hwy	N-S	Princ. Arterial		6.75
10	Siegen Lane	Airline Hwy	Perkins Rd	N-S	Princ. Arterial		2.47
11	N Foster Dr	Florida Blvd	Greenwell Springs Rd	N-S	Princ. Arterial		0.78
12	Government St	River Front	Jefferson Hwy	E-W	Princ. Arterial		3.44
13	Jefferson Hwy	Airline Hwy	Government St	E-W	Princ. Arterial		5.41
14	Staring Lane	Perkins Rd	Highland Rd	N-S	Princ. Arterial		1.99
15	Essen Lane	Perkins Rd	Jefferson Hwy	N-S	Princ. Arterial		1.86
16	Bluebonnet Rd	Jefferson Hwy	I-10	N-S	Princ. Arterial		1.31
17	Burbank Dr	Gardere Ln	Bluebonnet Rd	E-W	Princ. Arterial		0.92
18	Nicholson Dr	Roosevelt St	Burbank Dr	N-S	Princ. Arterial		1.25
19	LA 1	I-10	ICWW	N-S	Princ. Arterial		0.42
20	Greenwell Springs Rd	N Foster Dr	Sherwood Forest Blvd	E-W	Princ. Arterial		5.24
21	Scenic Hwy	Airline Hwy	LA 64	N-S	Princ. Arterial		10.67
22	Range Ave	Florida Blvd	I-12	N-S	Princ. Arterial		2.29
			TOTAL			**45.36**	**105.63**

Also part of the equipment was a microcassette audio tape recorder used to document changing weather conditions, disruptive traffic incidents, and queues at major intersections. The use of the device eliminated the need to make written notes while driving. Incidents on highways throughout the city were also recorded by using a scanner to monitor the Police band and by listening to the traffic reports produced by one of the local radio stations.

The configuration shown in figure 17 was appropriate for monitoring traffic in an off-line mode. In this mode, once the run was completed, the equipment was brought back to the office and the ASCII file was downloaded to one of the network drives, where it remained until processing, reduction, and transfer to the database were completed at a later time. For real-time traffic monitoring, the laptop computer was replaced by an 800 MHz radio transmitter which sent the GPS signal to a base station back at the office (figure 15). As explained previously, the base station was connected to a PC computer which ran a program to process the radio information and updated tables and graphics in a web page in almost real-time.

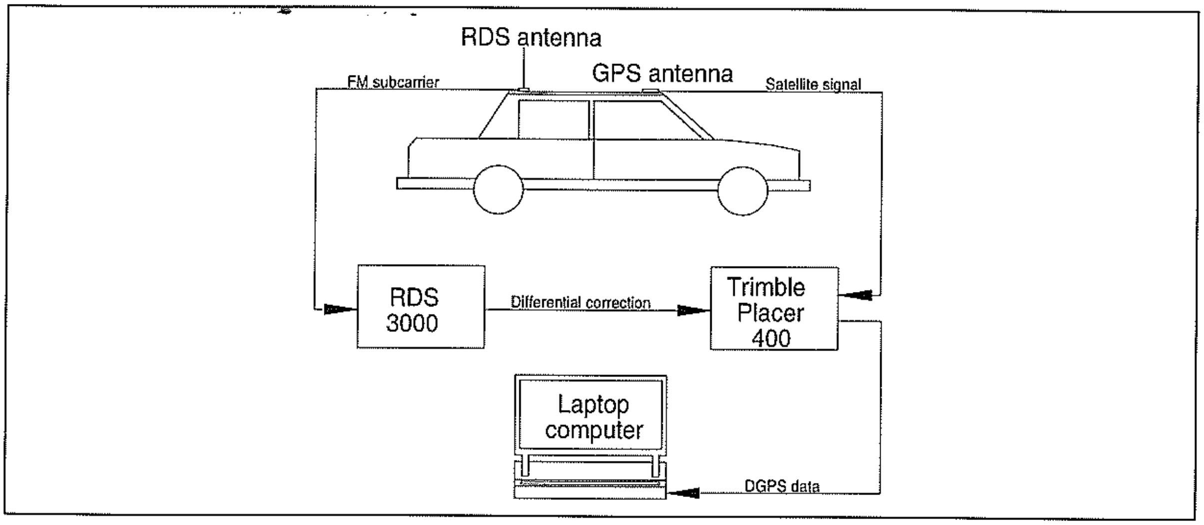


Figure 17
GPS equipment, laptop computer, and probe vehicle configuration

Base vector map

Differentially corrected GPS (DGPS) points, recorded every one second, were used to construct the centerline base map. For each direction of traffic, at least two GPS runs were made: one for the far right lane, and the other one for the far left lane. Physical discontinuities such as interchanges, on-ramps, off-ramps, and signalized intersecting streets were also surveyed. The resulting DGPS points were imported into an Intergraph's MGE design file to create linear graphical elements. A procedure was then developed to eliminate as many redundant points as possible without jeopardizing positional accuracy. Initially, the MGE line weeding algorithm was used for this purpose. However, exclusive use of this algorithm was not suitable because of GPS data stochastic behavior, despite differential correction. This was particularly evident at signalized intersections, where the probe vehicle frequently had to stop, or where bridges, buildings, and tree canopies caused signal degradation. It was then necessary to augment the weeding process by manually editing the line work with the GPS data in the background. While the resulting line work did not represent a non-biased estimate for the location of the corridor axes, given the low number of samples, this procedure was very quick and provided a workable base map with sufficient accuracy for this study.

Each corridor directional centerline was then partitioned using 0.2 mi segments, following the procedure already described before. A total of 2,397 segments, covering 369 mi of directional centerlines were defined. A total of 1,852 of these segments were located along the main routes. The remaining 545 segments were located on interchanges, on-ramps, off-ramps and intersecting streets.

Routes and driving schedules

Once the corridor network was surveyed and the directional centerline base map developed, a set of runs were scheduled to measure travel time and speed. Runs were made from May 1995 to July 1996. As a result, records for Summer 1995, academic year 1995-1996, and Summer 1996 were gathered. Because the objective was to measure typical traffic congestion conditions, runs were not made for three weeks at the beginning of the academic year to allow the network to stabilize. For a similar reason, runs were not scheduled during academic breaks including Christmas-New Year and Spring breaks. Runs were made during three traffic time periods: weekday morning (6:00 to 9:00 am), weekday afternoon (3:00 to 6:00 pm), and off peak (9:00 to 3:00 pm). Off peak runs also included runs on selected Sunday mornings to obtain free flow speeds.

Data reduction

A summary of dates, file names, file sizes, and differential correction and reduction percentages is included in appendix F. A total of 25,000 mi of travel runs on the 151 mi highway network were made, resulting in 428 GPS data files and 2.5 million GPS point records. The data reduction GIS application described previously was used to process these records, resulting in 155,300 segment records. On average, there were about 65 records per segment. Obviously, some corridors and segments ended up with more records than others, as shown in table 6. Between May 1995 and May 1996, runs were made on all corridors, although efforts were concentrated on routes that were perceived to have the most severe congestion problems. After May 1996, runs were limited to the Interstate routes in order to collect data for developing the real-time WWW travel time data dissemination application. Notice that only records and number of segments associated with segments located directly on the main routes are listed separately. All non-main route segments such as interchanges and ramps, as well as their corresponding records, are lumped together.

Data reporting

As mentioned before, three reporting options were considered for displaying segment travel time data in Baton Rouge: Color coded maps, archival tabular reports, and WWW reports. An example gray scale shaded map showing average observed speeds from 4:30 pm to 5:30 pm during the 1995-1996 academic year in Baton Rouge was already shown in figure 9. Similar maps can be generated by modifying the query to include different time periods or show other speed values such as median speeds and minimum speeds. Appendix G in volume II contains samples of these color coded maps.

A sample tabular report was already shown in figure 10. Appendix G in volume II contains the complete set of archival tabular reports for summer 1995, academic year 1995-1996, and summer 1996. These reports contain summary information such as average speed and travel time for each segment at specific time periods and date ranges. All 155,300 segment records are available in the database. Alternatively, they can be retrieved using a WWW home page that allows users to make off-line queries related to the Baton Rouge CMS. The address of this home page is http://www.rsip.lsu.edu/cms/cmsbtr/cms-query.html. As mentioned previously, another WWW home page was developed to display real-time travel time data at specific checkpoints along the I-10 and I-12 corridors (figure 14). The address of this home page is http://www.rsip.lsu.edu/cms/cmsbtr/hwyckpoints.html.

Table 6
Segment record summary by corridor and time period in Baton Rouge

Corridor			Summer 1995			Academic year 1995-1996			Summer 1996			Total
ID	Name	# segm.	6-9 am	Off peak	3-6 pm	6-9 am	Off peak	3-6 pm	6-9 am	Off peak	3-6 pm	
1	I-10	219	2,314	3,614	2,507	7,101	1,937	6,036	2,404	1,019	2,286	**29,218**
2	I-110	115	2,695	4,710	3,540	3,431	501	2,456	132	517	9	**17,991**
3	I-12	194	2,127	3,238	2,518	4,532	1,343	3,037	3,117	601	2,256	**22,769**
4	LA 19	68	268	424	388	939	267	474				**2,760**
5	Plank Rd	181	949	1,297	1,126	1,647	570	544				**6,133**
6	Airline Hwy	277	2,392	4,369	2,832	6,013	2,278	3,738			2	**21,624**
7	Florida Blvd	190	1,913	1,933	1,645	4,148	2,423	3,226		159	158	**15,605**
8	Mickens Rd	34	170	187	221	235	121	245				**1,179**
9	Sherwood Forest Blvd	86	707	1,192	817	1,264	472	569				**5,021**
10	Siegen Lane	32	332	525	372	551	148	249				**2,177**
11	N Foster Dr	12	63	49	58	68	41	69				**348**
12	Government St	52	390	765	582	612	190	395		2	9	**2,945**
13	Jefferson Hwy	70	528	1,121	679	1,191	389	677				**4,585**
14	Staring Lane	20	220	251	232	493	204	270				**1,670**
15	Essen Lane	28	268	354	298	679	293	369				**2,261**
16	Bluebonnet Rd	18	88	96	94	315	160	193				**946**
17	Burbank Dr	10	104	130	109	217	83	131				**774**
18	Nicholson Dr	15				229	108	156	5	35	13	**546**
19	LA 1	6	13	45	49	68	20	34				**229**
20	Greenwell Springs Rd	70	500	435	554	565	236	599				**2,889**
21	Scenic Hwy	125	426	635	568	1,287	248	842		4		**4,010**
22	Range Ave	30	139	114	148	523	383	300	26	2	19	**1,654**
	Subtotal	**1,852**	**16,606**	**25,484**	**19,337**	**36,108**	**12,415**	**24,609**	**5,684**	**2,339**	**4,752**	**147,334**
	Non-main route segm.	545	1,010	1,651	1,211	1,793	485	1,180	233	163	240	**7,966**
	Total	**2,397**	**17,616**	**27,135**	**20,548**	**37,901**	**12,900**	**25,789**	**5,917**	**2,502**	**4,992**	**155,300**

Shreveport

The Shreveport MPO, the Northwest Louisiana Council of Governments (NLCOG), defined a congestion corridor network composed of 10 corridors covering 93 mi (figure 18). As shown in table 7, of the 10 corridors, two were located on the interstate highway system and covered 19 mi. The remaining seven corridors were located on principal arterials and covered 74 mi. RSIP was responsible for all activities, except data collection. NLCOG made all GPS runs between July 1995 and August 1996. A description of all activities is provided below.

Data collection

Equipment

NLCOG used a rover/base station approach for DGPS data collection *[19]*. The GPS rover receiver was a Pathfinder ProXL Trimble unit, equipped with a TDC1 datalogger. This unit collected data from the constellation of satellites and determined coordinates (in latitude, longitude), time (in Universal Coordinated Time - UTC), speed (in mph), and other ancillary data. The base station was a 4000 SSE geodetic surveyor receiver. Differential correction was made by postprocessing the raw GPS data from the rover unit with the GPS data from the base station. NLCOG obtained submeter positional accuracy with this system configuration.

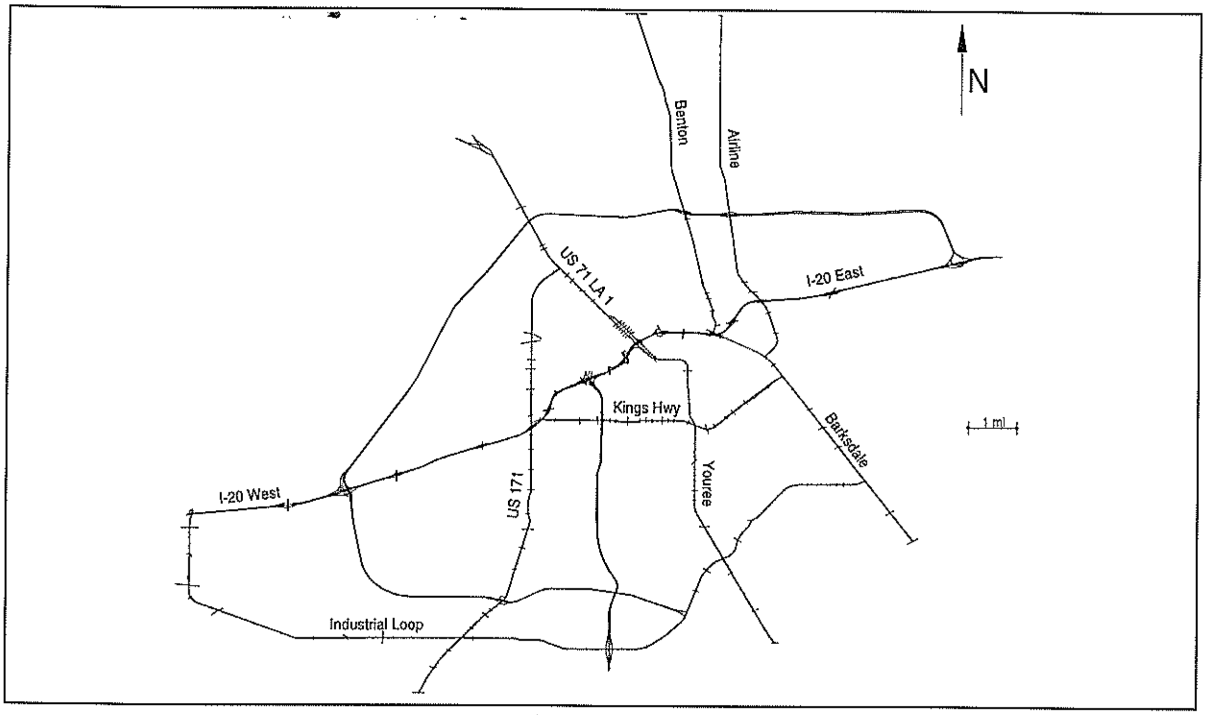


Figure 18
Congestion corridor network in Shreveport

Table 7
Congestion corridor network in Shreveport

ID	Name	From	To	Dir.	Functional class	Length (mi)	
1	I-20 West	LA 526	Spring St	E-W	Interstate	10.69	
2	I-20 East	Spring St	I-220	E-W	Interstate	8.20	
3	US 71 LA 1	I-20	US 71 LA 1 Split	N-S	Princ. Arterial		7..04
4	Youree	I-20	Flournoy Lucas	N-S	Princ. Arterial		8.58
5	US 171	Williamson Way	Market St	N-S	Princ. Arterial		11.57
6	Industrial Loop	I-20	Barksdale Blvd	E-W	Princ. Arterial		18.58
7	Kings Hwy	Hearne Ave	Barksdale Blvd	E-W	Princ. Arterial		5.76
8	Benton	Old Minden Rd	Kingston Rd	N-S	Princ. Arterial		8.22
9	Airline	Barksdale Blvd	Kingston Rd	N-S	Princ. Arterial		8.60
10	Barksdale Blvd	Airline Dr	Curtis Sligo Rd	N-S	Princ. Arterial		5.54
						18.89	**73.89**

Base vector map

As in Baton Rouge, DGPS points, recorded every one second, were used to construct the centerline base map. In Shreveport, only one GPS run per corridor and per direction of travel was made for the purposes of creating the base line work. This run was usually made on the right-most lane. Physical discontinuities such as interchanges, on-ramps, off-ramps, and signalized intersecting streets were also surveyed. The resulting DGPS points were imported into an Intergraph's MGE design file to create linear graphical elements. Whenever possible, all line work was offset a short distance in order for the graphical elements to represent the location of the directional centerlines. As in Baton Rouge, a weeding procedure was applied to eliminate as many redundant points as possible without jeopardizing positional accuracy.

Each corridor directional centerline was then partitioned using 0.2 mi segments, following the procedure summarized in table 3. A total of 1,473 segments, covering 238 mi of directional centerlines were defined. A total of 1,092 of these segments were located along the main routes. The remaining 381 segments were located on interchanges, on-ramps, off-ramps, and intersecting streets.

Routes and driving schedules

NLCOG made GPS runs between July 1995 and August 1996. Following ITE guidelines *[4]*, they scheduled two runs per direction and per time period on each CMS corridor. Because not much variation in average speed was found between the two runs, NLCOG did not schedule additional runs. Runs were made during four traffic time periods: morning (7:30 to 8:30 am), noon (12:00 to 12:30 pm), afternoon (4:30 to 5:30 pm), and off peak. The noon runs were made on the Kings Hwy corridor, which was reported to exhibit congestion problems during the lunch hours. Most off peak runs were made late at night, between 9:00 pm to 3:00 am.

Data reduction

A summary of dates, file names, and file sizes is included in appendix F. A total of 844 mi of travel runs on the 93 mi highway network were made, resulting in 100 GPS data files and 85,000 GPS point records. The data reduction GIS application described previously was used to process these records, resulting in 5,048 segment records. On average, there were about 3.4 records per segment. Obviously, some corridors and segments ended up with more records than others, as shown in table 8. As in Baton Rouge, only records and number of segments associated with segments located directly on the main routes are listed separately. All non-main route segments such as interchanges and ramps, as well as their corresponding records, are lumped together.

Data reporting

In the case of Shreveport, two reporting options were considered for displaying segment travel time data: Color coded maps; and archival tabular reports. The procedure to generate color coded maps is exactly the same as the one described for Baton Rouge. A variety of maps can be generated by adjusting the database query to include different time periods or show different speed values such as harmonic mean speeds, median speeds, minimum speeds, and free flow speeds. Samples of color coded maps are shown in appendix H (volume III).

The complete set of archival tabular reports for the time period July 1995 - August 1996 is also shown in appendix H (volume III). These reports contain summary information such as average speed and travel time for each segment at specific time periods and date ranges. All 5,048 segment records are available in the database.

Table 8
Segment record summary by corridor and time period in Shreveport

Corridor			Time period				Total
ID	Name	# segm.	AM peak	Noon peak	PM peak	Off peak	
1	I-20 West	122	203		202	64	**469**
2	I-20 East	95	173		169	88	**430**
3	US 71 LA 1	87	169		204	104	**477**
4	Youree	105	213		217	112	**542**
5	US 171	144	288		288	142	**718**
6	Industrial Loop	211	256		207	134	**597**
7	Kings Hwy	78		153	156	78	**387**
8	Benton	90	180		183	90	**453**
9	Airline	96	190		239	96	**525**
10	Barksdale Blvd	64	133	2	137	71	**343**
	Subtotal	**1,092**	**1,805**	**155**	**2,002**	**979**	**4,941**
	Non-main route segments	381	42	0	49	16	**107**
	Total	**1,473**	**1,847**	**155**	**2,051**	**995**	**5,048**

New Orleans

The New Orleans MPO, the Regional Planning Commission (RPC), defined a congestion corridor network composed of 31 corridors covering 248 mi. For this study, data collection, data reduction, and data reporting was limited to seven corridors, covering 86 mi (figure 19). As shown in table 9, of the seven corridors three were located on the Interstate highway system and covered 31 mi. The remaining four corridors were located on principal arterials and covered 54 mi. RSIP was responsible for all activities, except data collection. RPC made all GPS runs. RPC staff made the runs for identifying both directions of travel and physical discontinuities on all 31 corridors. Actual travel time runs on the reduced seven-corridor network were made by a consultant to RPC between June 1996 and July 1996. A description of all activities is provided below.

Data collection

Equipment

RPC used a GPS equipment configuration similar to the one used in Baton Rouge (figure 17). The only difference between New Orleans and Baton Rouge is that in New Orleans no microcassette audio tape recorder was used to document changing weather conditions, disruptive traffic incidents, or queues at major intersections.

Base vector map

As in Baton Rouge, DGPS points, recorded every one second, were used to construct the centerline base map. In New Orleans, only one GPS run per corridor and per direction of travel was made for the purposes of creating the base line work. This run was usually made on the right-most lane. Physical discontinuities such as interchanges, on-ramps, off-ramps, and signalized intersecting streets were also surveyed. The resulting DGPS points were imported into an Intergraph's MGE design file to create linear graphical elements. As in Baton Rouge, a weeding procedure was applied to eliminate as many redundant points as possible without jeopardizing positional accuracy.

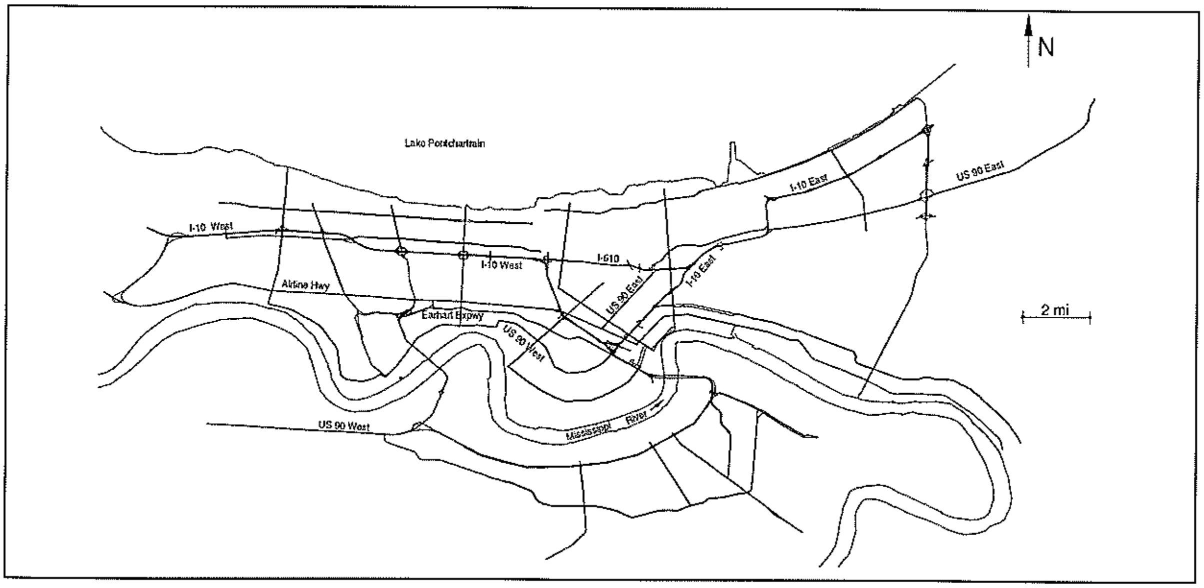


Figure 19
Congestion corridor network in New Orleans

Table 9
Congestion corridor network in New Orleans

ID	Name	From	To	Dir.	Functional class	Length (mi)	
1	I-10 West	I-310	Poydras St at Camp	E-W	Interstate	10.35	
2	I-10 East	Poydras St at Camp	I-510	E-W	Interstate	16.30	
3	I-610	I-10 West	I-10 East	E-W	Interstate	4.68	
4	Airline Hwy	Jefferson Parish line	Tulane at Loyola	E-W	Princ. Arterial		15.37
5	US 90 East	Claiborne at Tulane	US 11 at US 90	E-W	Princ. Arterial		16.71
6	Earhart Expwy	Hickory	Poydras at Loyola	E-W	Princ. Arterial		5.22
7	US 90 West	Jefferson Parish line	Claiborne at Tulane	E-W	Princ. Arterial		16.90
						31.33	**54.20**

Each corridor directional centerline was then partitioned using 0.2 mi segments, following the procedure summarized in table 3. A total of 3,794 segments, covering 581 mi of directional centerlines were defined. For corridors No. 1 to 7 (table 9), 1,338 segments, covering 214 mi of directional centerlines were defined. A total of 1,039 of these segments were located along the main routes. The remaining 299 segments were located on interchanges, on-ramps, off-ramps, and intersecting streets.

Routes and driving schedules

RPC began making runs on June 1996. Since then, RPC has been collecting GPS data on a continuous basis, and it intends to continue doing so until the end of 1997. For this report, data collected on corridors No. 1 to 7 between June and July 1996 were made available. In general, these runs were conducted during the morning peak (6-9 am) and afternoon peak (3-6 pm) time periods. A few GPS data points were collected outside these time periods, particularly just after 9 am. For the most part, however, these extra points were collected during extremely heavy traffic conditions, possibly due to incidents. As a result, this data should not be considered representative of off peak traffic conditions.

Data reduction

A summary of dates, file names, and file sizes is included in appendix F. A total of 3,805 mi of travel runs were made, resulting in 68 GPS data files and 322,000 GPS point records. The data reduction GIS application described previously was used to process these records, resulting in 22,613 segment records. On average, there were about six records per segment. Obviously, some corridors and segments ended up with more records than others, as shown in table 10. As in Baton Rouge, only records and number of segments associated with segments located directly on the main routes are listed separately. All non-main route segments such as interchanges and ramps, as well as their corresponding records, are lumped together.

Table 10
Segment record summary by corridor and time period in New Orleans

Corridor			Time period			Total
ID	Name	# segm.	6-9 am	3-6 pm	Other	
1	I-10 West	115	2,027	1,643	359	**4,029**
2	I-10 East	194	3,093	2,741	711	**6,545**
3	I-610	56	3,330	1,967	178	**5,475**
4	Airline Hwy	186	419	348	92	**859**
5	US 90 East	201	796	863	127	**1,786**
6	Earhart Expwy	58	457	261		**718**
7	US 90 West	229	658	566	21	**1,245**
	Subtotal	**1,039**	**10,780**	**8,389**	**1,488**	**20,657**
	Non-main route segments	299	669	412	54	**1,135**
	Other corridors	2,456	379	346	96	**821**
	Total	**3,794**	**11,828**	**9,147**	**1,638**	**22,613**

Data reporting

As in the case of Shreveport, two reporting options were considered for displaying segment travel time data: Color coded maps; and archival tabular reports. Samples of color coded maps are shown in appendix I (volume IV). The complete set of archival tabular reports for the time period June 1996 - July 1996 is also shown in appendix I (volume IV).

CONCLUSIONS

This report describes the results of a study undertaken to demonstrate the feasibility of using GPS and GIS to measure travel time and speed data on urban highways. This data is being used to develop Congestion Management Systems in Baton Rouge, Shreveport, and New Orleans. The new methodology significantly shortens and automates data collection and data reduction. It also enables improved procedures for documenting and analyzing measured travel times and speeds on highway networks.

The methodology described includes data collection, data reduction, and data reporting procedures. The data collection procedure is based on the use of GPS receivers to automatically detect time, local coordinates, and speed every one second. This way, an accurate depiction of vehicle location and speed is obtained. A trade-off in the automation provided by GPS is the large amounts of data it produces. A data filtering and reduction procedure was devised to decrease the amount of data to be stored, without seriously impacting the quality of the original data. Such a procedure was based on the aggregation of GPS data using segments of highway which do not typically exceed 0.2 mi in length. However, the model is sufficiently general so that other segment sizes can be easily accommodated.

The data reduction and storage system is based on a geographic database that uses a relational database model. The GIS-based model allows for both spatial and non-spatial queries, therefore providing the possibility of a wide variety of data reporting options in addition to the traditional speed-distance or speed-time profiles. In this report, the following data reporting options were used: hard copy color coded maps and archival tabular reports; and on-line, Internet-based, reports. Color coded maps show the spatial variation of items such as speed and travel time, and are particularly suitable for explaining travel time delays and congestion issues at public meetings. Tabular reports offer a very compact way of archiving travel time and speed data along highway segments. This makes them suitable for archival and analytical purposes. The procedure to produce these tabular reports has been automated, therefore increasing the usefulness of such an approach. Reporting procedures using WWW resources were also implemented. These procedures allow any user with access to the Internet to select highway segments and retrieve all records associated with these segments, and to retrieve real-time travel time data between checkpoints located on a highway corridor.

The methodology described was used to collect, process, and report travel time data on Baton Rouge, Shreveport, and New Orleans. A summary of congestion corridor characteristics, travel time data, and reporting procedures completed on each corridor is shown in table 11.

As mentioned previously, segmentation schemes based on nominal spacing between checkpoints other than 0.2 mi are also possible. To evaluate the effect of using different segment lengths, segmentation schemes based on segment lengths ranging from 0.1 mi to 5 mi were used and compared (appendix C). In general, the richness of the original GPS data was lost completely when using 2 mi or longer. Even when using physical discontinuity segments or 0.5 mi segments the resemblance to the original GPS data was vague. Only when segments shorter than 0.5 mi in length were used, did the resemblance improve significantly. Not surprisingly, the 0.1 mi segmentation scheme showed the best results. The question then arises as to what segment length should be used in any particular situation. Based on the results described in this report, signalized highways seem to require shorter segments than freeways do to represent the spatial and temporal variability of speed with comparable degrees of accuracy. For freeways a segment length of 0.2 mi appears to be quite acceptable. In contrast, signalized highways appear to require considerably shorter segments, possibly around 0.1 mi. Additional research would be needed to completely settle this issue.

Table 11
Summary corridor, travel time, and reporting data for Baton Rouge, Shreveport, and New Orleans

Description	Baton Rouge	Shreveport	New Orleans
Congestion Corridor Network Information:			
Number of corridors:			
Interstate highways	3	2	3
Principal arterials	19	8	4
Total	22	10	7
Length			
Interstate highways (mi)	45.36	18.89	31.33
Principal arterials (mi)	105.63	73.89	54.20
Total	150.99	92.78	85.53
Segments:			
Number of segments			
Main routes	1,852	1,092	1,039
Interchanges, ramps, etc.	545	381	299
Total	2,397	1,473	1,338
Total length (mi)	368.62	237.90	213.80
Average segment length (mi)	0.15	0.16	0.16
Travel Time Data:			
Number of GPS data files	428	100	68
Number of GPS data points	2.5 million	85,000	322,000
Segment data:			
On main routes:			
6-9 am	58,398	1,805	10,780
Noon peak		155	
3-6 pm	48,698	2,002	8,389
Off-peak	40,238	979	1,488
Total	147,334	4,941	20,657
On non-main routes:	7,966	107	1,135
On other corridors			821
Total	155,300	5,048	22,613
Data Reporting:			
Color coded maps	yes	yes	yes
Archival tabular reports	yes	yes	yes
CMS WWW query interface	yes	no	no
Real time WWW travel time data	yes	no	no

A procedure to determine representative speed values when more than one run is made on a network was included (appendix C). Such a procedure was based on the arithmetic average of travel times, which resulted in a weighted harmonic average computation for speed. Because harmonic mean speeds are very sensitive to small speed values, an alternative procedure, using median speeds, was implemented. For most segments, the median speed and the harmonic mean speed are very similar. In some cases, however, the two values differ considerably. For such cases, the median speed approach represents typical traffic flow conditions better than the harmonic mean speed approach. In contrast, the harmonic mean speed tends to lump "normal" traffic with traffic affected by incidents. Eventually, overlaying harmonic mean speed maps on median speed maps may be useful for detecting areas affected by traffic incidents.

The procedures described in this report served the purpose of providing a sound methodology, based on GPS and GIS, to conduct the type of macro travel time studies needed to develop congestion management systems for metropolitan areas such as Baton Rouge, Shreveport, and New Orleans. However, at least in principle, such a methodology could also be applied to other scenarios such as before-and-after studies, signal timing optimization, and so on, which are much more localized. Because these studies are needed quite frequently, the methodology should be easily accessible to any district engineer. This is currently not possible because access to an Intergraph workstation with all the requisite GIS software is prohibitively expensive (around $15,000). However, over the past 22 months several inexpensive GIS packages have emerged that cost less than $3,000 and can run on an ordinary PC. It would be desirable then to adapt the current GPS/GIS based methodology to one of these cheaper packages so that district engineers and smaller MPOs would have the requisite tools for efficiently conducting travel time studies along signalized arterial routes.

Results from this research project have been disseminated to the Transportation Engineering community through several papers published or submitted to peer reviewed journals and professional conferences *[20]*, *[21]*, *[22]*, *[23]*, *[24]*, *[25]*, *[26]*.

LIST OF ACRONYMS AND ABBREVIATIONS

AVI:	Automatic vehicle identification
AVL:	Automatic vehicle location
CAAA:	Clean Air Act Amendment
CMAQ:	Congestion Mitigation and Air Quality Program
CMS:	Congestion management system
CRPC:	East Baton Rouge Capital Region Planning Commission
DCI:	Differential Corrections Inc.
DGPS:	Differentially corrected GPS (data)
DMI:	Distance measuring instrument
DOTD:	Louisiana Department of Transportation and Development
FHWA:	Federal Highway Administration
ft:	foot (1 ft = 0.3048 m)
GIS:	Geographic information system
GPS:	Global positioning system
HTML:	Hypertext markup language
in:	inch (1 in = 2.54 cm)
ISTEA:	Intermodal Surface Transportation Efficiency Act
ITEA:	Institute of Transportation Engineers
km:	kilometer (1 km = 0.6215 mi)
km/h:	kilometers per hour (1 km/h = 0.6215 mph)
LADOTD:	Louisiana Department of Transportation and Development
LSU:	Louisiana State University
LTRC:	Louisiana Transportation Research Center
MDL:	Microstation developing language
MGE:	Intergraph's Modular GIS Environment
m:	meter (1 m = 3.281 ft)
mi:	mile (1 mi = 1.609 km)
mph:	miles per hour (1 mph = 1.609 km/h)
MPO:	Metropolitan planning organization
NLCOG:	Northwest Louisiana Council of Governments
ODBC:	Open database connectivity
PC:	Personal computer
PERL:	Practical extraction and report language
RDS:	DCI's Remote Data System
RPC:	New Orleans Regional Planning Commission
RSIP:	Remote Sensing and Image Processing Laboratory
SA:	Selective availability
SEP:	Spherical error probability
SQL:	Structured query language
TIGER:	Topologically integrating geographic encoding and referencing
UTC:	Universal coordinated time
WWW:	World wide web

REFERENCES

1. Intermodal surface transportation efficiency act. US code, Title 23, Chapter 3, Section 303, 1991.
2. Clean air act amendment, US code, Title 1, Section 103, 1990.
3. May, A.D. *Traffic Flow Fundamentals*. Prentice-Hall, Englewood, New Jersey, 1990, 464 p.
4. Robertson, H.D. *Travel Time and Delay Studies*. Manual of Transportation Engineering Studies, Robertson, H.D. (Ed), Institute of Transportation Engineers, 4th Edition, Washington DC, 1994, pp. 52-68.
5. Liu, T.K., and Haines, M. "Travel Time Data Collection Field Tests - Lessons Learned." Report FHWA A-PL-96-010, US Department of Transportation, 1996, 116 p.
6. Turner, S.M. *Advanced Techniques for Travel Time Data Collection*. Transportation Research Record 1551, Transportation Research Board, Washington, D.C., 1996, pp. 51-58.
7. Benz, R.J., and Ogden, M.A. *Development and Benefits of Computer-Aided Travel Time Data Collection*. Transportation Research Record 1551, Transportation Research Board, Washington, D.C., 1996, pp. 1-7.
8. Taylor, M.A.P., Young, W., and Bonsall, P.W. *Understanding Traffic Systems: Data, Analysis and Presentation*. Avebury Technical, Aldershot, England, 1996, 443 p.
9. Laird, D. *Emerging Issues in the Use of GPS for Travel Time Data Collection*. National Traffic Data Acquisition Conference, Albuquerque, New Mexico, May 6-9, 1996.
10. CUTR. *Demonstration of Video-Based Technology for Automation of Traffic Data Collection*. Center for Urban Transportation Research, University of South Florida, Tampa, Florida, 1996, 68 p.
11. Leick, A. *GPS Satellite Surveying*. John Wiley & Sons, New York, 1995, 560 p.
12. Advanstar Communications. *Receiver Survey*. GPS World, 7(1), 1996, pp. 32-51.
13. Harding, J., Salwin, A., Shank, D., and Ghaman, R. *GPS Applications for Traffic Engineering Studies*. Preprint 960214, Transportation Research Board, 75th Annual Meeting, Washington D.C., January 7-11, 1996.
14. Guo, P., and Poling, A.D. *Geographic Information Systems/Global Positioning Systems Design for Network Travel Time Study*. Transportation Research Record 1497, Transportation Research Board, Washington, D.C., 1995, pp. 135-139.
15. Zito, R., D'Este, G., and Taylor, M.A.P. *Global Positioning Systems in the Time Domain: How Useful a Tool for Intelligent Vehicle-Highway Systems?* Transportation Research C, 3(4), 1995, pp. 193-209.
16. Quiroga, C.A., and Bullock, D. *Determination of Sample Sizes for Travel Time Studies*. Submitted to the ITE Journal, March, 1997.
17. Wall, L. *Perl Kit, v. 5.0*. Free Software Foundation, Cambridge, Massachusetts, 1994.
18. Trimble Navigation Ltd. *Placer GPS 400 Installation and Operator's Manual*. Sunnyvale, California, 1993.
19. Petro, C. *GPS Hits the Road: Pinpointing the Sources of Traffic Congestion*. GPS World, 7(9), 1996, pp. 42-52.
20. Quiroga, C.A., and Bullock, D. *Congestion Management System Architecture for Controlled Access Facilities*. Transportation Research Record 1551, Transportation Research Board, Washington D.C., 1996, pp. 105-113.

21. Quiroga, C.A., and Bullock, D. *Implementation of a Congestion Management System for Baton Rouge*. Paper presented at the ITE Southern District 44th Annual Meeting, Jackson Mississippi, April 21-24, 1996.

22. Bullock, D., Quiroga, C.A., and Kamath, N. *Data Collection and Reporting for Congestion Management Systems*. Paper presented at the National Traffic Data Acquisition Conference, Albuquerque, NM, May 6-9, 1996.

23. Bullock, D., Quiroga, C.A., and Schwehm, C. *Dissemination of Travel Time Information along Congested Corridors*. Paper presented at the International Symposium of Automotive Technology and Automation (ISATA) Dedicated Conference on Global Deployment of Advanced Transportation Telematics, Florence, Italy, June 3-5, 1996.

24. Quiroga, C.A., Bullock, D., and Schwehm, C. *Dissemination of Travel Time Information Using the World Wide Web*. Preprint 970322, Transportation Research Record, 76th Annual Meeting, Washington, D.C., January 12-16, 1997.

25. Quiroga, C.A., and Bullock, D. *Travel Time Studies with GPS: An Improved Methodology*. Submitted to Transportation Research Part C: Emerging Technologies, March, 1997.

26. Quiroga, C.A., and Bullock, D. *Travel Time Studies on Signalized Highways Using GPS*. To be presented at the ASCE 1997 Specialty Conference on Traffic Congestion and Traffic Safety, Chicago, Il, June 7-11, 1997.

27. Duncan, A.J. *Quality Control and Industrial Statistics*. Irwin, Homewood, Illinois, 5th Edition, 1986, 1123 p.

APPENDIX A: DATA COLLECTION METHODOLOGY

Segmentation Procedure Using MGE

Differentially corrected GPS (DGPS) points, recorded every one second, are used to construct the centerline base map. For each direction of traffic, at least two GPS runs should be made: one for the right-most lane, and the other one for the left-most lane. Physical discontinuities such as interchanges, on-ramps, off-ramps, and signalized intersecting streets are also surveyed. The resulting DGPS points are imported into an Intergraph's MGE design file to create linear graphical elements. The procedure to do this is summarized below:

1. Generate ASCII input GPS data file

- In Excel, open an .ndc GPS data file (see Data Collection Procedure in Baton Rouge in this appendix) and edit it using the Concatenate function to create entries having the following format:

```
ll=92.4345567,31.3479808
ll=92.4346782,31.3479805
ll=92.4348902,31.3479790
        :
        :
```

  Save the resulting file using the .prn extension (formatted text -space delimited) option.

2. Import GPS data into empty design file

- In MGE, create an empty .dgn file, say test1.dgn, having the following coordinate system configuration:
  - System: Geographic (longitude, latitude)
  - Geodetic datum: North American 1983 (all GPS data come with this datum)
  - Ellipsoid: GRS 80
  - Units and formats: Geographic (longitude, latitude); positive N,W
    Projection (m); Easting, Northing
- Under File/Working Units/Mapping, type in 3600000 in the resolution box. Also, select DG in the UOR per box.
- Select the multiline icon (line string tool in Microstation jargon) and type in "@" followed by the .prn file name at the Microstation prompt. Follow a similar procedure for all GPS runs made to create the base vector map, including those made to survey interchanges, on-ramps, off-ramps, signalized intersections, and so on. For efficiency and clarity, assign different Microstation levels to each GPS run.
- Once the line work is created, it is necessary to change the coordinate system of the file to measure distances in mi. To do this, create another empty file, say test2.dgn, having a polyconic coordinate system configuration (under parameters, choose suitable longitude and latitude values: the format should be 92:00... for longitude, and 31:00... for latitude.
- Under units and formats, select Longitude, Latitude - Positive N,W for Geographic Format, and click on OK twice. Do the same for both the primary and secondary coordinate systems.

- Under file - working units (in the coordinate system option), type in 63360 in the resolution box (this comes from 5,280 ft/mi * 12 in/ft). Also select mi in the UOR per box. Then click on OK.
- In Microstation, select Settings/Coordinate Readout/Coordinates Format and choose Master Units.
- Under File, select compress design and then save settings. Then exit Microstation
- In the main MGE window, choose Tools/Projection Manager. Select map convert and click on apply. Select test1.dgn as the input design file and test2.dgn as the output design file.

3. Create line work

- Create a new design file and attach the GPS design file test2.dgn as a reference file in the background. In the new design file, draw line strings by joining contiguous physical discontinuities following the approximate location of the directional centerline. Because the data reduction utility is direction sensitive, line strings must be drawn following the traffic direction, i.e., from upstream to downstream. If graphical elements are drawn in the opposite direction as the traffic flow, their direction must be reversed prior to using the data reduction utility. In general, it is sufficient to draw graphical elements using the line string tool. More sophisticated tools which involve curve forming and which work well under Microstation do not usually behave well when performing MGE coordinate transformation operations.
- Once the master directional centerline design file is created, the next step involves segmenting all graphical elements. For safety, maintain a safe copy of the master centerline design file. In file test2.dgn, use the point drawing tool to define points at regular intervals, say every 0.2 mi. To make points visible, use a weight value such as 5 or 7. See table 3 in the main text for indications as to how define points close to the upstream end of the graphical element.
- Once all points are drawn, draw short auxiliary lines perpendicular to the line work and snap them to the points. Points can then be erased. These lines are used to break the line work into segments. Points cannot be used for this purpose because they are treated as zero length lines in Microstation. One way to avoid dealing with points altogether is by creating Microstation cells containing short linear elements. Such cells can then be located along graphical elements using the point drawing tool. Since cells are complex features, their complex status must be dropped before the short lines they contain can be used for intersecting the graphical elements representing the directional centerlines. Breaking the line work into segments can be done either manually or automatically. The advantage of doing it automatically is that it is a one step operation. The disadvantage is that segments located on overpasses are also cut and, as a result, they must be reunited. The advantage of intersecting segments manually is that segment intersection is selective, therefore effectively avoiding the situation described above. The disadvantage is that it is a labor intensive exercise which requires considerable care and concentration.
- Link all segments to table CORR_SEGMENTS by using the Feature Maker tool. MGE assigns a unique MSLINK code to each segment. Obviously, only graphical elements representing highway segments should be linked to the database. This means that the small lines used for segmentation should be put in a separate design file prior to establishing any database linkage. Once this database linkage is established, table CORR_SEGMENTS can

be populated. Table population can be made at any time except for attribute SEGLENGTH. Since the design file still has a polyconic projection, segment lengths can be automatically computed using the length loader tool in MGE. After this is done, the design file can be transformed back into a geographic coordinate system.

Data Collection Procedure in Baton Rouge

The following pages describe a set of instructions given to students who made GPS runs in Baton Rouge. Because more GPS runs may be made in the future, either by students or by technicians, it was considered appropriate to include such instructions here. Three main topics are included: Vehicle operation, data collection, and tape transcription.

Vehicle operation

1. General

- The LTRC gate lock code is _____ (this space has been left blank intentionally). When you lock it back, set it to _____ again and then change the numbers to something else.
- Update the vehicle log on a daily basis. Every other Friday, tell Special Studies the mileage on the cars. Once a month, turn the vehicle log in to Special Studies.
- You are responsible for any traffic violations and the corresponding tickets.
- If you have any mechanical problems while driving, or if you are involved in a serious accident, call

  231-4166

  This is the number of the guardhouse at DOTD to report mechanical problems.
  If you are using a radio, switch to Channel 6 (Emergency Channel) to report the situation.
  Also call Special Studies.
- If you are involved in an accident, KEEP COOL and:
  - Call the Police to the site. This is very important. As long as you drive a State vehicle, you MUST call the Police to the site. Make sure to get a copy of the Police Report.
  - Draw a sketch of the accident site (intersecting streets, # of lanes, etc..).
  - Get the following information from the driver of the other vehicle: Name, address, phone number, driver's license number and expiration date, insurance company, policy number and expiration date; vehicle make, model, color, and VIN number.
  - Get an Accident Report Form from Special Studies and fill it out. The first page must be turned in to Special Studies within 24 hours of the accident.

2. Maintenance

- Check the car fluids regularly.
- Every 5,000 mi or so, schedule a visit to the DOTD shop at Tom Drive:
  - Check in the glove compartment for the next service date (mileage or date; whichever comes first).

    - Coordinate with another person who has the same driving schedule as you do to take the vehicle to the DOTD Central Repair Shop (7686 Tom Drive) so that you get a ride back to LSU.
    - Fill out a repair form provided to you by the shop foreman. You need to show him the vehicle card (white plastic card).
    - Remember not to leave any GPS equipment in the vehicle.
- Once in a while, wash the vehicle. Take turns with other drivers for completing this activity.

3. Refueling

- Each vehicle has a vehicle FUELMAN card. There is also an Employee FUELMAN card that needs to be used for refueling.
- Go to a gas station that accepts the FUELMAN card. To refuel,
    - Scan employee card
    - Key in the PIN number (___) and press Enter
    - Scan vehicle card
    - Enter odometer reading
    - Get the receipt, initial it and date it
- Turn in all gasoline receipts to Special Studies (once a week).

Data collection

1. Install equipment in vehicle

- Before leaving the office,
    - Make sure equipment is complete. See checklist for details. Do not forget to take both the scanner and one of the radios with you. Occasionally it will be necessary to contact you while you are driving and, consequently, having radio communication is essential.
    - Mark your assigned or intended route on the bulletin board chart. Make sure you do this so that other drivers know what route you are surveying.
    - Make sure that all batteries are OK. This includes the receiver's 9-V battery, audio recorder batteries, and scanner batteries.
- Mount magnetic GPS receiver antenna on vehicle roof. For consistency, place antenna halfway between driver door and passenger door, and one foot back from windshield. Pass cable through passenger window, and connect antenna to GPS receiver.
- Mount FM antenna on roof, approximately 1.5 ft back from GPS receiver antenna. Pass cable through passenger window and connect FM antenna to differential correction unit.
- Roll up passenger window completely, so that no cable is loose or flapping
- Connect laptop to its battery recharger. Then connect battery recharger to adapter.
- Plug in GPS receiver cable and laptop adapter to power plug.
- Turn vehicle engine on. THEN, plug in power plug to cigarette lighter.

2. Activate data collection equipment

- Power on laptop computer
- Type "cd gpssk" and press "Enter". Then type "gpssk" and press "Enter"

- Press "F2" (log) and then "F4" (both)
- To ensure a proper communication between receiver and computer, verify that the GPS time is being displayed and updated. Also verify that GPSSK is saving data onto a file. The file name is visible on the lower right side of the screen.
- To view the latest GPS points, press "F9" (backup) and then "F4" (plot). A map will be displayed showing the vehicle location on a grid. Press "F9" again to display navigational data.
- A "DGPS" sign on the lower right corner of the screen is visible when using a differential correction unit. If the sign is not visible within a couple of minutes, check that communication between the receiver and the RDS 3000 unit is properly set. To do this,
  - Press "F9" twice to exit gpssk.
  - At the DOS prompt, type "cd c:\procomm" and press "enter".
  - Type "procomm" to activate the communications utility.
  - Press "page up" (PgUp) to activate the UPLOAS menu.
  - Type "7" to select the ASCII option.
  - Type "fix" to change the baud rate from 4800 to 1200.
  - Press "Alt" "x" to exit procomm. Now run gpssk again, as described above.
- Before starting to drive, record the following on the microcassette recorder:
  - GPS Unit #, location, date, and time. Time should be given in Greenwich Mean Time. Therefore, it is important to set the 24-hr clock six hours ahead of the local time (five hours during daylight saving months -April to October).
  - Scheduled route
  - Weather and lighting conditions

3. While driving

- Remember you are driving a "floating" car, i.e., make sure you pass as many vehicles as vehicles pass you. Every 15 minutes or so, count the number of vehicles you pass, as well as the number of vehicles that pass you, for about one minute. This will help you adjust your driving speed accordingly. If driving on a six-lane highway (three lanes in each direction), drive on the middle lane, except when passing a slow moving vehicle. If driving on a four-lane highway (two lanes in each direction), try to stay on the right lane. Occasionally, however, you will need to move to the left lane to comply with the floating car condition described above.
- Check the laptop computer screen once in a while to make sure that GPS data is being received and saved properly.
- Record abnormal situations on the microcassette recorder. Examples of abnormal situations include changing weather and lighting conditions; queues at major intersections; incidents on both directions of travel (note specific impact on traffic - rubber necking/lane closures/debris); other incidents reported on the radio; and problems with GPS equipment. To ensure a smooth tape transcription operation, use the following protocol for recording data on the microcassette recorder:
  - Time: As mentioned, it is expressed as Greenwich Meridian Time. When recording, use the actual digit sequence, after the word "time". There is no need for specifying "hours", "minutes" and "seconds", as long as the digit sequence is pronounced loud and

clear. For example, if the time is 00:24:16, say "Time: zero zero, twenty four, sixteen", instead of "time: zero hours, twenty four minutes, and sixteen seconds".

- Route: Name of route you are driving at that moment. Try to use local route names instead of State route numbers. For example, say "Route: Government Street" instead of "Route: LA 73".
- Direction: For simplicity, only 4 cases are considered: "East bound", "West bound", "North bound", and "South bound".
- Location: One short sentence describing the closest landmark you are passing. Whenever possible, use intersecting streets as references. For example, say: "Just south of Corporate Blvd" or "Between Corporate Blvd and Rienzi Blvd" instead "In front of Shopping Center". Since you already mentioned the route you are driving, do not repeat it again here.
- Comment: Summarized description of what you want to describe. When recording incidents reported on the radio, it is OK to record the report directly on the tape recorder. However, while transcribing the tape, you will need to filter that information so that only incidents and abnormal situations are actually recorded in the database. Examples of situations to record in the database are incidents (such as accidents and signal malfunctions) on major highways, even if you drove a different route that day. In contrast, you should NOT record things like "traffic on I-12 WB is heavy between Sherwood and Jefferson" because such a message is too generic and may actually be inaccurate.

4. **At the end of the run**

- Record the following data on the audio recorder:
  - Date, stop time, and stop location
  - Weather and lighting conditions
- Exit GPSSK by pressing "F9" until the exiting message appears on the screen
- Turn off laptop computer
- Unplug power from cigarette lighter
- Remove GPS receiver and FM antennas from roof
- Put a twist-tie or a rubber band on each antenna
- Unplug GPS receiver cable and adapter from power plug
- Disconnect battery recharger from laptop
- Store equipment **neatly** inside briefcase (remember: somebody else will use the same equipment).

Data archival

1. **Archive GPS data**

- Connect laptop to network in the Transportation Lab
- Power on laptop, and select 3 (load everything including network)
- Type "cd gpssk" and press "Enter"
- Type "cms-cvt filename" to filter each .ltf file of interest. Example: cms-cvt 09281441 (do not type the file name extension; cms-cvt automatically assumes an .ltf extension). Then

verify the size in bytes of the .dc and .ndc files. Keep this information so that you can update file gpssum.doc later.

- Type "pkzip filename filename.ltf" to create a compressed copy of the .ltf file. You do not need to specify a .zip extension. Pkzip does it for you
- Type "dir *.zip" to verify that the compression process was properly done. File size ratios of 3:1 to 5:1 are typical. If the size of the .zip file is something like 134 bytes, no compression was achieved. In this case, try it again.
- Type "net use" and press "Enter" to make sure that the network connection is working properly. You should see a letter associated with /home2. Normally this letter is "j".
- Go to the j drive and then type "cd cmsbtr\gpsdata"
- Type "mkdir yymmdd" to create a subdirectory. Valid examples: 950903, 951011, 951102.
- Type "cd yymmdd"; then type "copy c:\gpssk\filename.ndc" and copy c:\gpssk\filename.zip"
- Take a ZIP disk to a the computer with a ZIP drive (blue unit on top of machine)
- Log into the network and then insert ZIP disk into ZIP drive
- Check what drive letter is associated with the Iomega ZIP disk. You can do this either by using File Manager or by typing "net use" at the DOS prompt. In general, the ZIP drive letter is "E". Also check the drive letter that is associated with /home2. As before, such letter is usually "j".
- Copy filename.zip from j:\cmsbtr\gpsdata\yymmdd\ to e:\
- Delete filename.zip from the network drive
- Remove ZIP disk and load Word to update file gpssum.doc with the information you had from the .ndc and .dc files. This file is in the cmsbtr\reports\ subdirectory. Include date (yymmdd), traffic condition (AMP, PMP, OP), file name, driver's initials, size in bytes of the .dc and .ndc files, and receiver information.
- Back in the Transportation Lab, go to the c:\gpssk\ directory and delete filename.ltf, filename.ndc, and filename.dc. **CAUTION: DO NOT type "delete *.*"; if you do it, you will also delete the gpssk program files!!!**

2. Transcribe tape

- In Windows, load Microsoft Access and open file j:\cmsbtr\database\conlog1.mdb to begin the tape transcription process
- Click on Forms and select form VEHICLE_AUDIO_LOG
- On the bar that says "record 1 of XXXX" on the lower part of the form, click on ">*" to create a new record. You should now see "record XXXX+1 of XXXX+1"on that bar.
- Fill in the data collected with the tape recorder. Create as many records in the database as events you recorded in the tape.
- When you are finished, close Access and properly label cassette tape with GPS Unit #, driver name initials, date, and traffic condition (AMP, PMP, OP)
- Turn off laptop computer and store <u>**neatly**</u> in briefcase

Equipment list

Laptop computer
GPS receiver Trimble Placer 400
Differential correction unit RDS 3000
GPS receiver antenna
FM antenna
Laptop - GPS receiver connector
Laptop battery recharger
Battery charger adapter
GPS receiver - power plug connector
Cigarette lighter power plug
Microcassette recorder
24-hr clock
Scanner
Screwdriver
Field procedures manual
Map of Baton Rouge
note pad
Pens
Sticky notes

APPENDIX B: DATA REDUCTION METHODOLOGY

Data Filtering Utilities

cms-cvt conversion program

This program reads raw GPS data from an .ltf file, filters it, and stores coordinates, time, and speed into two files: one with an .ndc extension, and another one with a .dc extension. The program is run at the DOS prompt by typing

```
cms-cvt filename
```

where "filename" is the name of the file. It is not necessary to type in ".ltf". The application automatically assumes that "filename" has an .ltf extension. This .ltf file is produced by the Trimble Placer 400 receiver and contains navigational data such as vehicle ID, GPS time, coordinates, speed, differential correction status, and diagnostics. A sample of records from file 08171241.ltf follows:

```
>RVR SVEESIX   D; VERSION 4.06 (05/18/94); CORE VERSION 1.17 (12/21/93); COPYRIGHT (C) 1991, 1992, 1993,
1994
TRIMBLE NAVIGATION;*38<
>RLN77637250+304234730-0911442978+000013120187-000409640606D8260921DF17DB090C23A2000270000032;*3D<
>QLN<
>RLN77638250+304234651-0911442109+000012440188-000309660606D8260921DF17DB090C23A2000270000032;*32<
>QLN<
>RPV77640+3042345-0911440801909632;*7F<
>RLN77639750+304234528-0911440798+000011130195-000309690606D8260921DF17DB090C23A2000270000032;*35<
>QLN<
```

The .dc file contains DGPS data. The .ndc data contains DGPS data plus GPS data that, for one reason or another, could not be differentially corrected. As a result, the size of the .ndc file is always equal or larger than the size of the .dc file. The ratio of the size of the .dc file to the size of the .ndc file gives an indication of the percentage of GPS points that were differentially corrected. Under normal circumstances, such a ratio is usually larger than 95%. The format of both the .dc file and the .ndc file is the same. A sample of records from file 08171241.ndc follows:

```
77635.250 30.4234872 -91.1444724 18.9 3
77635.750 30.4234840 -91.1444287 18.9 3
77637.250 30.4234730 -91.1442978 18.7 3
77638.250 30.4234651 -91.1442109 18.8 3
77639.750 30.4234528 -91.1440798 19.5 3
77640.250 30.4234485 -91.1440347 20.0 3
77641.250 30.4234390 -91.1439400 21.1 3
```

The fields represent GPS time (in seconds), latitude, longitude, speed (in mph); and DGPS status, respectively.

Program cms-cvt was written in C++. The corresponding source code file, cms-cvt.cpp, follows:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
void fatal_error(char *);
void open_data(char *);
void process();
int get_string(char *);

FILE *infile, *outfile, *outfile2;

void main(int argc, char *argv[])
{
  if (argc <= 1) {
    fatal_error("Not enough arguments!\n");
  }
  open_data(argv[1]);
  process();
  fclose(infile);
  fclose(outfile);
  fclose(outfile2);
}

void open_data(char *fname)
{
  char
inname[100],outname[100],outname2[100];
  strcpy(inname,fname);
  strcpy(outname,fname);
  strcpy(outname2,fname);
  strcat(inname,".ltf");
  strcat(outname,".dc");
  strcat(outname2,".ndc");
  printf("Input file is %s, and output file
are %s and %s\n",inname,
  outname,outname2);
  if ((infile=fopen(inname,"r")) == NULL)
    fatal_error("Can not open input file");
  if ((outfile=fopen(outname,"w")) == NULL)
    fatal_error("Could not open output
file");
  if ((outfile2=fopen(outname2,"w")) == NULL)
    fatal_error("Could not open output file
    2");
}

void process()
{
  char buf_read[1000],buf[1000];
  int done;
  int str_len;
  unsigned int no_sv,source,age;
  double time,lat,lon,speed;

  done=0;
  while (!done) {
    done=get_string(buf_read);
    if ((!done) &&
(strncmp(buf_read,">RLN",4)
    == 0) &&
    (strlen(buf_read) >= 82)) {

      strncpy(buf,&buf_read[55],2);
      buf[2]=NULL;
      no_sv=(unsigned int) atoi(buf);

      buf[0]=buf_read[67+4*no_sv];
      buf[1]=NULL;
      source=(unsigned int) atoi(buf);

      buf[0]=buf_read[68+4*no_sv];
      buf[1]=NULL;
      age=(unsigned int) atoi(buf);

      if (strstr(buf_read,";ID=") == NULL)
        str_len=74;
      else
        str_len=82;
      if (((strlen(buf_read) ==
        str_len+4*no_sv) ||
        (strlen(buf_read) ==
str_len+4*no_sv-
        4)) && (age == 2)) {
        strncpy(buf,&buf_read[4],5);
        buf[5]='.';
        strncpy(&buf[6],&buf_read[9],3);
        buf[9]=NULL;
        time=atof(buf);

        strncpy(buf,&buf_read[12],3);
        buf[3]='.';
        strncpy(&buf[4],&buf_read[15],7);
        buf[11]=NULL;
        lat=atof(buf);

        strncpy(buf,&buf_read[22],4);
        buf[4]='.';
        strncpy(&buf[5],&buf_read[26],7);
        buf[12]=NULL;
        lon=atof(buf);

        strncpy(buf,&buf_read[42],3);
        buf[3]='.';
        strncpy(&buf[4],&buf_read[45],1);
        buf[5]=NULL;
        speed=atof(buf);

        fprintf(outfile2,"%8.3lf %11.7lf
          %12.7lf %5.1f %lu\n",
        time,lat,lon,speed,source);
        if ((source == 2) || (source == 3))
          fprintf(outfile,"%8.3lf %11.7lf
            %12.7lf %5.1f %lu\n",
          time,lat,lon,speed,source);
      }
    }
  }
}

void fatal_error(char *errorstring)
{
  printf("%s\n",errorstring);
  exit(1);
}

int get_string(char *buf)
{
  int i;
  char c=0;

  while ((c != '>') && (!feof(infile)))
    c=getc(infile);
  if feof(infile)
    return(1);
  buf[0]='>';
  i=1;
  while ((c != '<') && (!feof(infile))) {
    c=buf[i++]=getc(infile);
    if (i >= 1000)
      fatal_error("Input string is too
long");
  }
  if (feof(infile))
    return(1);
  buf[i]=NULL;
  return(0);
}
```

shv-cvt conversion program

This program reads GPS data from an .asc file, and changes its format so that it conforms to the .ndc format produced by cms-cvt. This program was written to convert data produced with Pfinder, which is the Trimble software that NLCOG normally uses to postprocess GPS data. The program is run at the DOS prompt by typing

shv-cvt filename

where "filename" is the name of an .asc file. It is not necessary to type in ".asc". The application automatically assumes that "filename" has an .asc extension. A sample of records from file 08171241.ltf follows:

32.51369064,	-93.72292225,	61.5,	12:46:06
32.51368541,	-93.72264058,	60.1,	12:46:07
32.51368085,	-93.72236697,		12:46:08
32.51367568,	-93.72209847,	57.0,	12:46:09
32.51366716,	-93.72183101,	56.1,	12:46:10
32.51365387,	-93.72156492,	56.1,	12:46:11
32.51363891,	-93.72130258,	55.6,	12:46:12

The fields represent latitude, longitude, speed (mph) , and GPS time (in hh:mm:ss UTC). A sample of records from the converted file 08171241.dc follows:

```
45966.000 32.5136906 -93.7229222 61.5 3
45967.000 32.5136854 -93.7226406 60.1 3
45969.000 32.5136757 -93.7220985 57.0 3
45970.000 32.5136672 -93.7218310 56.1 3
45971.000 32.5136539 -93.7215649 56.1 3
45972.000 32.5136389 -93.7213026 55.6 3
```

The fields represent GPS time (in seconds UTC), latitude, longitude, speed (mph), and DGPS status (assumed to be equal to 3, meaning that differential correction was achieved).

shv-cvt was written in C++. The corresponding source code program, shv-cvt.cpp, follows:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

void fatal_error(char *);
void open_data(char *);
void process();
int get_string(char *);

FILE *infile, *outfile;

void main(int argc, char *argv[])
{
  if (argc <= 1) {
    fatal_error("Not enough arguments!\n");
  }

  open_data(argv[1]);
  process();
  fclose(infile);
  fclose(outfile);
}
```

```
void open_data(char *fname)
{
  char inname[100],outname[100];
  strcpy(inname,fname);
  strcpy(outname,fname);

  strcat(inname,".asc");
  strcat(outname,".dc");
  printf("Input file is %s, and output file
is
    %s\n",inname,
  outname);
  if ((infile=fopen(inname,"r")) == NULL)
    fatal_error("Can not open input file");
  if ((outfile=fopen(outname,"w")) == NULL)
    fatal_error("Could not open output
file");
}

void process()
{
  char buf_read[1000],buf[1000];
  int done;
```

```
    char **ptr;
    unsigned int hour,min,sec;
    double time,lat,lon,speed;

    done=0;
    while (!done) {
      done=get_string(buf_read);

      strncpy(buf,&buf_read[4],20);
      buf[20]=NULL;
      sscanf(buf,"%lf",&lat);

      strncpy(buf,&buf_read[28],20);
      buf[20]=NULL;
      sscanf(buf,"%lf",&lon);

      strncpy(buf,&buf_read[54],10);
      buf[10]=NULL;
      speed=strtod(buf,ptr);

      if (*ptr != buf) {

        strncpy(buf,&buf_read[74],10);
        buf[10]=NULL;

   sscanf(buf,"%d:%d:%d",&hour,&min,&sec);
        time=(double) hour*3600+(double)
min*60+
            (double) sec;

        fprintf(outfile,"%8.3lf %11.7lf
%12.7lf
              %5.1f 3\n",
                time,lat,lon,speed);
      }
      else {
        printf("No speed string found\n");
      }
    }
}

void fatal_error(char *errorstring)
{
  printf("%s\n",errorstring);
  exit(1);
}

int get_string(char *buf)
{
  int i=0;
  char c=0;

  while ((c != '\n') && (!feof(infile))) {
    c=buf[i++]=getc(infile);
    if (i >= 1000)
      fatal_error("Input string is too
long");
  }
  if (feof(infile))
    return(1);
  buf[i]=NULL;
  return(0);
}
```

Data Reduction Application

The application written in MDL to interactively link each segment with the filtered GPS data is called cms2. Like other MDL applications, it runs within Microstation. In order to run properly, the database server and the underlying Oracle database must already be loaded. In addition to this, an input .ndc or .dc file must exist in a directory having a date for a name. For example, if the GPS data was collected on August 17, 1995, the directory under which the .ndc file resides should be called 950817. cms2 takes this string and assigns it to the GPSDATE field in table SEG_TRAVEL_TIME. Finally, an empty file must also exist in the same directory so that cms2 can write output data into it. By convention, all output files have had an .nas extension.

A summary of steps needed to run cms2 is provided below:

1. Load Microstation and open design file, say segm.dgn
2. Load database server by typing "mdl load server" at the Microstation prompt
3. Specify the underlying Oracle database by typing, say, "db=cms95/cms95" at the Microstation prompt
4. Load cms2 by typing "mdl load cms2" at the Microstation prompt. The cms2 graphical interface should appear on the screen
5. Type in "corr_segments" in the Table field
6. Click on Setup, and then on Input to select input .ndc file
7. Click on output to select output .nas file
8. Click on View to display all GPS points. Note that these points are not actually Microstation point features. They are displayed only for visualization purposes

9. Note where the GPS points first entered the network and zoom in around that area. Choose a comfortable zoom level so that there is no risk of selecting wrong segments. Having segments on the screen so that their actual size is between 1" and 2" seems to provide an adequate zoom level to most users.
10. Click on Rewind and then on Go to begin the data reduction process
11. Click on the first segment for which there is a complete GPS coverage. Occasionally, the probe vehicle enters the network in the middle of a segment. Since cms2 assumes by default that GPS data covers the segment completely, that first segment should be ignored for data reduction purposes
12. Immediately after clicking on the first segment, cms2 refreshes the view and GPS points appear on the screen. By default, the selected segment is displayed in the middle of the screen. The points linked to the segment are clearly delimited by two green dots located on both segment ends. Summary segment and travel information is also displayed on the graphical interface. Data displayed includes segment code, name, and length (in mi), starting time, ending time, travel time, and speed. Time is given in seconds, and speed is given in mph. The starting time is the time the probe vehicle enters the segment. The ending time is the time the probe vehicle leaves the segment. The difference between the two times is the travel time. Two values of speed are given: average speed and GPS speed. Average speed results from dividing the segment length by the travel time. GPS speed results from averaging the instantaneous speed values given for all GPS points associated with the segment. The ratio between the two speed values computed is an indication of how uniform speed is kept within the segment.
13. After selecting the first segment, the operation reduces to clicking on consecutive segments, one at a time. Each time a new segment is selected, the application refreshes the view, displays updated GPS points, and displays summery segment and travel information on the graphical interface. Since cms2 is an event-driven application fully compatible with Microstation, standard Microstation tools such as zooming and windowing tools can also be executed any time. This way the display can be adjusted so that it fits the needs of the data reduction process. Two scroll bar features were added to aid in this process. The first scroll bar feature changes the size of the search radius. This is useful for situations where differential correction was not achieved and, as a result, there was a considerable data scatter. In this case, enlarging the search radius is advantageous. Modifying the size of the search radius is also useful for situations where there are several GPS points near the end of a segment and it is required to narrow down the number of GPS points so that a meaningful time stamp can be associated with the end of the segment. In this case, reducing the search radius is needed. The second scroll bar feature modifies the number of GPS points displayed on the screen. This is useful every time the data reduction process approaches a physical discontinuity involving a highway split. By momentarily displaying up to 200 points ahead (or 200 seconds ahead because GPS data is collected every one second), the user immediately knows what way to go next.
14. After playing back the entire input .ndc data file, click on Cancel, and use a text editor such as Notepad to check the output .nas file. Items to check include differences between the two values of speed recorded; occasional extremely high speed values; and empty records.
15. Import segment travel time data into Oracle using the SQL loader utility in the Database Administration Tools group. SQL Loader needs the following information: database user

name (for example, cms95) and password (for example, cms95), database name (for example, CMS-TD3), and a control file (for example, c:\sqlload\btr.ctl). This control file contains basic information such as .nas file name, table name, column names, and field separators. The following control file is used to import data into table SEG_TRAVEL_TIME:

```
LOAD DATA
INFILE 'g:\cmsbtr\gpsdata\96sum\960514\05141832.nas'
APPEND
INTO TABLE SEG_TRAVEL_TIME
FIELDS TERMINATED BY WHITESPACE
(SEGCODE,
GPSDATE DATE "YYMMDD",
STARTTIME,
VEHICLEID,
TRAVELTIME,
AVSPEED,
GPSSPEED)
```

The source code of the data reduction application, called cms2.mc, is provided below:

```
/*-------------------------------------------+
|   cms2.mc - Program for processing GPS data
+-------------------------------------------*/
/*-------------------------------------------+
|       Include Files
+-------------------------------------------*/
#include        <mdl.h>
#include        <global.h>
#include        <dlogbox.h>
#include        <cexpr.h>
#include        <cmdlist.h>
#include        <dlogman.fdf>  /* Dialog Box
Manager Function Prototypes  */
#include        <ctype.h>
#include        <mselems.h>
#include        <userfnc.h>
#include        <rscdefs.h>
#include        <tcb.h>
#include        <scanner.h>
#include        "cms2cmd.h"
#include        "cms2ids.h"
#include        "dblib.mc"

/*-------------------------------------------+
|       Private Global Variables
+-------------------------------------------*/
Private FILE    *dataFP = NULL, *outFP;
Private GlobalVars      globalVars;
char featureName[35], featureTable[15],
       columnName[15];
char *fname=featureName,
*ftable=featureTable,
       *cname=columnName;
void model3d_dialogHook ();
void locateTol_scrollBarHook ();
void advancedPnts_scrollBarHook ();
void publish_variables ();
int theme, textLevel = 63, featureLevel = 1;
int runMode, lTolerance, num = 0;
int advancedPnts;
unsigned long lastMSlink;
int segId = 0, preTolerance, prePointNum;
Dpoint3d   oldnode, oldgps, vCenter;
MSElementUnion endNode;
char targetDir[50];
char targetDrive[10];
char myDate[8];

Private DialogHookInfo uHooks[] =
  {
  {HOOKDIALOGID_Model3d, model3d_dialogHook},
  {HOOKITEMID_ScrollBar_locateTol,
    locateTol_scrollBarHook},
  {HOOKITEMID_ScrollBar_advancedPnts,
    advancedPnts_scrollBarHook},
  };

/*-------------------------------------------+
|       name    main
+-------------------------------------------*/
main
(
void
)
  {
  RscFileHandle       rfHandle;
  /*--- load our command table ---*/
  if (mdlParse_loadCommandTable (NULL) ==
NULL)
  mdlOutput_error ("Unable to load command
    table");

  mdlResource_openFile (&rfHandle, NULL,
    FALSE);

/*  Set up variables that will be evaluated
within C expression strings */
  publish_variables ();

  /*  Publish the dialog item hooks */
  mdlDialog_hookPublish
(sizeof(uHooks)/sizeof(DialogHookInfo),uHooks
);

  /*  open the main dialog box   */

  mdlDialog_open (NULL, DIALOGID_Model3d);
  }

/*-------------------------------------------+
|  model3d_dialogHook --- Unload the
model3d.ma
+-------------------------------------------*/
Private void model3d_dialogHook
(
DialogMessage   *dmP
)
```

```
    {
    Point2d pointP;
    char string[15];
    int state;


    if (dmP->dialogId != DIALOGID_Model3d)
      return;

    dmP->msgUnderstood = TRUE;
    switch (dmP->messageType)
    {
    case DIALOG_MESSAGE_DESTROY:
      {
      mdlDialog_cmdNumberQueue (FALSE,
        CMD_MDL_UNLOAD,
        mdlSystem_getCurrTaskID(),TRUE);
      break;
      }
    case DIALOG_MESSAGE_CREATE:
      {
      dmP->u.create.interests.updates = TRUE;
      dmP->u.create.interests.mouses = TRUE;
      dmP->u.create.interests.keystrokes =
TRUE;
      dmP-
>u.create.interests.dialogFocuses=TRUE;
      dmP->u.create.interests.itemFocuses =
TRUE;
      dmP->u.create.interests.resizes = TRUE;
      break;
      }

    case DIALOG_MESSAGE_BUTTON:
      {
      break;
      }
    case DIALOG_MESSAGE_UPDATE:
      {
      DialogItem     *outFieldDiP;
      DialogBox      *db;

      break;
      }
    default:
      dmP->msgUnderstood = FALSE;
      break;
    }
    }

/*-------------------------------------------+
|   Distance tolerance_dialogHook
+-------------------------------------------*/
Private void locateTol_scrollBarHook
(
DialogItemMessage     *dimP
)
    {
    dimP->msgUnderstood = TRUE;

    switch (dimP->messageType)
    {
    case DITEM_MESSAGE_CREATE:
      lTolerance = 2500;
      mdlDialog_synonymsSynch(NULL,SYNONYMID_
        locationTol, NULL);
      preTolerance = lTolerance;
      break;

    case DITEM_MESSAGE_STATECHANGED:
      if (dimP->u.stateChanged.reallyChanged)
      {
      mdlParams_setActive (4,
ACTIVEPARAM_COLOR);
      mdlParams_setActive (1,
        ACTIVEPARAM_LINEWEIGHT);
      mdlEllipse_create(&endNode,NULL,&vCenter,
        preTolerance,preTolerance,NULL,0);
      mdlElement_display(&endNode,ERASE);
      mdlEllipse_create(&endNode,NULL,&vCenter,
        lTolerance,lTolerance,NULL,0);
      mdlElement_display(&endNode,NORMALDRAW);
      preTolerance = lTolerance;

      mdlDialog_synonymsSynch(NULL,
        SYNONYMID_locationTol,NULL);
      break;
      }
    default:
      dimP->msgUnderstood = FALSE;
      break;
    }
    }

/*-------------------------------------------+
|   advancedPnts scrollbar item hook
+-------------------------------------------*/
Private void advancedPnts_scrollBarHook
(
DialogItemMessage     *dimP
)
    {
    dimP->msgUnderstood = TRUE;

    switch (dimP->messageType)
    {
    case DITEM_MESSAGE_CREATE:
      {
      advancedPnts = 15;
      mdlDialog_synonymsSynch(NULL,
        SYNONYMID_advancedPnts,NULL);
      prePointNum = advancedPnts;
      break;
      }
    case DITEM_MESSAGE_STATECHANGED:
      {
      mdlDialog_synonymsSynch(NULL,
        SYNONYMID_advancedPnts,NULL);
      drawAdvancedPoints ();
      prePointNum = advancedPnts;
      break;
      }
    default:
      dimP->msgUnderstood = FALSE;
      break;
    }
    }

/*-------------------------------------------+
| name   waitForProcessing
+-------------------------------------------*/
Private int waitForProcessing
(
int position
)
    {
    mdlInput_sendResume (position);
    mdlInput_waitForMessage ();
    }

/*-------------------------------------------+
| name openInputFile
+-------------------------------------------*/
Private void openInputFile
(
void
)
    {
    int   status, stringLength, currLength;
    char  dir[MAXDIRLENGTH];

    DialogItem     *outFieldDiP;
    DialogBox      *db;

    while ((dataFP = fopen
(globalVars.fileName,
      "r")) == NULL)
      {
      mdlFile_getcwd (dir, MAXDIRLENGTH);
```

```
      stringLength = strlen (dir);
      dir[stringLength] = '/';
      dir[stringLength+1] = 0;

      if (mdlDialog_fileOpen
         (globalVars.fileName, NULL, 0, NULL,
         "*.*", dir, "File Manager Window"))
         {
         mdlState_startDefaultCommand ();
         return;
         }

         db=mdlDialog_find(DIALOGID_Setup,
NULL);
         outFieldDiP =
         mdlDialog_itemGetByTypeAndId(db,
            RTYPE_Text, TEXTID_GpsFile, 0);
         mdlDialog_itemSetValue (NULL,0,NULL,
            globalVars.fileName,db,
            outFieldDiP->itemIndex);
      }

   db=mdlDialog_find(DIALOGID_Setup, NULL);
   mdlFile_parseName
(globalVars.fileName,NULL,
      targetDir,NULL,NULL);
   currLength = strlen(targetDir) - 7;
   strncpy (globalVars.date,
      targetDir+currLength, 6);
   sprintf (myDate, "%s", globalVars.date);
   outFieldDiP =
      mdlDialog_itemGetByTypeAndId(db,
         RTYPE_Text, TEXTID_Date, 0);
   mdlDialog_itemSetValue (NULL,0,NULL,
      globalVars.date,db,outFieldDiP-
>itemIndex);

   mdlFile_parseName (globalVars.fileName,
      targetDrive,NULL,NULL,NULL);

   strcpy (globalVars.fileName, "");
   }

/*-----------------------------------------------+
| name openOutputFile
+-----------------------------------------------*/
Private void openOutputFile
(
void
)
   {
   DialogItem     *outFieldDiP;
   DialogBox      *db;
   char           dir[MAXDIRLENGTH];

   sprintf (dir, "%s:%s", targetDrive,
      targetDir);

   while ((outFP = fopen
(globalVars.fileName1,
      "a+")) == NULL)
   {

   if (mdlDialog_fileOpen
(globalVars.fileName1,
      NULL, 0, NULL, "*.*",
      dir, "Output File Specification"))
      {
      mdlState_startDefaultCommand ();
      return;
      }

      db=mdlDialog_find(DIALOGID_Setup, NULL);
      outFieldDiP =
mdlDialog_itemGetByTypeAndId
   (db, RTYPE_Text, TEXTID_outputFile, 0);
      mdlDialog_itemSetValue (NULL,0,NULL,
         globalVars.fileName1,db,
         outFieldDiP->itemIndex);
   }
   strcpy (globalVars.fileName1, "");
   }

/*-----------------------------------------------+
| name setSearchMask
+-----------------------------------------------*/
Private int      setSearchMask
(
void
)
   {
   static int   searchType[] = {TEXT_ELM};
   mdlLocate_noElemNoLocked ();
   mdlLocate_setElemSearchMask
   (sizeof(searchType)/sizeof(int),searchType)
;
   }

/*-----------------------------------------------+
| name locateTargetFeature
+-----------------------------------------------*/
Private int locateTargetFeature
(
Dpoint3d    vert[2],
int         numvert
)
   {
   Dpoint3d        dataPt;
   MSElementUnion  newpt;
   MSElementDescr  *ptDP;
   int             corner, status;
   char            textString[7];
   char            amslink[7];

   mdlLocate_init ();

   setSearchMask ();

   dataPt.x = vert[numvert-1].x;
   dataPt.y = vert[numvert-1].y;
   dataPt.z = 0;


   if (mdlLocate_findElement (&dataPt,
      tcb->lstvw,0,0,TRUE))
      {
      mdlText_extractString (textString,
dgnBuf);
      mdlOutput_printf(MSG_ERROR, "Text Found:
         %s", textString);
      if (runMode == 2)
      {
      vert[0].x = vert[0].x/3600000;
      vert[0].y = vert[0].y/3600000;
      vert[0].z = 0;
      fprintf
(dataFP,"%s,%f,%f,%f\n",textString,
         vert[0].x,vert[0].y,vert[0].z);
      }
      }
   }

/*-----------------------------------------------+
| name extractLine
+-----------------------------------------------*/
Private int      extractLine
(
MSElementUnion   *elP
)
   {
   int         numvert;
   Dpoint3d    vert[2];

   if ((mdlElement_getType (elP) != LINE_ELM)
||
      mdlLinear_extract (vert, &numvert, elP,
0))

      return MODIFY_STATUS_NOCHANGE;
```

```
   /*  print the origin to the message field
*/
   mdlElement_display (elP, HILITE);
   locateTargetFeature (vert, numvert);
   return MODIFY_STATUS_NOCHANGE;
   }

/*----------------------------------------+
| name    locateTraining
+----------------------------------------*/
Private void    locateTraining
(
void
)
   {
   ULong elemAddr[8000], eofPos, filePos;
   int scanWords,status,i,numAddr,temp,class;
   ExtScanlist   scanList;

   dataFP = fopen (fname,"w");

   mdlScan_initScanlist (&scanList);
   mdlScan_noRangeCheck (&scanList);
   scanList.scantype       = ELEMTYPE | LEVELS;
   scanList.extendedType   = FILEPOS;
   scanList.typmask[0]     = TMSK0_LINE;

   /*  set level to be searched */

   class = (textLevel - 1)/16;
   switch (class)
      {
      case 0:
         temp = pow(2,textLevel-1);
         scanList.levmask[0] = temp;
         break;
      case 1:
         temp = pow(2,textLevel-17);
         scanList.levmask[1] = temp;
         break;
      case 2:
         temp = pow(2,textLevel-33);
         scanList.levmask[2] =
            scanList.levmask[2] | temp;
         break;
      case 3:
         temp = pow(2,textLevel-49);
         scanList.levmask[3] =
            scanList.levmask[3] | temp;
         break;
      }

   eofPos = mdlElement_getFilePos
(FILEPOS_EOF,
      NULL);
   filePos = 0L;

   mdlScan_initialize (0, &scanList);

   do
      {
      scanWords =
sizeof(elemAddr)/sizeof(short);
      status =
mdlScan_file(elemAddr,&scanWords,
         sizeof(elemAddr),&filePos);
      numAddr = scanWords / sizeof(short);

      for (i=0; i<numAddr; i++)
         {
         if (elemAddr[i] >= eofPos)
         break;

         mdlModify_elementSingle
(0,elemAddr[i],
            MODIFY_REQUEST_NOHEADERS,
            MODIFY_ORIG, extractLine, NULL, 0L);
         }
      }while (status == BUFF_FULL);

   fclose (dataFP);

   mdlState_startDefaultCommand ();
   }

/*----------------------------------------+
| name shift_viewingArea
+----------------------------------------*/
Private void shift_viewingArea
(
double    x_center,
double    y_center
)

{
   Dpoint3d llcorner, center, extents,
range[2];
   RotMatrix rmatrix;
   double az;

   mdlView_getParameters(&llcorner, &center,
      &extents, &rmatrix, &az,
      statedata.inPoint.view);

   range[0].x = x_center - (extents.x / 2.0 );
   range[0].y = y_center + (extents.y / 2.0 );
   range[1].x = x_center + (extents.x / 2.0 );
   range[1].y = y_center - (extents.y / 2.0 );
   center.x = x_center;
   center.y = y_center;

   mdlView_setArea(statedata.inPoint.view,
      range, &center,az+0.1, az, &rmatrix);
   mdlView_updateSingle(statedata.inPoint.view
);
}

/*----------------------------------------+
| name         place_activePoint
+----------------------------------------*/
Private void  place_activePoint
(
Dpoint3d    *pnt,
int         wt,
int         co,
int         drawMode
)
   {
   Dpoint3d aPoint[2];
   MSElementUnion   gpspnt;

   aPoint[0].x = pnt->x;
   aPoint[0].y = pnt->y;
   aPoint[1].x = pnt->x;
   aPoint[1].y = pnt->y;
   mdlParams_setActive (wt,
      ACTIVEPARAM_LINEWEIGHT);
   mdlParams_setActive (co,
ACTIVEPARAM_COLOR);
   mdlLine_create(&gpspnt,NULL,aPoint);
   if (drawMode < 1)
      mdlElement_display (&gpspnt, ERASE);
   else
      mdlElement_display (&gpspnt, NORMALDRAW);
   }

/*----------------------------------------+
| name         dataButtonHit
+----------------------------------------*/
Private void    dataButtonHit
(
Dpoint3d    *point,
int         view
)
   {
   Dpoint3d    closestPoint;
   int        segment, numvert;
   Dpoint3d pnt[101], *pntP;
   Dpoint3d firstpnt,lastpnt, gpspnt, mypnt;
```

```
char      line[92];
char      tempStr[12];
char      gpsTime[10], oldTime[10],
          beginTime[10], gpsSpeed[6];
MSElementUnion  gps_dot, seg_node;
MSElementDescr  *lineDP;
double    tempVal1, tempVal2,dist1, dist2,
          dist3,dist0,dist00;
Ulong     filePos, myLink;
DialogItem *outFieldDiP;
DialogBox  *db;
double    segLength, travelTime, meanSpeed,
          speedSum;

if (!mdlLocate_findElement (point, view, 0,
  0, TRUE))
  mdlOutput_printf (MSG_ERROR, "Element not
    found");
else
  {
  getMslink (globalVars.ftName, &myLink);
  mdlOutput_printf (MSG_STATUS,"MS = %ld",
    myLink);

  sprintf (globalVars.dbLink, "%ld",
myLink);

  db=mdlDialog_find(DIALOGID_Model3d,
NULL);
  outFieldDiP =
    mdlDialog_itemGetByTypeAndId(db,
    RTYPE_Text, TEXTID_txtLevel, 0);
  mdlDialog_itemSetValue (NULL,0,NULL,
  globalVars.dbLink,db,
    outFieldDiP->itemIndex);

  mdlDB_readColumn (globalVars.segLength,
    globalVars.ftName,
myLink,"seglength");

  outFieldDiP =
    mdlDialog_itemGetByTypeAndId(db,
    RTYPE_Text, TEXTID_callOut, 0);
  mdlDialog_itemSetValue (NULL,0,NULL,
    globalVars.segLength,db,
    outFieldDiP->itemIndex);

  mdlDB_readColumn
    (globalVars.segName,globalVars.ftName,
    myLink, "segname");

  outFieldDiP =
mdlDialog_itemGetByTypeAndId
  (db, RTYPE_Text, TEXTID_ftLevel, 0);
  mdlDialog_itemSetValue (NULL,0,NULL,
    globalVars.segName,db,
    outFieldDiP->itemIndex);

  mdlDB_readColumn (globalVars.segType,
    globalVars.ftName, myLink, "segcode");

  outFieldDiP =
    mdlDialog_itemGetByTypeAndId
    (db, RTYPE_Text, TEXTID_segType, 0);
  mdlDialog_itemSetValue (NULL,0,NULL,
    globalVars.segType,db,
    outFieldDiP->itemIndex);

if (dgnBuf->ehdr.type == LINE_STRING_ELM |
  CMPLX_STRING_ELM)
  {
  filePos = mdlElement_getFilePos
    (FILEPOS_CURRENT,NULL);
  mdlElmdscr_read (&lineDP,
    filePos,0,FALSE,NULL);
  mdlElmdscr_stroke (&pntP, &numvert,
lineDP,
    0);
  firstpnt.x = (pntP[0]).x;
  firstpnt.y = (pntP[0]).y;
  firstpnt.z = 0.0;

  lastpnt.x = (pntP[numvert-1]).x;
  lastpnt.y = (pntP[numvert-1]).y;
  lastpnt.z = 0.0;

  shift_viewingArea (lastpnt.x, lastpnt.y);

  mdlElmdscr_display (lineDP,0, HILITE);
  mdlParams_setActive (3,
ACTIVEPARAM_COLOR);
  mdlParams_setActive (1,
    ACTIVEPARAM_LINEWEIGHT);

 mdlEllipse_create(&seg_node,NULL,&firstpnt,
    lTolerance,lTolerance,NULL,0);
  mdlElement_display(&seg_node,NORMALDRAW);
  mdlParams_setActive (1,
ACTIVEPARAM_COLOR);
  mdlEllipse_create(&seg_node,NULL,

 &lastpnt,lTolerance,lTolerance,NULL,0);
  mdlElement_display(&seg_node,NORMALDRAW);

  vCenter.x = lastpnt.x;
  vCenter.y = lastpnt.y;

  place_activePoint (&oldnode,6,5,1);
  place_activePoint (&oldgps,3,4,1);

  dist1 = 0.0;
  dist00 = 0.0;
  while (fgets(line,90,dataFP) != NULL)
  {
  strncpy (gpsTime, line, 9);
  gpsTime[9]='\0';
  strncpy (tempStr, line+10,11);
  tempStr[11]='\0';
  gpspnt.y = atof(tempStr)*3600000.0;
  strncpy (tempStr, line+22,12);
  tempStr[12]='\0';
  gpspnt.x = atof(tempStr)*3600000.0;
  gpspnt.z = 0.0;
  strncpy (gpsSpeed, line+35,5);
  gpsSpeed[5]='\0';

  dist2 = mdlVec_distance
(&lastpnt,&gpspnt);
  if (dist2 < 0.0000) dist2 = dist2 * -1.0;
  dist3 = mdlVec_distance (&mypnt,
&gpspnt);
  if (dist3 < 0.0000) dist3 = dist3 * -1.0;

 if (dist2<dist1 || dist1 > lTolerance ||
  dist1 == 0.0 || dist3 < 50.0)
  {
  place_activePoint (&gpspnt, 3, 4, 1);
  dist1 = dist2;
  speedSum = speedSum + atof (gpsSpeed);
  num = num + 1;

  if (segId < 1)
    {
    dist0 = mdlVec_distance (&firstpnt,
      &gpspnt);
    if (dist0 < 0.000) dist0 = dist0 * -
1.0;

    if (dist0 < dist00 || dist00 >
      lTolerance || dist00 == 0.0)
      {
      dist00 = dist0;
      sprintf (oldTime, "%s", gpsTime);
      }
    else
      {
      place_activePoint (&mypnt, 6, 5, 1);
      sprintf (beginTime,"%s",oldTime);
      segId = segId + 1;
      speedSum = atof (gpsSpeed);
```

```
        num = 1;
        }
      }
      mypnt.x = gpspnt.x;
      mypnt.y = gpspnt.y;
      sprintf (oldTime, "%s",gpsTime);
    }
    else
    {
    place_activePoint (&mypnt, 6, 5, 1);
    place_activePoint (&gpspnt, 3, 1, 1);
    sprintf (globalVars.startTime, "%s",
      beginTime);
    sprintf (globalVars.endTime, "%s",
      oldTime);

    outFieldDiP =
      mdlDialog_itemGetByTypeAndId(db,
      RTYPE_Text, TEXTID_startTime, 0);
    mdlDialog_itemSetValue (NULL,0,NULL,
      globalVars.startTime,db,
      outFieldDiP->itemIndex);

    outFieldDiP =
      mdlDialog_itemGetByTypeAndId
      (db,RTYPE_Text, TEXTID_Cname, 0);
    mdlDialog_itemSetValue(NULL,0,NULL,
      globalVars.endTime,db,
      outFieldDiP->itemIndex);

    segLength = atof (globalVars.segLength);
    travelTime = atof(globalVars.endTime) -
      atof(globalVars.startTime);
    meanSpeed = (3600.0 *
      segLength)/travelTime;
    sprintf(globalVars.tSpeed, "%7.4f",
      meanSpeed);
    meanSpeed = speedSum/num;
    sprintf(globalVars.aSpeed, "%7.4f",
      meanSpeed);

    sprintf(globalVars.inTable,"%9.3f",
      travelTime);

    outFieldDiP=
      mdlDialog_itemGetByTypeAndId
      (db,RTYPE_Text,TEXTID_Fname, 0);
    mdlDialog_itemSetValue(NULL,0,NULL,
      globalVars.inTable,db,
      outFieldDiP->itemIndex);

    outFieldDiP=
      mdlDialog_itemGetByTypeAndId
      (db,RTYPE_Text,TEXTID_Speed, 0);
    mdlDialog_itemSetValue
      (NULL,0,NULL,globalVars.tSpeed,db,
      outFieldDiP->itemIndex);

    outFieldDiP=
      mdlDialog_itemGetByTypeAndId
      (db,RTYPE_Text,TEXTID_aSpeed, 0);
    mdlDialog_itemSetValue
      (NULL,0,NULL,globalVars.aSpeed,db,
      outFieldDiP->itemIndex);

    fprintf(outFP,"%s %s %s %s %9.3f %s
%s\n",
      globalVars.segType, myDate, beginTime,
      globalVars.carID, travelTime,
      globalVars.tSpeed, globalVars.aSpeed);
    mdlOutput_printf (MSG_STATUS, "Date=%s",
      myDate);

    mdlUtil_beep (1);

    mdlOutput_printf (MSG_PROMPT,"Select the
      next segment");
    dist1 = 0.0;
    oldnode.x = mypnt.x;
    oldnode.y = mypnt.y;
      oldgps.x = gpspnt.x;
      oldgps.y = gpspnt.y;
      sprintf (beginTime, "%s", oldTime);
      num = 1;
      speedSum = atof (gpsSpeed);
      mdlParams_setActive (0,
        ACTIVEPARAM_LINEWEIGHT);
      break;
      }
      }  /*  End of while loop  */
      }
    else

      {
      mdlOutput_printf (MSG_ERROR,"Invalide
        Element Type");
      }
    drawAdvancedPoints();
    }
    }
/*-----------------------------------------+
|   Draw advanced points function
+-----------------------------------------*/
Private void drawAdvancedPoints
(
)

   {
   char        line[92];
   Dpoint3d    pathpnt;
   char        tempStr[13];
   ULong       recPos;
   int         nextp;

   if (dataFP != NULL)
   {
   recPos = ftell (dataFP);
   for (nextp = 0; nextp < prePointNum; nextp
=
     nextp + 1)
     {
     fgets(line,90,dataFP);
     strncpy (tempStr, line+10,11);
     tempStr[11]='\0';
     pathpnt.y = atof(tempStr)*3600000.0;
     strncpy (tempStr, line+22,12);
     tempStr[12]='\0';
     pathpnt.x = atof(tempStr)*3600000.0;
     pathpnt.z = 0.0;

     place_activePoint (&pathpnt, 3, 1, 0);
     }
   fseek (dataFP, recPos, SEEK_SET);

   for (nextp = 0; nextp < advancedPnts; nextp
=
     nextp + 1)
     {
     fgets(line,90,dataFP);
     strncpy (tempStr, line+10,11);
     tempStr[11]='\0';
     pathpnt.y = atof(tempStr)*3600000.0;
     strncpy (tempStr, line+22,12);
     tempStr[12]='\0';
     pathpnt.x = atof(tempStr)*3600000.0;
     pathpnt.z = 0.0;

     place_activePoint (&pathpnt, 3, 1, 1);
     }
   fseek (dataFP, recPos, SEEK_SET);
   }
   }
/*-----------------------------------------+
| name getSegment - print out element type of
| elements pointed to by user.
+-----------------------------------------*/
cmdName     getSegment
(
void
```

```
}
cmdNumber CMD_GETSEGMENT
  {
  mdlState_startPrimitive (dataButtonHit,
    getSegment, 0, 0);

  /* reset the location logic */
  mdlLocate_init();

  /* allow any element */
  mdlLocate_allowLocked();

  /* change cursor to be the "locate" cursor
*/
  mdlLocate_setCursor();
  }

/*-------------------------------------------+
|   Show GPS Path Command
+-------------------------------------------*/
Public cmdName toSelect
(
void
)
cmdNumber   CMD_GETFILE
  {
  char             line[92];
  char             tempStr[12];
  MSElementUnion   gps_dot;
  Dpoint3d         gpspnt, updateBox[2],
prePnt;
  int              dcnum, udnum;
  double           speed;

  udnum = 0;
  while (fgets(line,90,dataFP) != NULL)
    {
    udnum = udnum + 1;
    strncpy (tempStr, line+10,11);
    tempStr[10]='\0';
    gpspnt.y = atof(tempStr)*3600000.0;
    strncpy (tempStr, line+22,12);
    tempStr[11]='\0';
    gpspnt.x = atof(tempStr)*3600000.0;
    gpspnt.z = 0.0;
    strncpy (tempStr, line+35, 5);
    tempStr[5]='\0';
    speed = atof (tempStr);
    strncpy (tempStr, line+41,2);
    tempStr[2]='\0';
    dcnum = atoi(tempStr);

    if (dcnum < 2)
      {
      place_activePoint (&gpspnt, 1, 5, 1);
      }
    else
      {
      place_activePoint (&gpspnt, 2, 3, 1);
      }
    } /* End of while loop */
  }

/*-------------------------------------------+
|   Skip Record Command
+-------------------------------------------*/
Public cmdName SkipRecord
(
void
)
cmdNumber   CMD_PUTSEGMENT
  {
  segId = 0;
  mdlUtil_beep (3);
  mdlOutput_printf (MSG_STATUS, "Switched to
    SKIP Mode.");
  }

/*-------------------------------------------+
|   Rewind File Command
+-------------------------------------------*/
Public cmdName RewindFile
(
void
)
cmdNumber   CMD_MODEL
  {
  if (dataFP == NULL)
    mdlOutput_printf (MSG_STATUS, "No GPS
file
    has been opened yet!");
  else
    {
    rewind (dataFP);
    mdlOutput_printf (MSG_STATUS, "GPS file
has
    been rewinded!");
    segId = 0;
    }
  mdlUtil_beep (5);
  }

/*-------------------------------------------+
|   SETUP Command
+-------------------------------------------*/
Public cmdName Setup
(
void
)
cmdNumber   CMD_SETUP
  {
  int   lastAction;
  DialogBox     db;
  DialogItem    *outFieldDiP;

  db = mdlDialog_openModal (&lastAction,
NULL,
      DIALOGID_Setup);
  }

/*-------------------------------------------+
|   PICK Command
+-------------------------------------------*/
Public cmdName toPick
(
void
)
cmdNumber   CMD_PICKFILE
  {
  strcpy (globalVars.fileName, "");
  openInputFile ();
  }

/*-------------------------------------------+
|   SET Command
+-------------------------------------------*/
Public cmdName toSet
(
void
)
cmdNumber   CMD_DEFINEFILE
  {
  openOutputFile ();
  }

/*-------------------------------------------+
|   TOQUIT
+-------------------------------------------*/
Public cmdName toQuit
(
char     *unparsedP
)
cmdNumber   CMD_TOQUIT
  {
  mdlSystem_exit(NULL,1);
  }

/*-------------------------------------------+
| publish variables
+-------------------------------------------*/
```

```
void publish_variables()
{
char      *setP;

setP = mdlCExpression_initializeSet
   (VISIBILITY_DIALOG_BOX, 0, FALSE);
mdlDialog_publishBasicVariable
   (setP,&shortType,"textLevel",&textLevel);
mdlDialog_publishBasicVariable (setP,
   &shortType,"featureLevel",&featureLevel);
mdlDialog_publishBasicVariable
   (setP,&shortType,"lTolerance",&lTolerance);
mdlDialog_publishBasicVariable (setP,
   &shortType,"advancedPnts",&advancedPnts);
mdlDialog_publishBasicVariable(setP,&shortTyp
e,
   "theme",&theme);
mdlDialog_publishBasicPtr (setP,&charType,
"fname",&fname);
mdlDialog_publishBasicPtr (setP, &charType,
   "ftable",&ftable);
mdlDialog_publishBasicPtr (setP,&charType,
   "cname",&cname);
mdlDialog_publishComplexVariable(setP,
   "globalvars","globalVars",&globalVars);

/* strcpy
(globalVars.ftName,"corr_segments");
*/
}
```

APPENDIX C: TRAVEL TIME DATA AGGREGATION THEORY

GPS Data Aggregation

Figure 20 shows the time-distance diagram of the probe vehicle as it traverses a segment of length L. The GPS equipment in the probe vehicle receives information from the constellation of satellites on a continuous basis and computes coordinates and speed values. Consider a procedure that defines the points that can be associated with the segment by searching the closest points to the segment entrance and exit. Since GPS points can occur anywhere along the segment, it is very unlikely that the cutoff points will coincide with the segment entrance and exit points. This is illustrated in figure 20 with GPS points P_0 and P_p. P_0 is closest to the segment entrance (detected when selected the previous segment), and P_p is closest to the segment exit. Segment travel time and average speed could be obtained by interpolating the time stamps of the two GPS points located immediately before and after the segment entrance, and the time stamps of the two GPS points located immediately before and after the segment exit. However, in order to perform a time interpolation, the location of the GPS points involved is also needed. Because of uncertainties regarding the effect of GPS point positional errors on the computation of segment travel time and speed, an alternate procedure, based on GPS speeds, is considered here. This procedure bypasses the need for interpolation.

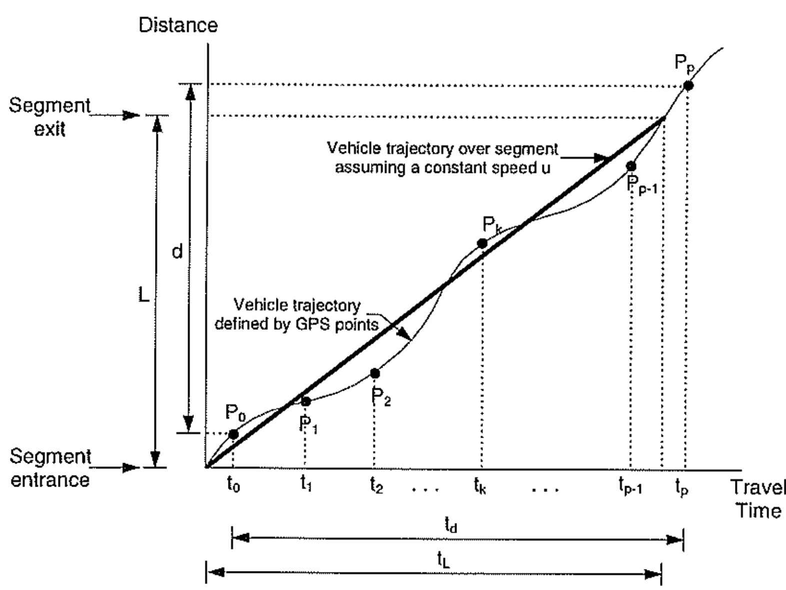


Figure 20
Time-distance diagram for GPS points on a segment

The alternate GPS speed procedure is based on the assumption that the GPS receiver used for the travel time study has the capability to record speeds in addition to coordinates and time. For the procedure to work properly, it is necessary that the GPS receiver calculate speeds independently of position fixes, for example by using pseudorange data (distance from satellite to receiver) and pseudorange rate data. A receiver such as the Trimble GPS Placer 400 complies with this requirement (according to information provided by telephone by Trimble officials). For speed computations, the receiver polls satellite data for a fraction of a second and, as a result, the computed speeds are almost instantaneous. In the case of the Trimble GPS Placer 400, which was the receiver used for this research, the specified accuracy of speed measurements is 0.1 mph (1 sigma) *[18]*.

In figure 20, let the instantaneous speeds associated with points P_0 to P_p be v_0 to v_p, respectively. The total distance covered by the probe vehicle between t_0 and t_p is

$$d = \int_{t_0}^{t_p} v dt \approx v_0\left(\frac{t_1 - t_0}{2}\right) + \left[\sum_{k=1}^{p-1} v_k\left(\frac{t_{k+1} - t_{k-1}}{2}\right)\right] + v_p\left(\frac{t_p - t_{p-1}}{2}\right) \tag{1}$$

The corresponding average speed u is

$$u = \frac{d}{t_d} = \frac{1}{t_d}\left\{v_0\left(\frac{t_1 - t_0}{2}\right) + \left[\sum_{k=1}^{p-1} v_k\left(\frac{t_{k+1} - t_{k-1}}{2}\right)\right] + v_p\left(\frac{t_p - t_{p-1}}{2}\right)\right\} \tag{2}$$

where t_d is travel time over the distance d. If GPS data is collected at regular time intervals Δt, equation (2) can be reduced to

$$u = \frac{\Delta t}{t_d}\left[\frac{v_0}{2} + \left(\sum_{k=1}^{p-1} v_k\right) + \frac{v_p}{2}\right] = \frac{1}{p}\left[\frac{v_0}{2} + \left(\sum_{k=1}^{p-1} v_k\right) + \frac{v_p}{2}\right] \tag{3}$$

If the variation of GPS speeds within the segment is small, v_0 should be very similar to v_p. In this case, equation (3) could be approximated to

$$u = \frac{1}{p}\sum_{k=1}^{p} v_k \tag{4}$$

Normally, the segment length L is known. If the initial and final GPS points associated with the segment are close to the entrance and exit points, respectively, d and L should be very similar. In this case, the mean speed u can be assumed to apply over the entire segment length and, as a result, the travel time t_L along the segment can be estimated as

$$t_L = \frac{L}{u} \tag{5}$$

If d and L are similar, t_d and t_L should also be similar.

Equations (4) and (5) are valid for a single run on a single segment and provide the necessary tools to transform a set of GPS point time stamp and speed values into a single pair of segment travel time and average speed values for the segment. In general, however, several runs involving several contiguous segments may be made *[4]*. In this case, it may be of interest to compute not only speed values for individual segments due to individual runs, but also representative speed values for each segment and for all segments combined. To keep the discussion general, the number of runs per segment is assumed to be different. This is particularly important if there are interchanges or intersections along the route and some segments have more records than other segments. Because of this, the procedure to compute representative speed values involves aggregating the data at the segment level first.

Let the number of samples per segment be m_i. The average travel time per segment is

$$\bar{t}_i = \frac{1}{m_i}\sum_{j=1}^{m_i} t_{L_j} = \frac{1}{m_i}\sum_{j=1}^{m_i}\frac{L_i}{u_j} = L_i \frac{1}{m_i}\sum_{j=1}^{m_i}\frac{1}{u_j} \tag{6}$$

where L_i is the length of each segment. Let the number of segments be n. The total distance L_T is

$$L_T = \sum_{i=1}^{n} L_i \tag{7}$$

The total travel time t_{T_L} over L_T is

$$t_{T_L} = \sum_{i=1}^{n}\left[L_i \frac{1}{m_i}\sum_{j=1}^{m_i}\frac{1}{u_j}\right] \tag{8}$$

The average speed for all runs over L_T is

$$\bar{u}_L = \frac{L_T}{t_{T_L}} \tag{9}$$

Equation (9) can be rewritten as

$$\bar{u}_L = \frac{1}{\sum_{i=1}^{n}\left[\frac{L_i}{L_T}\frac{1}{m_i}\sum_{j=1}^{m_i}\frac{1}{u_j}\right]} \tag{10}$$

This equation represents a weighted harmonic mean, where the weight is the ratio of the length of each segment to the total length considered. If all segments have the same length, the weight associated with each segment will be the same.

Notice that equations (8) and (10) are general in the sense that they can be used either for one or several runs, and either for one or several contiguous segments. To check their accuracy, an alternate formulation based on t_d values (figure 20) is also provided. For m_i samples per segment and n segments, it can be shown that the total travel time t_{T_d} is

$$t_{T_d} = \sum_{i=1}^{n}\left[\frac{1}{m_i}\sum_{j=1}^{m_i} t_{d_j}\right] \tag{11}$$

An approximate value of the average speed for all runs is

$$\bar{u}_d = \frac{L_T}{t_{T_d}} \tag{12}$$

To illustrate the use of this methodology, three examples with data collected in Baton Rouge are used. Table 12 shows a few records for selected segments. Figure 4 shows the location of the segments considered. For simplicity, the data shown has already been reduced from GPS points to highway segments. The first example involves the computation of total travel time and average speed for a single run. The second example involves several runs. The third example involves entire corridors.

Table 12
Sample of records associated with selected segments in figure 4

Date	Segment Code	L (mi)	t_d (seconds)	u (Eqn. (4)) (mph)
06/25/95	12444	0.200	12.5	52.92
	12453	0.106	8.0	51.25
	12454	0.106	10.0	51.53
07/23/95	12444	0.200	11.0	60.86
	12451	0.104	7.0	58.52
	12450	0.104	5.5	59.08
	12463	0.200	12.5	60.64
	12464	0.200	11.5	62.21
07/25/95	12444	0.200	11.5	56.50
	12451	0.104	7.0	57.30
	12450	0.104	7.0	57.30
	12463	0.200	12.5	58.78
	12464	0.200	12.5	59.17
08/13/95	12444	0.200	11.5	61.27
	12453	0.106	6.0	59.94
	12454	0.106	7.0	58.38

Example 1

Find individual and total travel time and average speeds for the stretch of I-10 EB composed of segments 12444, 12453, and 12454 (figure 4) using data from the August 13, 1995 run shown in table 12. The number of segments n is 3. For segment 12444, using equation (5),

$$t_{T_L} = 3600\left[\frac{0.200}{61.27}\right] = 11.75 \text{ s}$$

For segments 12453 and 12454, the corresponding values are 6.37 and 6.54 seconds. These values compare well with the measured values of t_d (11.5, 6.0, and 7.0 seconds, respectively), as shown in table 12. Notice that all differences are less than one second, even though in relative terms they range from 2 to 6%. This is obviously due to the order of magnitude of the travel times considered. Using now equation (12), the resulting value of u_d for segment 12444 is

$$\bar{u}_d = \frac{0.200 * 3600}{11.5} = 62.61 \text{ mph}$$

For segments 12453 and 12454, the corresponding values of u_d are 63.69 and 54.57 mph. In relative terms, the difference between these speeds and those shown in table 12 is still between 2 and 6%. However, in absolute terms, the three values of u_d do not compare well with those of table 12. The first two values are higher while the third value is smaller. Judging from the values on table 12 (61.27, 59.94, and 58.38 mph) it can be inferred that the probe vehicle was gradually reducing speed as it traversed segments 12444, 12453, and 12454. A closer look at the original GPS records confirms this trend. However, their counterpart u_d speed values (62.61, 63.69, and 54.57 mph) would suggest that the probe vehicle initially accelerated and then decelerated. This is a clear indication that segment speed values based on the average of all GPS speeds associated with a segment are better than those based on segment length and only two GPS time stamp values.

As longer segments, or as more contiguous segments are considered, the difference between u_d and u_L and that between t_d and t_L should decrease. For example, for segments 12444, 12453, and 12454, using equation (7) to find the total length L yields

$$L = 0.200 + 0.106 + 0.106 = 0.412 \text{ mi}$$

From equation (8), the total travel time is

$$t_{T_L} = 3600\left[\frac{0.200}{61.27} + \frac{0.106}{59.94} + \frac{0.106}{58.38}\right] = 11.75 + 6.37 + 6.54 = 24.65 \text{ s}$$

By comparison, the total travel time using t_d values is

$$t_{T_d} = 11.5 + 6.0 + 7.0 = 24.5 \text{ s}$$

which is less than 1% different with respect to the 24.65 s figure. Now, from equation (9), the average speed is

$$\bar{u}_L = \frac{0.412 * 3600}{24.65} = 60.17 \text{ mph}$$

By comparison,

$$\bar{u}_d = \frac{0.412 * 3600}{24.5} = 60.5 \text{ mph}$$

Again, the difference between the two values of speed is very small (less than 1%).

Example 2

Find the average travel time and speed associated with the freeway section defined by the following segments: 12444, 12451, 12450, 12463, and 12464. The number of segments n is 5. The number of records per segment varies. From equation (7), the total length L is

$$L = 0.200 + 0.104 + 0.104 + 0.200 + 0.200 = 0.808 \text{ mi}$$

From equation (8),

$$t_{T_L} = 3600 * \left[\frac{0.200}{4}\left(\frac{1}{52.92} + \frac{1}{60.86} + \frac{1}{56.50} + \frac{1}{61.27}\right) + \frac{0.104}{2}\left(\frac{1}{58.52} + \frac{1}{56.85}\right)\right.$$

$$\left. + \frac{0.104}{2}\left(\frac{1}{59.08} + \frac{1}{57.30}\right) + \frac{0.200}{2}\left(\frac{1}{60.64} + \frac{1}{58.78}\right) + \frac{0.200}{2}\left(\frac{1}{62.21} + \frac{1}{59.17}\right)\right]$$

$$= 12.48 + 6.49 + 6.44 + 12.06 + 11.87 = 49.34 \text{ s}$$

From equation (9), the average speed is

$$\bar{u}_L = \frac{0.808 * 3600}{49.34} = 58.95 \text{ mph}$$

By comparison, when using equations (11) and (12),

$$t_{T_d} = \frac{12.5 + 10.97 + 11.5 + 11.5}{4} + \frac{7.0 + 6.5}{2} + \frac{5.5 + 7.0}{2} + \frac{12.5 + 12.5}{2} + \frac{11.5 + 12.5}{2}$$

$$= 11.62 + 6.75 + 6.25 + 12.5 + 12.0 = 49.12 \text{ s}$$

$$\bar{u}_d = \frac{0.808 * 3600}{49.12} = 59.21 \text{ mph}$$

In this case, the difference in travel time is 0.22 s and the difference in speed is 0.26 mph.

Example 3

An extension of Example 2 is the computation of total travel time and average speed for entire corridors. These results are useful when measuring global corridor characteristics. As an example, consider all existing records for Baton Rouge between 7:00 and 8:00 am between May and August, 1995. Table 13 shows a summary of travel time and average speed values for each direction of travel on the I-10, I-12, and Airline Hwy corridors in Baton Rouge. Notice that the differences between the two values of travel time and those between the two values of speed are very small.

Table 13
Total travel time and average speed for selected corridors in Baton Rouge (Summer 1995, 7:00-8:00 am data)

Corridor				Results Based on GPS Speed		Results Based on Travel Time		Residual
Name	Direction	Length (mi)	# Segments	t_{T_L} (min)	$\bar{u}_L$ (mph)	t_{T_t} (min)	$\bar{u}_t$ (mph)	$t_{T_L} - t_{T_t}$ (min)
I-10	EB	93	16.18	17:16	56.23	17:21	55.96	00:05
	WB	78	13.51	15:22	52.77	15:23	52.71	00:01
I-12	EB	97	17.56	18:05	58.23	18:03	58.34	00:02
	WB	96	17.62	23:13	45.53	23:07	45.72	00:06
Airline Hwy	SB	126	20.72	35:57	34.58	35:48	34.71	00:08
	NB	125	20.57	37:33	32.86	37:12	33.18	00:21

Effect of Using Different Segment Lengths

The examples described previously were based on a segmentation scheme that assumed a nominal spacing of 0.2 mi between checkpoints (or five checkpoints per mi). Most segments were fixed at 0.2 mi. However, since distances between contiguous physical discontinuities are seldom, if ever, exact multiples of 0.2 mi, a procedure was also implemented to avoid very short segments immediately after physical discontinuities. This explains the 0.106 mi figure associated with segments 12453 and 12454, and the 0.104 mi figure associated with segments 12451 and 12450, immediately after the I-10, I-12 Split (figure 4).

Segmentation schemes based on nominal spacing between checkpoints other than 0.2 mi are also possible. Shorter segments imply a larger number of segments but, at the same time, aggregated speeds which are closer to the original GPS point speeds. Conversely, longer segments imply a lesser number of segments but, at the same time, aggregated speeds which are farther apart from the original GPS point speeds. Clearly, there is a trade off between number of segments and accuracy. Such a trade-off depends on the spatial variability of traffic. For example, figure 21 shows several speed-space profiles along the I-10, I-12, and Airline Hwy corridors in Baton Rouge, during the 1995-1996 academic year. I-10 and I-12 are Interstate freeways with a variable number of lanes between 4 and 6. Airline Hwy is a four-lane signalized highway. These profiles are based on GPS data collected every one second. Table 14 shows the dates and time periods associated with each run selected. Notice that for traffic situations that do not involve a great deal of spatial variability (figure 21a), using long segments (1 mi or longer) may be sufficient. In contrast, for traffic situations involving a lot of fluctuations (figures 21b, c, and d), using short segments (0.2 mi or shorter) may be needed.

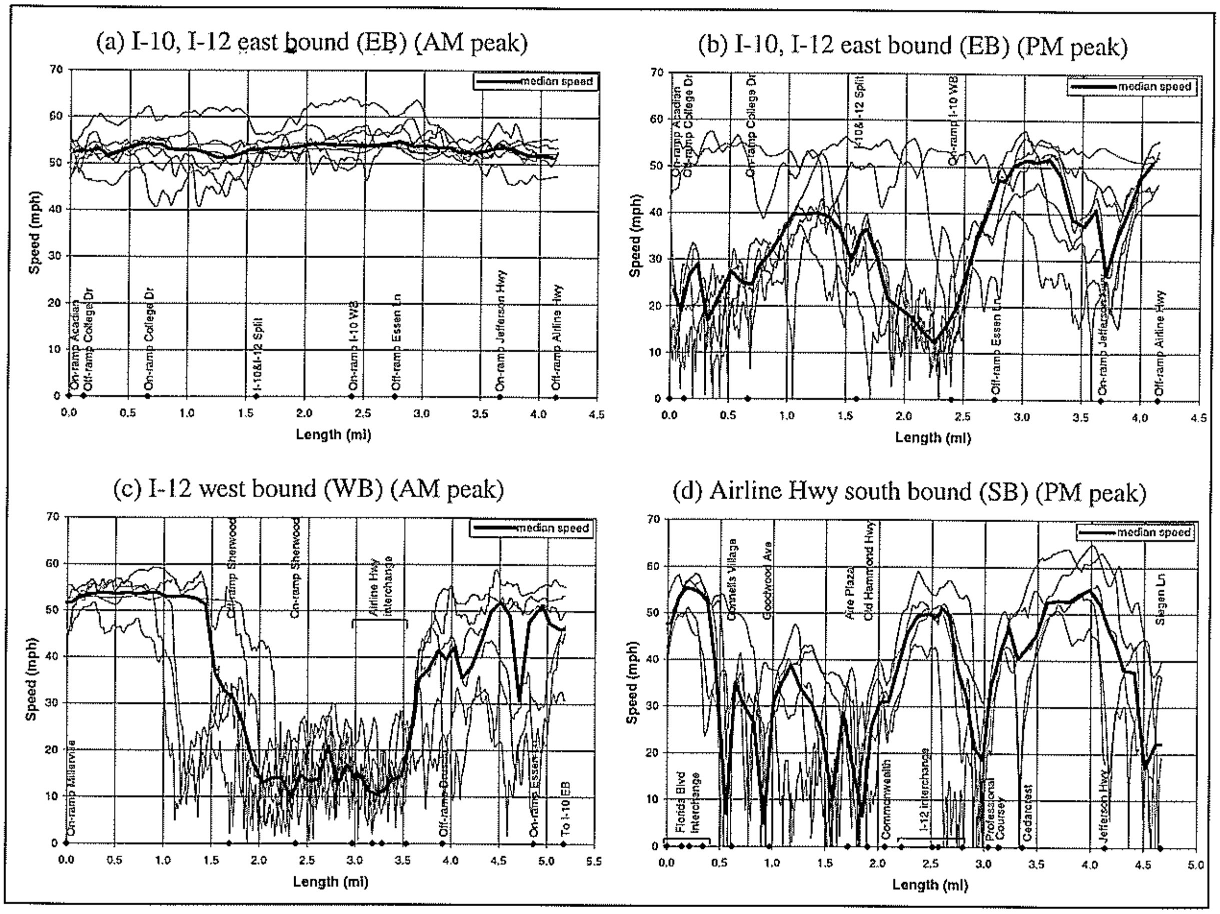


Figure 21
Space-speed profiles using GPS on selected corridors in Baton Rouge

Table 14
Dates and time periods associated with the runs shown in figure 21

Date	Start Time	End Time	Total Travel Time (min)	Date	Start Time	End Time	Total Travel Time (min)
(a) I-10, I-12 EB (AM peak)				**(b) I-10, I-12 EB (PM peak)**			
09/28/95	7:21:57 am	7:26:33 am	4:36	10/16/95	5:11:51 pm	5:21:33 pm	9:42
10/02/95	7:40:19 am	7:44:53 am	4:34	10/18/95	4:36:25 pm	4:46:11 pm	9:46
10/03/95	7:10:12 am	7:15:13 am	5:01	10/23/95	4:58:59 pm	5:10:10 pm	11:11
10/04/95	7:25:00 am	7:29:56 am	4:56	01/11/96	4:56:18 pm	5:09:39 pm	13:21
01/08/96	7:18:29 am	7:23:09 am	4:40	03/05/96	4:31:32 pm	4:36:12 pm	4:40
01/23/96	7:09:25 am	7:13:41 am	4:16	03/25/96	4:55:12 pm	5:03:16 pm	8:04
02/01/96	7:15:16 am	7:19:59 am	4:43	03/28/96	5:05:33 pm	5:11:43 pm	6:10
02/12/96	7:30:20 am	7:35:01 am	4:41				
(c) I-12 WB (AM peak)				**(d) Airline Hwy SB (PM peak)**			
09/28/95	7:02:19 am	7:15:02 am	12:43	11/28/95	4:47:00 pm	5:00:56 pm	13:56
10/02/95	7:21:13 am	7:35:01 am	13:48	02/06/96	5:11:18 pm	5:26:20 pm	15:02
10/03/95	7:06:54 am	7:28:42 am	21:48	03/07/96	4:54:03 pm	5:03:44 pm	9:41
10/04/95	7:07:57 am	7:19:28 am	11:31	03/26/96	4:44:40 pm	4:58:45 pm	14:05
01/23/96	7:40:15 am	7:54:00 am	13:45	04/26/96	4:39:20 pm	4:54:15 pm	14:55
02/01/96	7:33:53 am	7:52:49 am	18:56				
02/12/96	8:11:48 am	8:22:45 am	10:57				

To quantify the effect of using different segment lengths, each of the three stretches of highway referred to in figure 21 was segmented using the following segmentation schemes: 10 checkpoints per mi (or 0.1 mi segments), 5 checkpoints per mi (or 0.2 mi segments), 2 checkpoints per mi (or 0.5 mi segments), checkpoints at physical discontinuities (or 1 segment between physical discontinuities), 1 checkpoint every other mi (or 2 mi segments), and 1 checkpoint at each end of the route (or just one long segment). For each route and segmentation scheme, the data reduction application described previously was used to process the GPS data from all runs shown in table 14. Sets of speed-space profiles were then drawn to visually observe the aggregation effect. For every GPS point in a run, the absolute difference between the GPS point speed and the aggregated speed was computed. An average absolute difference and a corresponding standard deviation for all GPS points in a file were also computed.

Figure 22 shows the speed-space profiles for the PM peak runs on the I-10, I-12 EB corridor. For completeness, the median and harmonic mean speeds based on all seven runs are also included (a complete discussion on the use of the median speed vs. the harmonic mean speed is provided later). For visualization purposes, each profile was constructed by drawing points representing the speed values associated with the middle of all segments. Straight lines were then drawn to connect these points. This way, the problem of dealing with profiles composed of horizontal lines connected by vertical lines was significantly reduced.

A comparison between the profiles of figure 21b and those of figure 22 clearly indicates that the richness of the original GPS data was lost completely when using 2 mi segments (figure 22e) and one 4.15 mi segment (figure 22f). Even when using segments separated by physical discontinuities (figure 22d) and 0.5 mi segments (figure 22c), there was only a vague resemblance to the original GPS data. Both of them show the constraining effect due to the ramp from I-10 WB immediately after the I-10, I-12 Split (figure 4). However, the 0.5 mi segment profile is a bit better because it suggests a local effect due to the on-ramp at Jefferson Hwy. The 0.2 mi segment profile (figure 22b) further proves this effect by emphasizing the fact that traffic resumes 50-mph speeds immediately after the Jefferson Hwy on-ramp. This profile shows more detail in other areas as well. It suggests a little bit of improvement immediately after the I-10, I-12 Split and unstable flow at the College Dr interchange. The 0.1 mi profile further shows these effects.

Notice, however, that most gains in detail were achieved when using segments shorter than 0.5 mi segments, particularly when moving from 0.5 mi segments to 0.2 mi segments. Using 0.1 mi segments certainly contributed to improve the picture but such an improvement does not appear to be so significant. To better substantiate this point, a numerical test was conducted. First, the absolute differences between all GPS point speeds and the aggregated speeds were computed. Second, the arithmetic average and standard deviation of all absolute differences for each run were computed. Table 15 shows these values. Finally, plots of average differences as a function of average segment length for all runs were constructed (figure 23a). Notice that both average and standard deviation increase with segment length, but that such an increase tends to level off for segments larger than 0.5 mi. Conversely, it can be argued that differences between GPS speeds and aggregated speeds begin to decrease significantly only when segments are smaller than 0.5 mi. Notice also that the average and the standard deviation turned out to have similar values in most cases. This means that using longer segments not only results in larger differences with respect to actual instantaneous speeds but also in increased uncertainty with respect to the magnitude of such differences.

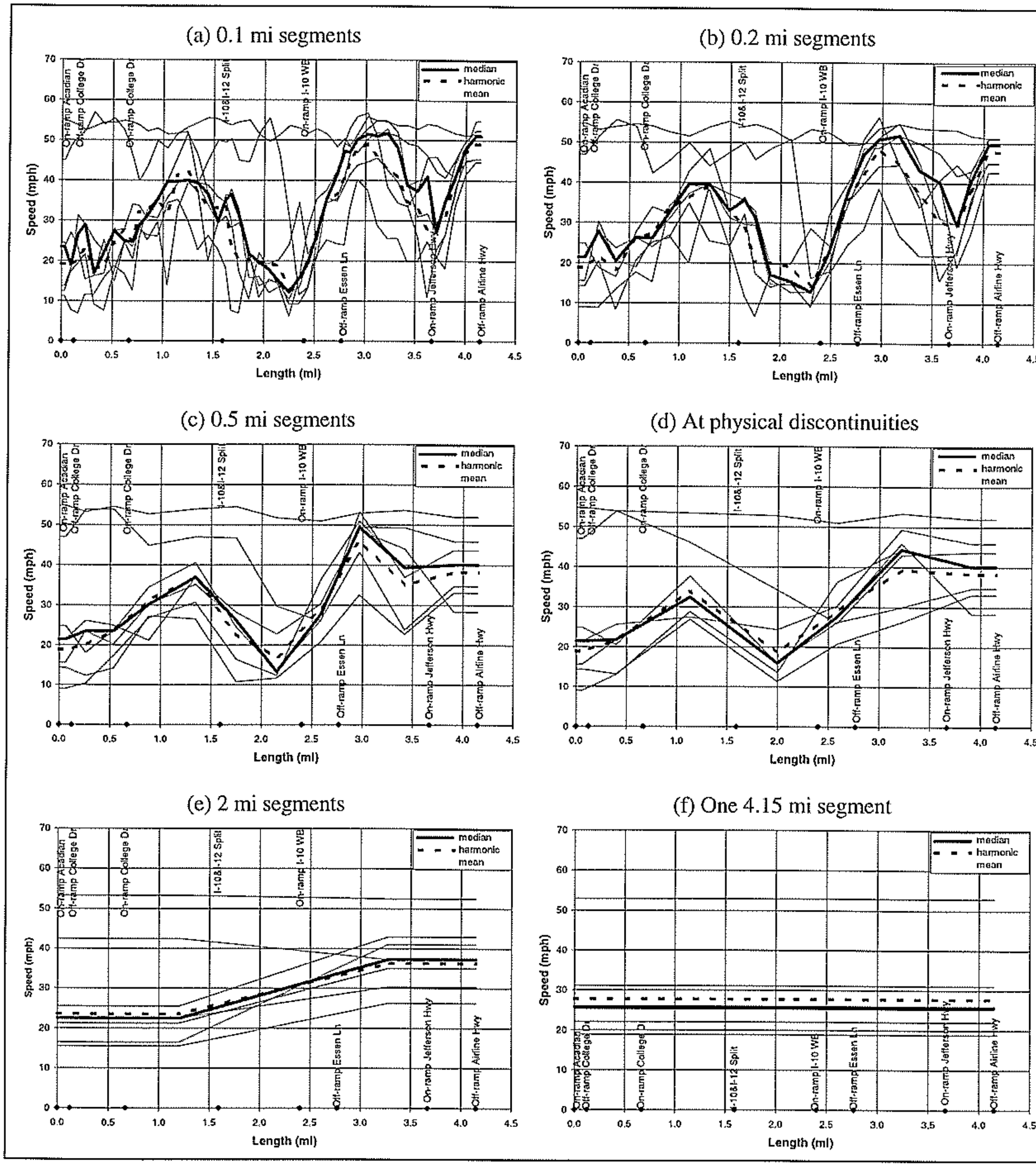


Figure 22
PM peak speeds on the I-10, I-12 Corridor using various segment lengths for aggregation

Table 15
Averages and standard deviations of differences between GPS speeds and speeds for various aggregation levels (in mph) - I-10, I-12 EB Corridor

Segmentation Level	Average Segment Length (mi)	Date														Overall	
		10/16/95		10/18/95		10/23/95		01/11/96		03/05/96		03/25/96		03/28/96			
		Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
0.1 mi segments	0.09	2.19	2.19	2.97	2.74	2.47	2.27	2.42	2.43	0.54	0.59	2.74	2.61	1.83	2.12	2.32	2.4
0.2 mi segments	0.17	3.20	3.12	3.43	3.10	3.28	2.85	3.10	2.70	0.78	0.83	3.62	3.46	3.57	3.44	3.14	3.03
0.5 mi segments	0.38	4.90	3.78	5.97	5.14	5.80	4.46	5.13	3.57	1.16	1.04	4.65	3.92	6.49	6.86	5.12	4.56
Physical discontinuities	0.59	5.72	4.17	7.34	5.44	6.28	4.71	5.27	3.90	1.32	1.11	5.58	4.72	8.07	6.92	5.86	4.96
2 mi segments	2.07	8.87	5.28	9.78	6.26	7.85	6.15	7.18	4.98	1.44	1.17	6.99	5.27	9.91	8.73	7.78	6.16
One 4.15 mi segment	4.15	10.19	6.62	10.35	6.65	12.20	7.53	8.16	5.89	1.50	1.18	9.97	6.52	10.50	8.48	9.49	7.10

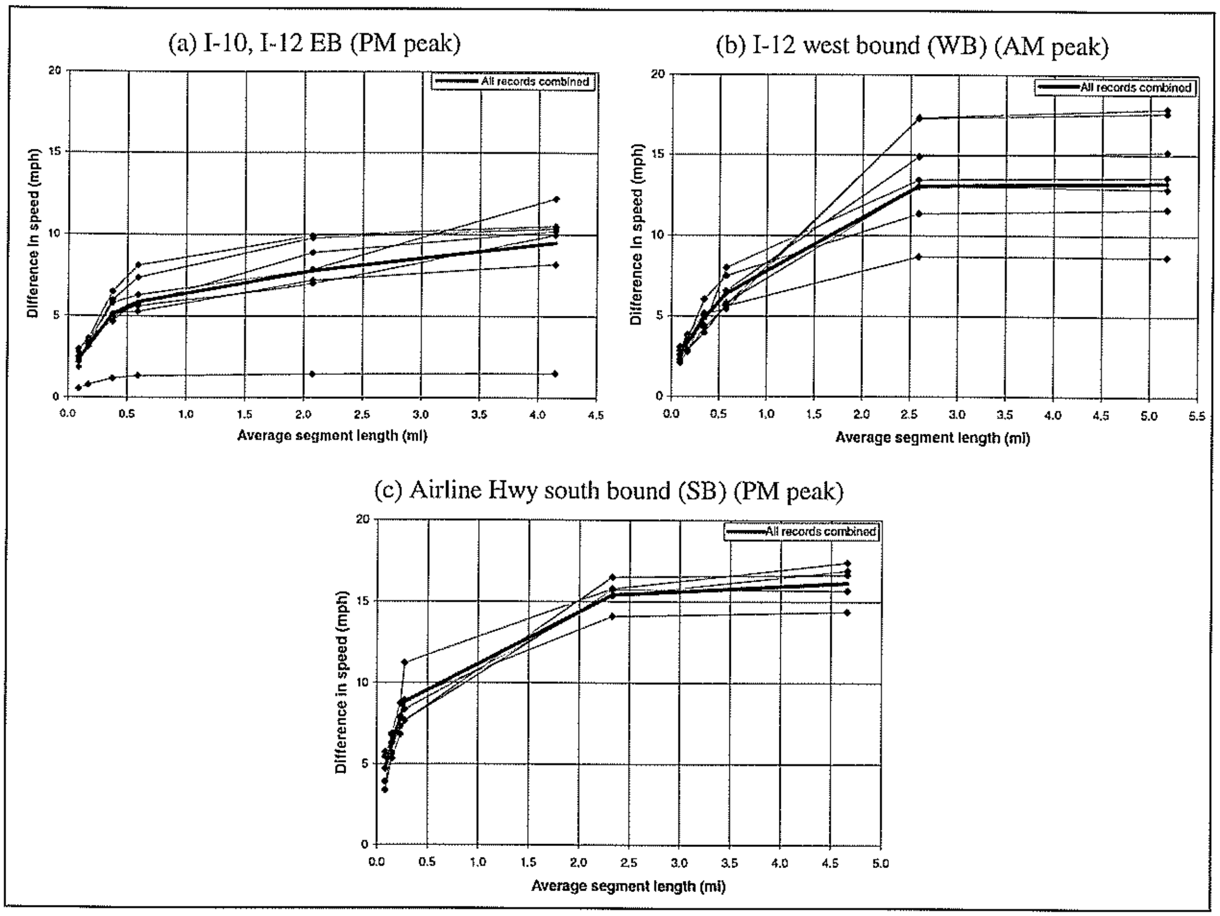


Figure 23
Average difference between GPS speeds and aggregated speeds as a function of average segment length

Figure 23b shows the average segment length vs. speed difference plot for the AM peak runs on the I-12 WB corridor. Figure 23c shows the corresponding plot for the PM peak runs on the Airline Hwy corridor. Trends are similar, despite obvious numerical differences between plots. Of interest of course is to compare the plots for average segment lengths of 0.5 mi or smaller. Notice that speed differences in figures 23a and 23b are very similar, both in shape and order of magnitude, and that speed differences in figure 23c are higher. For example, for an average segment length of 0.2 mi, figures 23a and 23b show speed differences of around 3 mph. In contrast, figure 23c shows speed differences around 7 mph. To achieve a speed difference level of less than 5 mph in figure 23c, an average segment length of 0.1 mi would have to be used. Notice also that figures 23a and 23b apply to two stretches of similar Interstate freeways while figure 23c applies to a signalized highway. This could suggest that smaller segments would be needed for a signalized highway than for a controlled access facility to achieve the same level of accuracy in the computation of speed.

Comparison between Harmonic Mean Speed and Median Speed

The harmonic mean speed procedure described previously provides a sound approach to the problem of defining representative speed values both at the segment and corridor levels. One disadvantage of using harmonic mean speeds is that they tend to be very sensitive to small value outliers. As shown in equation (10), since individual speed values appear in the denominator of the inner summation, small speed values carry a much heavier weight than larger values. As a result, outlying low speeds, which happen on atypically adverse traffic conditions, would result into very small harmonic mean speed values.

One possible solution to deal with this situation would be to manually reject records that are "atypical". However, with the introduction of intelligent transportation systems (ITS) that allow the collection of vast amounts of travel time and speed data it has become important to automatically obtain representative speed values without having to manually prune the data. One way to accomplish this is by using median speeds instead of harmonic mean speeds. The median is known for not being seriously affected by outliers and in many cases it is preferred by statisticians as a measurement of central tendency *[27]*. In many cases, however, the difference between median speed values and harmonic speed values may be quite noticeable. This is shown in figure 24a, which depicts a situation in which both congested and non-congested traffic conditions are included in the same speed distribution and for which there is no congestion most of the time. Notice in figure 24a that the median speed is larger than the harmonic mean speed. This is usually the case. Notice also that the difference between the two values can be significantly reduced by considering specific time periods for the analysis. This is shown in figure 24b.

The comparison between harmonic mean and median speed values can be generalized to include many more segments in the network. Figure 25 shows the distribution of differences between the two values of speed for nearly 1,900 segments in the Baton Rouge network. Figure 25a shows the distribution of differences for the 7:00-8:00 am time period during the 1995-1996 academic year. Figure 25b shows the distribution of differences for the 4:30-5:30 pm time period. The shape of both distributions is very similar. Only a few values are negative, showing that for most segments the median speed is larger than the harmonic mean speed. However, most differences are smaller than 2 mph. In fact, the mean difference for both time periods is very small: 2.11 and 1.59 mph, respectively. Furthermore, a simple test reveals that a difference value of zero is not significantly different from these average values at the 0.05

significance level. This means that for the great majority of segments the median speed can be used instead of the harmonic mean speed and still will produce essentially the same results.

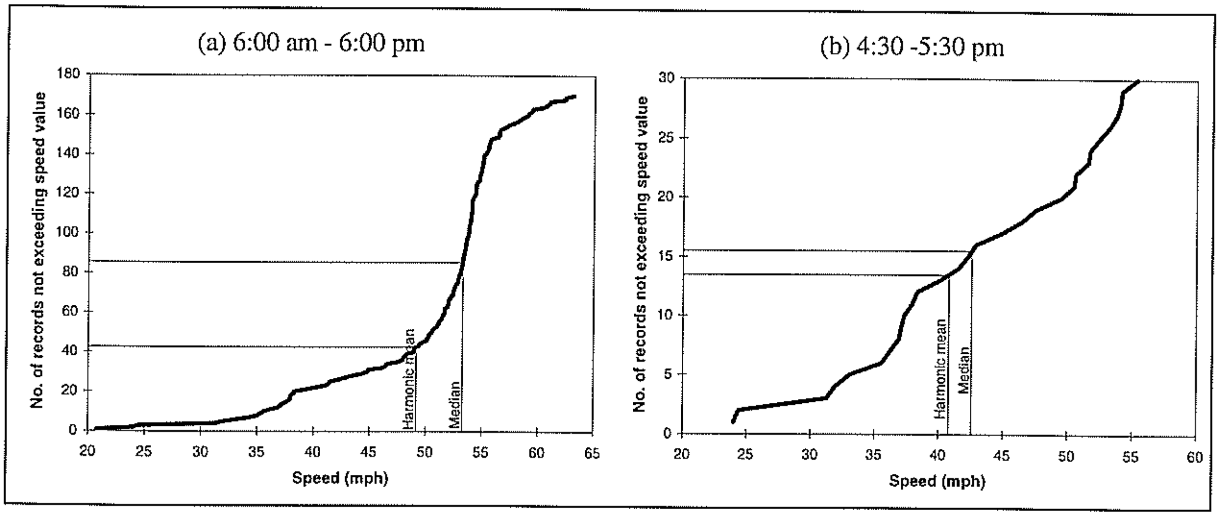


Figure 24
Speed distribution for segment 12444 (1995-1996 academic year)

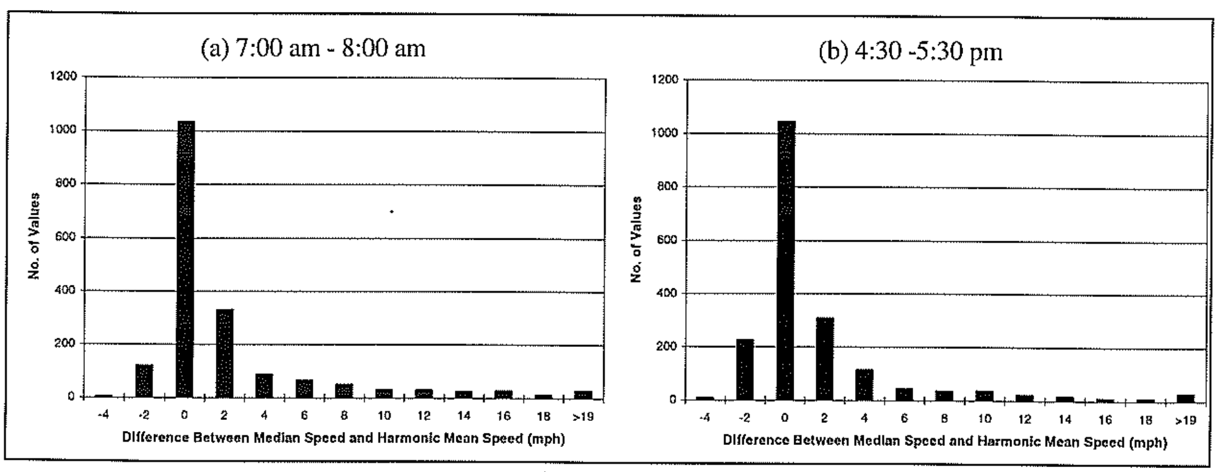


Figure 25
Distribution of differences between median speed and harmonic mean speed (1995-1996 academic year)

There are still some cases for which the difference between median speed and harmonic mean speed is just too great to be ignored. Table 16 shows the largest difference values observed during the 1995-1996 academic year, 4:30-5:30 pm time period, and the associated segment ID numbers. In all cases, differences are of the order of 30 mph. However, most low speed values were associated with incidents (stalled vehicles, accidents, or extremely heavy rain). Such very low speed values were largely responsible for the low harmonic mean speeds and, therefore, for the big difference between median and harmonic mean speeds. For comparison purposes, table 16 also shows the median, harmonic mean speeds, and their differences once the speed values associated with incidents are removed from the computation. Notice that for most segments harmonic mean speeds turned out to be much higher, while median speeds barely changed at all. The only segments that did not exhibit a significant reduction in speed difference were segments 13275 and 12271. Segment 13275 is located on Florida Blvd WB (inbound), just before the O'Neal Ln signalized intersection. Between 4:30 and 5:30 pm, the WB direction does not usually exhibit any congestion at all. Only two of the seven times the probe vehicle crossed this intersection while going west the stop light was on red causing the vehicle to stop. Because WB traffic is barely affected by the signalized intersection, then, it would make sense to use a speed estimator that closely resembles free flow conditions. The median speed serves this purpose. Segment 12271 is located on I-10 EB on the west part of town, halfway between LA 1 and the I-10&I-110 Split (figure 9). While this area is frequently congested, congestion becomes particularly severe as one drives closer to the I-10&I-110 Split. Segment 12271 is sort of a boundary segment after which congestion becomes severe. As a result, traffic behavior on segment 12271 is much more unstable than that of its neighbors and, consequently, it is more unpredictable. In this case, choosing between the median and the harmonic mean does not really make any significant difference. However, because the difference between median speeds and harmonic mean speeds for segments located just downstream of segment 12271 tends to be small, the argument can be made that the median speed approach is still better for segment 12271 because it does not involve executing a separate query to compute the harmonic mean speed value for just one segment.

Table 16
Largest differences between median speeds and harmonic mean speeds in Baton Rouge, Louisiana (1995-1996 academic year, 4:30-5:30 pm time period)

Segment	Functional Class	All records				After removing records associated with incidents			
		No. of records	Median Speed (mph)	Harm. Mean Speed (mph)	Difference (mph)	No. of records	Median Speed (mph)	Harm. Mean Speed (mph)	Difference (mph)
13275	Principal Arterial	7	51.33	23.57	27.76	7	51.33	23.57	27.76
12300	Interstate	17	52.84	24.81	28.03	15	53.88	48.57	5.32
12270	Interstate	17	44.34	16.00	28.34	12	52.18	48.11	4.07
13564	Principal Arterial	5	42.24	13.76	28.48	4	43.09	43.57	-0.49
12269	Interstate	15	49.18	20.08	29.10	11	50.29	48.07	2.22
12198	Interstate	13	49.63	19.61	30.02	10	50.96	50.48	0.49
12199	Interstate	13	52.06	21.73	30.33	11	53.07	53.73	-0.66
12271	Interstate	18	46.47	15.84	30.63	13	49.54	28.36	21.18
12268	Interstate	17	49.70	18.94	30.76	14	50.53	49.33	1.20

APPENDIX D: DATABASE QUERIES

The following queries are described in this appendix:

- Selection of records associated with a specific segment
- Computation of segment minimum, average, and maximum speed per date range and per time period
- Computation of segment median speed per date range and per time period
- Determination of free flow speeds
- Computation of segment travel time delay
- Computation of speed and travel time at the corridor level.

Selection of Records Associated with Specific Segments

Suppose that all records associated with segment 12444 are needed. The corresponding query would be:

```
SELECT    SEGCODE, GPSDATE, STARTTIME, GPSSPEED
FROM      SEG_TRAVEL_TIME
WHERE     SEGCODE = 12444;
```

As described later, this is a query used to retrieve records from the database using the WWW CMS home page. Suppose now that only the records from 7:00 am to 8:00 am during the academic year 1995-1996 are needed. The corresponding query would be:

```
SELECT    SEGCODE, GPSDATE, STARTTIME, GPSSPEED
FROM      SEG_TRAVEL_TIME
WHERE     (SEGCODE = 12444
          AND GPSDATE > '01-SEP-95' AND GPSDATE <= '28-OCT-95'
          AND STARTTIME > 43200 AND STARTTIME <= 46800)
OR        (SEGCODE = 12444
          AND GPSDATE > '28-OCT-95' AND GPSDATE <= '7-APR-96'
          AND STARTTIME > 46800 AND STARTTIME <= 50400)
OR        (SEGCODE = 12444
          AND GPSDATE > '7-APR-96' AND GPSDATE <= '3-MAY-96'
          AND STARTTIME > 43200 AND STARTTIME <= 46800);
```

Notice that the WHERE condition includes three date ranges to account for the change from daylight saving time to standard time and then back to daylight saving time. Notice also that time periods are specified in seconds UTC since midnight. For example, 46800 is the UTC equivalent to 7:00 am from October to April.

Suppose now that the total number of weekday records per segment for the same date range and time periods is needed. The corresponding query would be:

```
SELECT    SEGCODE, COUNT (*)
FROM      SEG_TRAVEL_TIME, WEEK_DAYS
WHERE     (SEG_TRAVEL_TIME.GPSDATE > '01-SEP-95' AND
          SEG_TRAVEL_TIME.GPSDATE <= '28-OCT-95'
          AND STARTTIME > 43200 AND STARTTIME <= 46800
          AND SEG_TRAVEL_TIME.GPSDATE = WEEK_DAYS.GPSDATE
          AND DAYWEEK <> 'SU')
```

```
OR          (SEG_TRAVEL_TIME.GPSDATE > '28-OCT-95' AND
            SEG_TRAVEL_TIME.GPSDATE <= '7-APR-96'
            AND STARTTIME > 46800 AND STARTTIME <= 50400
            AND SEG_TRAVEL_TIME.GPSDATE = WEEK_DAYS.GPSDATE
            AND DAYWEEK <> 'SU')
OR          (SEG_TRAVEL_TIME.GPSDATE > '7-APR-96' AND
            SEG_TRAVEL_TIME.GPSDATE <= '3-MAY-96'
            AND STARTTIME > 43200 AND STARTTIME <= 46800
            AND SEG_TRAVEL_TIME.GPSDATE = WEEK_DAYS.GPSDATE
            AND DAYWEEK <> 'SU')
GROUP BY    SEGCODE;
```

This query is needed in Baton Rouge to separate weekday records from Sunday records.

Computation of Minimum, Average, and Maximum Speed

Suppose that color coded maps containing minimum, average, and maximum speeds are needed. The first step would be to execute a query that contains these values (in Oracle). The second step would be to use GIS query tools to join segment attribute data to their location on the design map. Because results from the first query are needed as input to the second query, it would be necessary to assign a name to the first query. In Oracle, query naming is usually handled by creating a "virtual" table or view. Such a view is then included into the MGE project using Intergraph's relational interface system (RIS). Occasionally, however, RIS does not accept Oracle views and, as result, the user is forced to create actual tables in Oracle to bypass the problem. The upside of doing this is that the second query, which must be generated and executed using MGE query tools, runs faster because it is based on actual tables. The downside, of course, is that additional storage space must be allocated to the new table.

For simplicity, the expression "CREATE TABLE" is used here, as opposed to "CREATE VIEW". Suppose that speed and travel time information is needed for the PM peak period (weekdays) during the academic year 1995-1996. Because travel time is derived from GPS speed, the first query only needs to compute speed values. The corresponding query would be:

```
CREATE TABLE     ACAD9596_430_530_SPEEDS (SEGCODE, RCOUNT, MINSPEED, AVSPEED,
                 MAXSPEED)
AS     SELECT    SEGCODE, COUNT (*), MIN(GPSSPEED), 1/AVG(1/GPSSPEED),
                 MAX(GPSSPEED)
       FROM      SEG_TRAVEL_TIME, WEEK_DAYS
       WHERE     (SEG_TRAVEL_TIME.GPSDATE > '01-SEP-95' AND
                 SEG_TRAVEL_TIME.GPSDATE <= '28-OCT-95'
                 AND STARTTIME > 77400 AND STARTTIME <= 81000
                 AND SEG_TRAVEL_TIME.GPSDATE = WEEK_DAYS.GPSDATE
                 AND DAYWEEK <> 'SU')
       OR        (SEG_TRAVEL_TIME.GPSDATE > '28-OCT-95' AND
                 SEG_TRAVEL_TIME.GPSDATE <= '7-APR-96'
                 AND STARTTIME > 81000 AND STARTTIME <= 84600
                 AND SEG_TRAVEL_TIME.GPSDATE = WEEK_DAYS.GPSDATE
                 AND DAYWEEK <> 'SU')
       OR        (SEG_TRAVEL_TIME.GPSDATE > '7-APR-96' AND
                 SEG_TRAVEL_TIME.GPSDATE <= '3-MAY-96'
                 AND STARTTIME > 77400 AND STARTTIME <= 81000
                 AND SEG_TRAVEL_TIME.GPSDATE = WEEK_DAYS.GPSDATE
```

```
              AND DAYWEEK <> 'SU')
GROUP BY      SEGCODE;
```

Once table ACAD9596_430_530_SPEEDS is included in the MGE project using RIS, the next step involves using MGE tools to generate design files based on speed ranges, say from 0 to 20 mph, 20 to 30 mph, and so on. The actual procedure to do this is included in appendix E. Here, only the query to select segments within each speed range is discussed. In the process of generating universal list files (ULFs), MGE searches for graphical elements having the same MSLINK number as the MSLINK value associated with segments that fall within the speed range considered. For example, if the minimum speed range considered is 20-30 mph, the corresponding query would be

```
SELECT    MSLINK
FROM      CORR_SEGMENTS, ACAD9596_430_530_SPEEDS
WHERE     CORR_SEGMENTS.SEGCODE = ACAD9596_430_530_SPEEDS.SEGCODE
AND       MINSPEED > 20 AND MINSPEED <= 30;
```

MGE would store the results of this query in a file called, say, min2030.ulf. Similarly, if the average speed range considered is 20-30 mph, the corresponding query would be

```
SELECT    MSLINK
FROM      CORR_SEGMENTS, ACAD9596_430_530_SPEEDS
WHERE     CORR_SEGMENTS.SEGCODE = ACAD9596_430_530_SPEEDS.SEGCODE
AND       AVSPEED > 20 AND AVSPEED <= 30;
```

MGE would store the results of this query in a file called, say, av2030.ulf.

Computation of Median Speeds

Both Oracle and Access lack a function to compute median values. As a result, it became necessary to create a utility outside Oracle to replace the missing function. The utility reads data from a generic Oracle table that contains SEGCODE and SPEED. This generic table can be the result from a query that produces all GPS speed values for specific date ranges and time periods. Based on all speed values associated with a segment, it computes the corresponding median speed. Finally it creates a new Oracle table with the term "MEDIAN" appended to the original table name. The utility source code is included at the end of this appendix.

Suppose that median speeds for the PM peak period (weekdays) during the academic year 1995-1996 are needed. The query to generate the input Oracle table would be:

```
CREATE TABLE      ACAD9596_430_530 (SEGCODE, SPEED)
AS      SELECT    SEGCODE, GPSSPEED
        FROM      SEG_TRAVEL_TIME, WEEK_DAYS
        WHERE     (SEG_TRAVEL_TIME.GPSDATE > '01-SEP-95' AND
                  SEG_TRAVEL_TIME.GPSDATE <= '28-OCT-95'
                  AND STARTTIME > 77400 AND STARTTIME <= 81000
                  AND SEG_TRAVEL_TIME.GPSDATE = WEEK_DAYS.GPSDATE
                  AND DAYWEEK <> 'SU')
        OR        (SEG_TRAVEL_TIME.GPSDATE > '28-OCT-95' AND
                  SEG_TRAVEL_TIME.GPSDATE <= '7-APR-96'
                  AND STARTTIME > 81000 AND STARTTIME <= 84600
```

```
                AND SEG_TRAVEL_TIME.GPSDATE = WEEK_DAYS.GPSDATE
                AND DAYWEEK <> 'SU')
OR              (SEG_TRAVEL_TIME.GPSDATE > '7-APR-96' AND
                SEG_TRAVEL_TIME.GPSDATE <= '3-MAY-96'
                AND STARTTIME > 77400 AND STARTTIME <= 81000
                AND SEG_TRAVEL_TIME.GPSDATE = WEEK_DAYS.GPSDATE
                AND DAYWEEK <> 'SU');
```

The utility reads table ACAD9596_430_530, computes median speeds, and generate a new table called ACAD9596_430_530_MEDIAN.

Determination of Free Flow Speeds

There are three options for estimating free flow speeds: (1) based on posted speed limits; (2) based on runs conducted specifically for this purpose; or (3) based on maximum observed speeds. One problem with using posted speed limits is that motorists routinely drive faster that the posted speed limit if traffic conditions permit it. In the case of Baton Rouge, a set of runs were then made on selected Sunday mornings with the hope that more realistic free flow conditions or nearly free flow conditions would be measured. However, it was observed that some speeds during Sunday morning runs were actually smaller than some off peak weekday speeds. One possible explanation for this phenomenon is that drivers tend to be less aggressive on Sunday mornings and are therefore more willing to drive at speeds closer to the posted speed limits.

Assume that the Sunday morning runs are chosen to represent free flow speeds. Assume also that results from the query will be used to compute travel time delays. As a result, it is necessary to generate a view or table. The corresponding query would be

```
CREATE TABLE        FREE_FLOW_SPEEDS (SEGCODE, FFSPEED)
AS      SELECT      SEGCODE, MAX(GPSSPEED)
        FROM        SEG_TRAVEL_TIME, WEEK_DAYS
        WHERE       SEG_TRAVEL_TIME.GPSDATE = WEEK_DAYS.GPSDATE
        AND         DAYWEEK = 'SU';
        GROUP BY    SEGCODE;
```

Alternatively, assume that the maximum observed speeds, regardless of day of week, are chosen to represent free flow speeds. The corresponding query would be

```
CREATE TABLE        FREE_FLOW_SPEEDS (SEGCODE, FFSPEED)
AS      SELECT      SEGCODE, MAX(GPSSPEED)
        FROM        SEG_TRAVEL_TIME
        GROUP BY    SEGCODE;
```

Computation of segment travel time delay

Segment travel time delay can be computed using tables CORR_SEGMENTS, FREE_FLOW_SPEEDS, and one of the tables containing representative segment speed values such as ACAD9596_430_530_SPEEDS or ACAD9596_430_530_MEDIAN. For simplicity, assume that table ACAD9596 _430_530_SPEEDS is used. The basic relationships among tables needed to compute delay are shown in figure 26. First, free flow and representative travel times are computed. Then, free flow times are subtracted from representative travel times to compute delays.

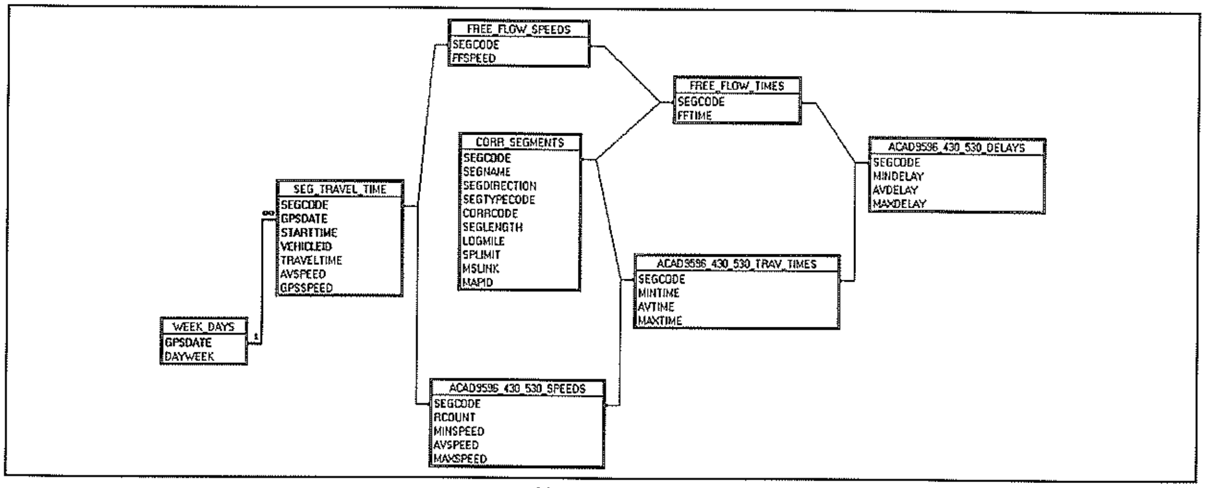


Figure 26
Database query schema

The query to compute free flow travel times can be expressed as

```
CREATE TABLE    FREE_FLOW_TIMES (SEGCODE, FFTIME)
AS      SELECT  FREE_FLOW_SPEEDS.SEGCODE, 3600*SEGLENGTH/FFPEED
        FROM    FREE_FLOW_SPEEDS, CORR_SEGMENTS
        WHERE   FREE_FLOW_SPEEEDS.SEGCODE=CORR_SEGMENTS.SEGCODE;
```

Notice that FFTIME is expressed in seconds. Now, the query to compute representative travel times can be expressed as

```
CREATE TABLE    ACAD9596_430_530_TRAV_TIMES (SEGCODE, MINTIME, AVTIME,
                MAXTIME)
AS      SELECT  ACAD9596_430_530_SPEEDS.SEGCODE,
                3600*SEGLENGTH/MAXSPEED, 3600*SEGLENGTH/AVSPEED,
                3600*SEGLENGTH/MINSPEED
        FROM    ACAD9596_430_530_SPEEDS, CORR_SEGMENTS
        WHERE   ACAD9596_430_530_SPEEDS.SEGCODE = CORR_SEGMENTS.SEGCODE;
```

The query to compute travel time delays can be expressed as

```
CREATE TABLE    ACAD9596_430_530_DELAYS (SEGCODE, MINDELAY, AVDELAY,
                MAXDELAY)
AS      SELECT  FREE_FLOW_TIMES.SEGCODE, MINTIME - FFTIME, AVTIME - FFTIME,
                MAXTIME - FFTIME
        FROM    FREE_FLOW_TIMES, ACAD9596_430_530_TRAV_TIMES
        WHERE   ACAD9596_430_530_TRAV_TIMES.SEGCODE =
                FREE_FLOW_TIMES.SEGCODE;
```

Computation of Speed and Travel Time at the Corridor Level

Occasionally it may be of interest to compute average speed and cumulative travel time for an entire corridor. Because segments follow directional centerlines, queries must take direction into account. Suppose that average speed and total travel time for the PM peak period (weekdays) during the academic year 1995-1996 is needed. Only segments located on the main routes are considered. In table CORR_SEGMENTS, these segments are characterized by having SEGTYPECODE = 1. The corresponding query would be

```
SELECT    CORRCODE, SEGDIRECTION, COUNT(*) "NRECORDS",
          SUM(SEGLENGTH) "LENGTH", SUM(AVTIME)/60 "TRTIME",
          SUM(SEGLENGTH)*3600/SUM(AVTIME) "SPEED"
FROM      CORR_SEGMENTS, ACAD9596_430_530_TRAV_TIMES
WHERE     ACAD9596_430_530_TRAV_TIMES.SEGCODE = CORR_SEGMENTS.SEGCODE
AND       SEGTYPECODE = 1
GROUP BY  CORRCODE, SEGDIRECTION;
```

Length is given in mi; travel time is given in minutes; and speed is given in mph.

Source Code of Utility to Compute Median Speeds

As mentioned previously, a utility outside Oracle, called median, was written to compute median speeds. The reason was that both Oracle and Access did not have a function to compute median values. The utility was written in C and, for simplicity, it was located on the /usr/local/ oracle/orahome/progs/ subdirectory of the Sun system at RSIP. It had embedded SQL subroutines to access the Oracle database over the network. Since some of these subroutines had already been developed for other purposes, it was decided to use and adapt existing resources in the Sun system to develop a working utility quickly rather than having to recreate the entire utility from scratch using a PC-based developing tool such as C++. In other computer systems, there are two options to deal this situation: (1) either a similar application is developed, taking into account local database and network characteristics; or (2) an external application such as Excel is used to compute median speeds.

The utility is run by typing

```
median tablename
```

at the /usr/local/oracle/orahome/progs/ prompt. For example, if the input table name is ACAD9596_430_530, the output table name will be ACAD9596_430_530_MEDIAN.

The source code of the utility, called median.pc, is provided below:

```
#include <stdio.h>
#include <sqlca.h>

#define UNAME_LEN      20
#define PWD_LEN        60
#define FULLTAB        10000

typedef char asciiz[PWD_LEN];

EXEC SQL TYPE asciiz IS STRING(PWD_LEN)
  REFERENCE;
asciiz     username;
asciiz     password;
asciiz    db_string;
EXEC SQL BEGIN DECLARE SECTION;
asciiz      tabname;
asciiz      newtabname;
char select_stmt[120];
char    create_stmt[120];
char    insert_stmt[120];
EXEC SQL END DECLARE SECTION;

int i, cursegcode,segcodes[FULLTAB];
float speeds[FULLTAB];
float median;

struct seg_info
{
  int         segcode;
  float       speed;
};

/* Declare function to handle unrecoverable
errors. */
void sql_error();

main(argc, argv)
int argc;
char *argv[];
{
```

```
  struct seg_info *seg_rec_ptr;

if (argc <= 1) {
  printf("Usage: %s tablename\n",argv[0]);
  exit(1);
}

/* EXEC SQL INCLUDE SQLCA; */

/* Allocate memory for cms_info struct. */
  if ((seg_rec_ptr =
    (struct seg_info *) malloc(sizeof(struct
    seg_info))) == 0)
  {
    fprintf(stderr, "Memory allocation
      error.\n");
    exit(1);
  }

/* Connect to ORACLE. */
  strcpy(username, "cms95");
  strcpy(password, "cms95");
  strcpy(db_string, "T:cms-td3:orcl");

  EXEC SQL WHENEVER SQLERROR DO
    sql_error("ORACLE error--");

  EXEC SQL CONNECT :username IDENTIFIED BY
    :password
  USING :db_string;
  printf("Connected...\n");

  strcpy(tabname,argv[1]);
  (newtabname,argv[1]);
  strcat(newtabname,"_MEDIAN");
  sprintf(select_stmt,"SELECT SEGCODE,SPEED
    FROM %s ORDER BY SEGCODE, SPEED",
tabname);
  printf ("Select statement: %s\n",
    select_stmt);

  EXEC SQL PREPARE sql_stmt FROM
:select_stmt;

  printf("After prepare statment\n");
  EXEC SQL DECLARE segdata CURSOR FOR
sql_stmt;

  printf("Declared cursor...\n");

/* Open the cursor. */
  EXEC SQL OPEN segdata;

  printf("Opened cursor...\n");
  printf("Creating output table...\n");
  sprintf(create_stmt,"CREATE TABLE %s
(SEGCODE
    NUMBER, SPEED NUMBER(8,4))", newtabname);
  EXEC SQL PREPARE sql_creat_stmt FROM
    :create_stmt;
  EXEC SQL EXECUTE sql_creat_stmt;

  printf("Segcode       Median \n");
  printf("-----------   --------\n");

  EXEC SQL WHENEVER NOT FOUND DO break;
  i=1;
  for (;;)
  {
    EXEC SQL FETCH segdata INTO :seg_rec_ptr;
    segcodes[i] = seg_rec_ptr->segcode;
      speeds[i] = seg_rec_ptr->speed;
      if (i > 1) {
        if (segcodes[i] != segcodes[i-1]) {
          cursegcode=segcodes[i-1];
          if (((i-1) % 2) == 0) {
            median=(speeds[(i-1)/2]+speeds
              [(i-1)/2+1])/2;
            printf("Even: %d   %f\n",
              cursegcode, median);
          }
          else {
            median=speeds[(i-1)/2+1];
            printf("Odd:  %d   %f\n",
              cursegcode, median);
          }
          sprintf(insert_stmt,"INSERT INTO %s
            VALUES (%d, %f)",newtabname,
              cursegcode,median);
          EXEC SQL PREPARE sql_insert_stmt
FROM
            :insert_stmt;
          EXEC SQL EXECUTE sql_insert_stmt;
          segcodes[1]=segcodes[i];
          speeds[1]=speeds[i];
          i=1;
        }
      i++;
      }
      else {
        i++;
      }
    }

    EXEC SQL WHENEVER NOT FOUND continue;

    /*  Do this for the last record.... */
      if (i > 1) {
        cursegcode=segcodes[i-1];
        if (((i-1) % 2) == 0) {
          median=(speeds[(i-1)/2]+speeds
            [(i-1)/2+1])/2;
          printf("Even: %d   %f\n",
            cursegcode,median);
        }
        else {
          median=speeds[(i-1)/2+1];
          printf("Odd:  %d   %f\n",
            cursegcode,median);
        }
    }
    else {
      cursegcode=segcodes[1];
      median=speeds[1];
    }
    sprintf(insert_stmt,"INSERT INTO %s VALUES
      (%d, %f)",newtabname,
    cursegcode,median);
    EXEC SQL PREPARE sqllast_insert_stmt FROM
      :insert_stmt;
    EXEC SQL EXECUTE sqllast_insert_stmt;

/* Close the cursor. */
  EXEC SQL CLOSE segdata;

  EXEC SQL COMMIT WORK RELEASE;
  exit(0);
}


void
sql_error(msg)
char *msg;
{
  char err_msg[512];
  int buf_len, msg_len;

  EXEC SQL WHENEVER SQLERROR CONTINUE;

  printf("\n%s\n", msg);

/* Call sqlglm() to get the complete text of
the error message. */
  buf_len = sizeof (err_msg);
  sqlglm(err_msg, &buf_len, &msg_len);
  printf("%.*s\n", msg_len, err_msg);

  EXEC SQL ROLLBACK RELEASE;
  exit(1);
}
```

APPENDIX E: DATA REPORTING METHODOLOGY

Color Coded Maps

Suppose that color coded maps containing minimum, average, and maximum speeds are needed. The first step would be to execute a query that contains these values using Oracle (see Computation of Minimum, Average, and Maximum Speed in appendix D). The second step would be to use GIS query tools to join segment attribute data to their location on the design map. The corresponding procedure is summarized below:

1. Include Oracle table in MGE project

a) In RIS Schema Manager, select the project name
b) Click on Data Definition and type in the schema password
c) Click on Include to select and add the table

2. Create ULF files for each speed category

a) In MGE, under the Tools menu, select Basic Nucleus Tools and then ULF Builder
b) Select design file segmth.dgn (master design file with thick lines)
c) Type in an .ulf file name such as av2030.ulf (do not select any file; the application tends to crash when this is done)
d) Click on Feature and select corridor segment
e) Click on Table and select corr_segments
f) Click on query
g) In the editing field, type in the following:
, acad9596_430_530_speeds
where corr-segments.segcode = acad9596_430_530_speeds.segcode
and avspeed > 20 and avspeed <= 30
h) Click on OK twice to go back to the ULF Builder window
i) Click on Apply, and then select Execute and Wait and click on OK. Another window showing the ULF creation process appears. At the end it shows how many segments were selected.
j) Repeat the procedure for other categories of speed values: 0-20, 30-40, 40-50, and 50-80.

3. Create .dgn files for each category of speed values

a) Under Tools, select the MGE Base Mapper Tools
b) Select Graphics processing, and then DGN Maker
c) Type in the name of the .ulf file (for example, av2030.ulf)
d) Type in the name of the output design file (for example, av2030.dgn)
e) Click on apply, and then Execute and wait
f) Repeat the procedure for other categories of speed values: 0-20, 30-40, 40-50, and 50-80

4. Merge all .dgn files to generate one color coded map

a) In MGE, attach each generated .dgn to a new design file
b) Make a copy of the contents of each attached file into a separate level in the active design file

c) Assign appropriate colors to the segments just copied. For example, make all segments with speeds between 0-20 mph red; all segments between 20 and 30 mph magenta, and so on.

11"x17" Archival Tabular Reports

Strip maps, overview maps, and thumbnail maps

These maps are treated as objects in Access. Because Microstation v.5 could not handle object linking and embedding (OLE), it was necessary to make a copy of the segment design file into a different platform that did handle OLE to generate all objects. Two software applications were used for this purpose: Autosketch and Coreldraw. Autosketch was used to generate the overview maps and the thumbnail maps. Coreldraw was used to generate the strip maps.

To generate overview and thumbnail maps, the original segment design file was first converted from Microstation format into .dxf format, and then from .dxf format into Autosketch format. The actual overview maps were generated by drawing a sketch of the corridors with the complete map in the background. No signalizing intersections, segment dividers, or street names were included in the overview maps. Having overview maps as simplified versions of the corridor network allowed for considerable savings in storage requirements. To complete the overview maps, boxes were drawn to indicate the specific area that the corresponding report page would cover.

To generate thumbnail maps, the original Autosketch map was cut into smaller areas. On each area, directions of travel were added, as were the names of the main crossing streets. This map also contained segment dividers and other details. Each thumbnail map illustrated the exact stretch of roadway that was analyzed on a particular report page and depicted by the overview map.

To generate strip maps, a linear template containing 20 segments in each direction was created in Coreldraw. This template was distorted so that each segment, regardless of its actual position and length, would be horizontal and have exactly the same graphical length as all other segments. Physical discontinuities were greatly simplified. For example, a vertical line was drawn to indicate a crossing street. Diagonal lines illustrated on-ramps, off-ramps, or interchanges. Dashed diagonal lines were drawn to connect vertical lines representing crossing streets that did not coincide with their counterpart in the opposite direction. All points of discontinuity were labeled, as was the main roadway. Direction of travel was also noted. Segment identification numbers for the initial and final segments in each direction were included to ensure that when data was added to the report page, it was certain that it was referring to the correct segment. Each strip overlapped at least one segment in both directions, at both ends, with the adjoining strips.

Procedure for generating 11"x17" archival tabular reports

1. Tables of interest

a) CORRIDOR: This table must be generated from table CORR_SEGMENTS. Use the ODBC driver to link the Microsoft Access database to the Oracle database. Create a link to table CORR_SEGMENTS and edit table CORRIDOR so that it contains all records in table CORR_SEGMENTS. For safety, create a back up copy of table CORRIDOR.

b) MAINSTT: This table must be generated from table SEG_TRAVEL_TIME. Create a link to table SEG_TRAVEL_TIME and edit table MAINSTT so that it contains all records in table SEG_TRAVEL_TIME. For safety, create a back up copy of table MAINSTT.

c) Reports: This table contains all segment code numbers order by position and report page. This table must be generated manually.

d) SM: This table contains all strip maps, overviews and thumbnail drawings. In general, it is best to generate all strip maps in Corel Draw, and all overviews and thumbnail drawings in Autosketch. Since the stretch option is used in Access to display the maps in the report pages, it may necessary to draw dots on the four corners of the original drawings to avoid distortions.

e) SP: This table contains all segment numbers associated with the third position in each report page. By default, all drawings are linked to the segment that is located on the third position. Run query QSP to generate table SP.

2. To change time periods

Occasionally it may be required to print reports for time periods other than the ones hard coded in the application. For example, assume that a noon peak period is of interest instead of a morning peak. To do this,

a) Make sure a table called DIR exists. If it does not exist, run query QRPAGE. Type any number for the parameter (for example 1). Table DIR should contain a list of attributes associated with report page No. 1.

b) Open query STA1 in design view and make all time period changes.

c) Save query STA1.

d) Open report REBAMP in design view.

e) Click on the field marked with a **AM PEAK** label. In properties, change "AM PEAK" with "NOON PEAK".

f) Save report REBAMP.

g) Open report RWBAMP in design view and repeat steps 5 and 6.

3. To print reports

a) Run macro MTEMPTABLE to delete table DIR and all other temporary files (except table STT; if this table needs to be deleted, it has to be done manually).

b) Run form Prepared By and type in the printing date. Click on OK.

c) Open form STARTDATE and type in the date range of interest. Make sure that the beginning date entered is one day before the actual first date of interest, and that the ending date entered is one day after the actual last date of interest. Click on OK.

d) Run query QMAIN to generate table STT. Table STT is a subset of table MAINSTT which contains only those records between the beginning and ending dates of interest. If table

STT already exists and a new date range is of interest, it is necessary to delete STT first, and then run QMAIN to generate a new version of STT.

e) Run form TESTFORM. Click on the pages of interest ONLY ONCE. Every time a cell is clicked on, a corresponding entry is generated in Table RP. This means that if a cell is clicked on 3 times, 3 entries will be generated in Table RP for that cell (and 3 copies of the same report page will be printed out). The number of print outs per page is equal to the number of times each cell has been selected.

f) Occasionally, it may be necessary to erase all records in Table RP to begin a new printing session.

In the process of running the application, several temporary tables are created. When the last page is processed and sent to the printer, these temporary tables are deleted. However, Access does not eliminate the storage space formerly occupied by these tables. As a result, the size of the database file increases every time new report pages are processed. For example, the Baton Rouge database usually takes about 22 Mbytes of storage. After printing the entire set of 54 pages, it is not unusual to find out that the size of the database file has increased to 400 Mbytes. To solve this problem, it is necessary to compact the database regularly. To do this, close the database and then select Utilities/Compact database. Access requires an input database and a target database. For safety, choose a target database file name which is different from the input database file name. It is advisable to use a sequential naming convention to keep track of the database evolution. For example, if the input database file is cmsbtr1.mdb, the target database file name should be cmsbtr2.mdb.

WWW PERL Script and Programs

CMS segment travel time data

The PERL script file is called query-data and is located on the /usr/etc/httpd/cgi-bin/ subdirectory of the Sun system at RSIP. The corresponding source code is given below:

```
#!/usr/local/bin/perl
#
# First show the information that is fed in.
#
require 'timelocal.pl';

&cgi_initialize;
&cgi_receive;
&cgi_header;
&cgi_split;
&cgi_check;
&cgi_send;
exit;

#  Subroutines

sub cgi_initialize {
  %dates=('JAN',1,'FEB',2,'MAR',3,'APR',4,
     'MAY',5,'JUN',6,'JUL',7,'AUG',8,'SEP',9,
     'OCT',10,'NOV',11,'DEC',12);
  for (0 .. 23) {
     $byhour[$_]=0;
     $byhour2[$_]=0;
     $bynumsamp[$_]=0;
  }
}

sub cgi_split_date {
  ($day,$month,$year) = split('-',$date);
}

sub cgi_calc_offset {
$perltime=timegm(0,0,0,$day,$dates{$month}-
  1,$year)+$time;
($s1,$m1,$h1,$m2,$m3,$y1,$w1,$y2,$isdst)=
  localtime($perltime);
if ($isdst) {
  $offset=5*3600;
}
else {
  $offset=6*3600;
}
}

sub cgi_calc_time {
  &cgi_calc_offset;
  $time -=$offset;
  if ($time < 0) {
    $time+=86400;
  }
  $hour = int $time/3600;
  $minute = int (($time-$hour*3600)/60);
  $second = int ($time-$hour*3600-
```

```
      $minute*60);
   $fraction = int (($time - int
($time))*100);
}

sub cgi_line_split {
     ($curlink,$date,$time,$speed) = split('
',$line);
}

sub cgi_split {
     @pairs = split(/&/,$incoming);



   foreach (@pairs) {
      ($name, $value) = split(/=/, $_);
      $name =~ tr/+/ /;
      $value=~ tr/+/ /;
      $name =~ s/%([A-F0-9][A-F0-9])/pack("C",
         hex($1))/gie;
      $value=~ s/%([A-F0-9][A-F0-9])/pack("C",
         hex($1))/gie;
      $FORM{$name} .= $value;
   }
}

sub cgi_header {
   print "Content-type: text/html\n";
   print "\n";
}

sub cgi_receive {
   $incoming=$ENV{'QUERY_STRING'};
}

sub cgi_check {
   &error_blank_field('segment number or
select
      a valid segment')
   unless ($FORM{'linkno'});
}

sub cgi_legend {
   print "Legend:<P>";
   print "<IMG SRC=/chris/query/blueball.gif>
      Average speed is 100% or more of posted
      speed<BR>\n";
   print "<IMG SRC=/chris/query/greenball.gif>
      Average speed is 90%-100% of posted
      speed<BR>\n";
   print "<IMG
SRC=/chris/query/yellowball.gif>
      Average speed is 80%-90% of posted
      speed<BR>\n";
   print "<IMG
SRC=/chris/query/orangeball.gif>
      Average speed is 50%-80% of posted
      speed<BR>\n";
   print "<IMG SRC=/chris/query/redball.gif>
      Average speed is 30%-50% of posted
      speed<BR>\n";
   print "<IMG
SRC=/chris/query/purpleball.gif>
      Average speed is less than 30% of posted
      speed<BR>\n";
}

sub cgi_footer {
   print "&#169; Copyright 1995,";
   print "<A HREF=\"http://www.rsip.lsu.edu/
chris/chris.home.html\">\n";
   print "Chris Schwehm</A>,";
   print "<A HREF=\"http://www.rsip.lsu.edu\">
      RSIP Lab</A>,";
   print "<A
HREF=\"http://www.lsu.edu\">LSU</A>
      <P>\n";
}
```

```
sub error_blank_field {
   local($problem) = @_;
   print "<HTML><HEAD><TITLE>Information
Request
      Error</TITLE></HEAD><BODY>\n";
   print "<H1>CMS Database Request Error
      </H1>\n";
   print "You did not fill in $problem.\n";
   print "Please go back to the form and do
      so.\n";
   print "Use the Back button on your browser
to
      do this; otherwise,\n";
   print "the data that you entered will be
      lost.<P>\n";
   print "<A HREF=/chris/query/cms-query.html>
      Back to query map</A><P>\n";
   &cgi_footer;
   print "</BODY></HTML>\n";
   exit;
}

sub cgi_send {

   $lkno=$FORM{'linkno'};
   print "<HTML><HEAD><TITLE>CMS Database
Query
      Results</TITLE></HEAD><BODY>\n";
   while (</ftp/www/cms/*.map>) {
      open (FINDMAP,$_);
      $fname=$_;
      while(<FINDMAP>) {
         if (index($_,"linkno=$lkno") != -1) {
            $mapfile=substr($fname,rindex
               ($fname,"/")+1);
            $mapfile=substr($mapfile,0,index
               ($mapfile,".map"));
            }
         }
      close(FINDMAP);
   }
   if (length($mapfile) > 0) {
      printf "<IMG
SRC=\"/chris/query/%s.gif\">",
         $mapfile;
   }

   printf "<H2>LSU CMS Database Query for
      Segment # %5s</H2>\n",$lkno;

   open(DATAFILE,"/usr/local/oracle/orahome/
      progs/get_speedlimit '$lkno' |");

   $nospeeds=0;

   while(<DATAFILE>) {
      $speedlimit=$_;
   $nospeeds++;
   if ($nospeeds == 1) {
      if ($speedlimit > 1) {
         printf("<H3>The posted speed limit on
            this segment is %3d</H3>\n",
            $speedlimit);
      }
      else {
         printf("<H3>The posted speed limit is
            not known!</H3>\n");
      }
   }
   }
   close(DATAFILE);

   open(DATAFILE,"/usr/local/oracle/orahome/
     progs/get_cms '$lkno' |");

  print "<PRE>    Seg. No.    Date
Time
        Speed\n";
  print "    --------    --------        --------
   -----\n</PRE>";
```

```
$totspeed=0;
$totlinks=0;
print "<PRE>";
while(<DATAFILE>) {
  $line=$_;
  $subst=substr($line,0,length($lkno));
  if ($subst eq $lkno) {
    &cgi_line_split;
    &cgi_split_date;
    &cgi_calc_time;
    $totspeed += 1.0/$speed;
    $byhour[$hour] += 1.0/$speed;
    $byhour2[$hour] += $speed;
    $bynumsamp[$hour]++;
    $totlinks++;
    printf("   %5s      %8s
      %2d:%02d:%02d.%02d   ",
    $curlink,$date,$hour,$minute,$second,
      $fraction);
    printf("%4.1f\n",$speed);
  }
}
close(DATAFILE);
print "</PRE>";
if ($totlinks == 0) {
  print "No segments found!<P>";
}
else {
  $avg=($totlinks)/($totspeed);
  printf("<PRE>-----------------------------
--
    -----------------\n");
  printf("Avg. Spd.
                %5.1f\n</PRE>",$avg);
  printf("<PRE>Total Segment Samples
  %5d\n\n</PRE>", $totlinks);
  print "Hourly average speeds:<P>";
  print "<TABLE>";
  print
"<TR><TD></TD><TD>Hour</TD><TD>Speed
    (mph)</TD><TD># Samples</TD>";
  print "<TD>Std. Dev.</TD></TR>\n";
  print "<TR><TD></TD><TD>------</TD><TD> -
--
    -------</TD>";
  print "<TD>---------</TD><TD>---------
</TD>
    </TR>\n";
  for (0 .. 23) {
    $hourno=$_;
    if ($bynumsamp[$hourno] != 0) {
      $v1=$bynumsamp[$hourno]/$byhour
        [$hourno];
      $v2=$byhour2[$hourno]/$bynumsamp
        [$hourno];
      if ($v2 >= $v1) {
        $stddev=sqrt($v1*($v2-$v1));
      }
      elsif ($v1-$v2 < 1e-8) {
        $stddev=0;
      }
      else {
        $stdev=-1;
      }
      if ($speedlimit !=0) {
        $speed_ratio = $v1/$speedlimit;
      }
      else {
        $speed_ratio=0;
      }

      if ($speed_ratio >= 1) {
        printf("<TR><TD><IMG
SRC=/chris/query/
          blueball.gif></TD>");
      }
      elsif ($speed_ratio >= 0.9) {
        printf("<TR><TD><IMG
SRC=/chris/query/
          greenball.gif></TD>");
      }
      elsif ($speed_ratio >= 0.8) {
        printf("<TR><TD><IMG
SRC=/chris/query/
          yellowball.gif></TD>");
      }
      elsif ($speed_ratio >= 0.5) {
        printf("<TR><TD><IMG
SRC=/chris/query/
          orangeball.gif></TD>");
      }
      elsif ($speed_ratio >= 0.3) {
        printf("<TR><TD><IMG
SRC=/chris/query/
          redball.gif></TD>");
      }
      else {
        printf("<TR><TD><IMG
SRC=/chris/query/
          purpleball.gif></TD>");
      }

      printf("<TD>%2d-%2d</TD><TD>%5.1f</TD>
        <TD>%3d</TD><TD>%7.4f</TD></TR>\n",
        $hourno,$hourno+1,$v1,$bynumsamp
        [$hourno],$stddev);
    }
  }
  print "</TABLE><P><P>";


}
&cgi_legend;
print "<P><A HREF=/chris/query/cms-
  query.html>Back to query map</A><P>\n";
&cgi_footer;
exit;
}
```

The PERL script file calls a C program to query the SQL database. This program is called get_cms.pc and is located on the /usr/local/oracle/orahome/progs/ subdirectory of the Sun system at RSIP. The corresponding source code is given below:

```
#include <stdio.h>
#include <sqlca.h>

#define UNAME_LEN      20
#define PWD_LEN        60

typedef char asciiz[PWD_LEN];

EXEC SQL TYPE asciiz IS STRING(PWD_LEN)
REFERENCE;
asciiz     username;
asciiz     password;
asciiz     db_string;
int        search_seg_no;

struct cms_info
{
  int        segcode;
  asciiz     gpsdate;
  float      starttime;
  float      gpsspeed;
};
```

```
/* Declare function to handle unrecoverable
     errors. */
void sql_error();

main(argc, argv)
int argc;
char *argv[];
{
  struct cms_info *cms_rec_ptr;

/* Allocate memory for cms_info struct. */
  if ((cms_rec_ptr = (struct cms_info *)
    malloc(sizeof(struct cms_info))) == 0)
  {
    fprintf(stderr, "Memory allocation
      error.\n");
    exit(1);
  }

/* Connect to ORACLE. */
  strcpy(username, "cms95");
  strcpy(password, "cms95");
  strcpy(db_string, "T:cms-td3:orcl");

/*  EXEC SQL DECLARE db_name DATABASE; */
  EXEC SQL WHENEVER SQLERROR DO sql_error
    ("ORACLE error--");


  EXEC SQL CONNECT :username IDENTIFIED BY
    :password USING :db_string;

    search_seg_no = atoi(argv[1]);

  EXEC SQL DECLARE cmsdata CURSOR FOR
    SELECT SEGCODE, GPSDATE, STARTTIME,
    GPSSPEED
    FROM SEG_TRAVEL_TIME
    WHERE SEGCODE = :search_seg_no;

/* Open the cursor. */
  EXEC SQL OPEN cmsdata;

  EXEC SQL WHENEVER NOT FOUND DO break;

  for (;;)
  {
    EXEC SQL FETCH cmsdata INTO :cms_rec_ptr;
    printf("%5d  %9s  %9.2f  %9.2f\n",
      cms_rec_ptr->segcode, cms_rec_ptr->
      gpsdate, cms_rec_ptr->starttime,
      cms_rec_ptr->gpsspeed);
  }

/* Close the cursor. */
  EXEC SQL CLOSE cmsdata;

  EXEC SQL COMMIT WORK RELEASE;
  exit(0);
}

void
sql_error(msg)
char *msg;
{
  char err_msg[512];
  int buf_len, msg_len;

  EXEC SQL WHENEVER SQLERROR CONTINUE;

  printf("\n%s\n", msg);

/* Call sqlglm() to get the complete text of
     the error message. */
  buf_len = sizeof (err_msg);
  sqlglm(err_msg, &buf_len, &msg_len);
  printf("%.*s\n", msg_len, err_msg);

  EXEC SQL ROLLBACK RELEASE;
  exit(1);
}
```

The PERL script file also calls program get_speedlimit.pc. This program is also located on the /usr/local/oracle/orahome/progs/ subdirectory of the Sun system at RSIP. The corresponding source code is given below:

```
#include <stdio.h>
#include <sqlca.h>

#define UNAME_LEN      20
#define PWD_LEN        60

typedef char asciiz[PWD_LEN];

EXEC SQL TYPE asciiz IS STRING(PWD_LEN)
  REFERENCE;
asciiz     username;
asciiz     password;
asciiz    db_string;
int      search_seg_no;

struct cms_speedlimit
{
  int  speedlimit;
};

/* Declare function to handle unrecoverable
     errors. */
void sql_error();

main(argc, argv)
int argc;
char *argv[];
{
  struct cms_speedlimit *cms_rec_ptr;

/* Allocate memory for cms_info struct. */
  if ((cms_rec_ptr = (struct cms_speedlimit
*)
    malloc(sizeof(struct
cms_speedlimit)))==0)
    {
      fprintf(stderr, "Memory allocation
        error.\n");
      exit(1);
    }

/* Connect to ORACLE. */
  strcpy(username, "cms95");
  strcpy(password, "cms95");
  strcpy(db_string, "T:cms-td3:orcl");

  EXEC SQL WHENEVER SQLERROR DO sql_error
    ("ORACLE error--");


  EXEC SQL CONNECT :username IDENTIFIED BY
    :password USING :db_string;

  search_seg_no = atoi(argv[1]);

  EXEC SQL DECLARE cmsdata CURSOR FOR
  SELECT sPEEDLIMIT FROM  SPEED_LIMITS
```

```
   WHERE SEGCODE = :search_seg_no;

/* Open the cursor. */
   EXEC SQL OPEN cmsdata;

EXEC SQL WHENEVER NOT FOUND DO break;

   for (;;)
   {
     EXEC SQL FETCH cmsdata INTO :cms_rec_ptr;
     printf("%6d\n", cms_rec_ptr->speedlimit);
   }

/* Close the cursor. */
   EXEC SQL CLOSE cmsdata;

   EXEC SQL COMMIT WORK RELEASE;
   exit(0);
}
```

```
void
sql_error(msg)
char *msg;
{
   char err_msg[512];
   int buf_len, msg_len;

   EXEC SQL WHENEVER SQLERROR CONTINUE;

   printf("\n%s\n", msg);

/* Call sqlglm() to get the complete text of
the error message */
   buf_len = sizeof (err_msg);
   sqlglm(err_msg, &buf_len, &msg_len);
   printf("%.*s\n", msg_len, err_msg);

   EXEC SQL ROLLBACK RELEASE;
   exit(1);
}
```

Real-time travel time data

The program used to update the WWW home page that displays real-time travel time data at checkpoints along the I-10, I-12 corridor in Baton Rouge is called trvclch.c and is located at /home4/staff/chris/cprogs/. The corresponding source code is given below:

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#define CKPTFILE "ckpoint.dat"
#define MINDIST 250
#define MPERDEGLAT 110860.5
#define MPERDEGLON 95900.4
#define NOCOORD 5000
#define NOCKPT  30
#define NOPASSSEC 600
struct ckptinfo {
  int nockpt;
  char ckptid[NOCKPT][5];
  double ckptx[NOCKPT];
  double ckpty[NOCKPT];
  char ckptdir[NOCKPT];
  int year[NOCKPT];

  int month[NOCKPT];
  int day[NOCKPT];
  double time[NOCKPT];
  int lastyear[NOCKPT];
  int lastmonth[NOCKPT];
  int lastday[NOCKPT];
  double lasttime[NOCKPT];
};
struct coordinfo {
  int nocoord;
  double utmtime[NOCOORD];
  int year[NOCOORD];
  int month[NOCOORD];
  int day[NOCOORD];
  double coordx[NOCOORD];
  double coordy[NOCOORD];
  double speed[NOCOORD];
  int type[NOCOORD];
};

void ssort(int n,double a[],double b[],double
c[],double d[],int e[],
int f[],int g[],int h[]);

double calcdist(double lat1, double lon1,
  double lat2, double lon2);
void readcoord(char *fname,int curdaysec,
  struct coordinfo *coords);
void readckpt(char *fname, struct ckptinfo
  *ckpts);
void calckpts(struct ckptinfo *ckpts, struct
  coordinfo *coords);
int matchpoint(char *ckptstring,struct
  ckptinfo *ckpts);

extern time_t timezone,altzone;
extern char *tzname[2];

main (int argc, char *argv[])
{
struct ckptinfo ckpts;
struct coordinfo coords;
int nockpt,i;
time_t cursec;
long int curdaysec;
struct tm *curtime;

/*  Now get current gmt time  */

  cursec=time(NULL);
  curtime=gmtime(&cursec);
  curdaysec=(curtime->tm_hour)*3600+(curtime-
    >tm_min)*60+curtime->tm_sec;

  printf("Current time is %d:%d:%d %ld\n",
    curtime->tm_hour,curtime->tm_min,
    curtime->tm_sec,curdaysec);

  coords.nocoord=0;
  readcoord(argv[1],curdaysec,&coords);
  readckpt(CKPTFILE, &ckpts);
  calckpts(&ckpts, &coords);
}

/*********************************************
 Subroutine to calculate checkpoint info
*********************************************/
void calckpts(struct ckptinfo *ckpts, struct
  coordinfo *coords)
{
int i,j,foundone,firstcoord,lastcoord;
char ckfname[30];
FILE *ckptfile;
double mindist,mindisttime;
double measdist;
  for (i=0;i<ckpts->nockpt;i++) {
```

```
      foundone=0;
    mindist=MINDIST;
    mindisttime=0;
    for (j=0;j<coords->nocoord;j++) {
      measdist=calcdist(ckpts->ckpty[i],ckpts-
      >ckptx[i], coords->coordy[j],coords-
      >coordx[j]);
      if ((coords->utmtime[j]-mindisttime) >
        NOPASSSEC)
        mindist=MINDIST;
      if (measdist < MINDIST) {
        printf("CK pt is %s time is %lf\n",
          &(ckpts->ckptid[i][0]),
          coords->utmtime[j]);
        printf("I: %d J: %d  Start: Meas dist
          %lf, min dist %lf\n",i,j,measdist,
          mindist);
      }
      if (measdist < mindist) {
        firstcoord=j-3;
      if (firstcoord < 0)
        firstcoord=0;
      lastcoord=j+3;
      if (lastcoord > NOCOORD-1)
        lastcoord=NOCOORD-1;
      if (((ckpts->ckptdir[i] == 'N') &&
(coords-
        >coordy[firstcoord]+.0004 <
        coords->coordy[lastcoord])) ||
((ckpts-
        >ckptdir[i] == 'S')
        && (coords->coordy[firstcoord] >
        coords->coordy[lastcoord]+.0004))
        || ((ckpts->ckptdir[i] == 'W') &&
        (coords->coordx[firstcoord]+.0004 >
        coords->coordx[lastcoord]))
        || ((ckpts->ckptdir[i] == 'E') &&
        (coords->coordx[firstcoord] < coords-
        >coordx[lastcoord]+.0004))) {

        if ((coords->utmtime[j]-ckpts-
>time[i])
          > NOPASSSEC) {
          ckpts->lasttime[i]=ckpts->time[i];
          ckpts->lastyear[i]=ckpts->year[i];
          ckpts->lastmonth[i]=ckpts->month[i];
          ckpts->lastday[i]=ckpts->day[i];
        sprintf(ckfname,"cph%02d%02d.%02d",
          ckpts->lastmonth[i],ckpts-
>lastday[i],
          ckpts->lastyear[i]);
        if (ckpts->lasttime[i] != 0) {
          if ((ckptfile=fopen(ckfname,"a")) ==
            NULL) {
            printf("File %s cannot be
opened!",
              ckfname);
            exit(1);
            }
          fprintf(ckptfile,"%5s %2d/%2d/%2d
            %lf\n",&(ckpts->ckptid[i][0]),
            ckpts->lastmonth[i],ckpts-
            >lastday[i],ckpts->lastyear[i],
            ckpts->lasttime[i]);
          fclose(ckptfile);
          }
        }
        ckpts->time[i]=coords->utmtime[j];
        ckpts->year[i]=coords->year[j];
        ckpts->month[i]=coords->month[j];
        ckpts->day[i]=coords->day[j];
        mindist=measdist;
        mindisttime=ckpts->time[i];
      }
    }
    }
    }
    for (i=0;i<ckpts->nockpt;i++) {
    sprintf(ckfname,"cph%02d%02d.%02d",ckpts-
      >month[i],ckpts->day[i],ckpts->year[i]);
      if (ckpts->time[i] != 0) {
        if ((ckptfile=fopen(ckfname,"a")) ==
NULL){
          printf("File %s cannot be opened!",
            ckfname);
          exit(1);
        }
        fprintf(ckptfile,"%5s %2d/%2d/%2d %lf\n",
          &(ckpts->ckptid[i][0]),
          ckpts->month[i],ckpts->day[i],ckpts-
          >year[i],ckpts->time[i]);
        fclose(ckptfile);
      }
      }
}

/******************************************/
/*  Subroutine to find matching checkpoint  /
/******************************************/
int matchpoint(char *ckptstring,struct
  ckptinfo *ckpts)
{
  int i=0;
  while (strcmp(ckptstring,&(ckpts-
    >ckptid[i][0])) !=0) i++;
  return(i);
}


/******************************************/
/*  Subroutine to read in coordinate files */
/******************************************/
void readcoord(char *fname,int curdaysec,
  struct coordinfo *coords)
{
FILE *coordfile;
struct stat fileinfo;
time_t accesstime;
struct tm *filetime;
double lat, lon, btime, curspeed;
int curtype;
  if ((coordfile=fopen(fname,"rb")) == NULL)
{
  printf("File %s cannot be opened!",fname);
  exit(1);
  }

  stat(fname,&fileinfo);
  accesstime=fileinfo.st_mtime;
  filetime=gmtime(&accesstime);

  while (!feof(coordfile)) {
    if ((fscanf(coordfile,"%lf %lf %lf %lf %d
      %*d",&btime,&lat,&lon,
      &curspeed,&curtype)) != 5) {
      if (!feof(coordfile)) {
        printf("Couldn't read coordinate
file
          %s !\n",fname);
        fclose(coordfile);
        exit(1);
      }
    }
    else {
      coords->year[coords-
>nocoord]=filetime-
        >tm_year;
      coords->month[coords-
>nocoord]=filetime-
        >tm_mon+1;
      coords->day[coords->nocoord]=filetime-
        >tm_mday;
      coords->utmtime[coords-
>nocoord]=btime;
      coords->coordx[coords->nocoord]=lon;
      coords->coordy[coords->nocoord]=lat;
      coords->speed[coords-
>nocoord]=curspeed;
      coords->type[(coords->nocoord)++]=
        curtype;
```

```
      }
    }
}
/*****************************************/
/*  Subroutine to read in checkpoint files */
/*****************************************/
void readckpt(char *fname, struct ckptinfo
   *ckpts)
{
FILE *ckptfile;
int i;
   if ((ckptfile=fopen(fname,"rb")) == NULL) {
   printf("File %s cannot be opened!",fname);
   exit(1);
   }

   if ((fscanf(ckptfile,"%d",&(ckpts-
>nockpt)))
      != 1) {
      printf("Couldn't read checkpoint file %s
         !\n",fname);
      fclose(ckptfile);
      exit(1);
   }
   for (i=0;i<ckpts->nockpt;i++) {
      if ((fscanf(ckptfile,"%s %lf %lf %c %d %d


         %d %lf %d %d %d %lf",
         &(ckpts->ckptid[i][0]),&(ckpts-
         >ckptx[i]),&(ckpts->ckpty[i]),
         &(ckpts->ckptdir[i]),&(ckpts-
>year[i]),
         &(ckpts->month[i]),&(ckpts->day[i]),
         &(ckpts->time[i]),&(ckpts-
>lastyear[i]),
         &(ckpts->lastmonth[i]),&(ckpts-
         >lastday[i]),&(ckpts->lasttime[i])))
         != 12) {
         printf("Couldn't read checkpoints\n");
         fclose(ckptfile);
      exit(1);
      }
   }
}

/*****************************************/
/*   Subroutine to calculate the distance
                 between 2 points          */
/*****************************************/
double calcdist(double lat1, double lon1,
   double lat2, double lon2)
{
double diffx,diffy,result;
   diffx=MPERDEGLON*(lon1-lon2);
   diffy=MPERDEGLAT*(lat1-lat2);
   result=sqrt(diffx*diffx+diffy*diffy);
   return(result);
}

/*****************************************/
/*     Subroutine to sort the coordinates
                according to time          */
/*****************************************/
void ssort(int n,double a[],double b[],double
   c[],double d[],int e[],int f[],int g[],
   int h[])
{
   int i,j,y,r,s,t;
   double v,u,w,x;

   for (j=2;j<=n;j++) {
     v=a[j-1];
     u=b[j-1];
     w=c[j-1];
     x=d[j-1];
     y=e[j-1];
     r=f[j-1];
     s=g[j-1];
     t=h[j-1];
     i=j-1;
     while (i > 0 && a[i-1] < v) {
       a[i]=a[i-1];
       b[i]=b[i-1];
       c[i]=c[i-1];
       d[i]=d[i-1];
       e[i]=e[i-1];
       f[i]=f[i-1];
       g[i]=g[i-1];
       h[i]=h[i-1];
       i--;
     }
     a[i]=v;
     b[i]=u;
     c[i]=w;
     d[i]=x;
     e[i]=y;
     f[i]=r;
     g[i]=s;
     h[i]=t;
   }
}
```

The program used to update the WWW home page that displays bus tracking real-time travel time data is called busmap.c. The corresponding source code is given below:

```
#include <stdio.h>
#include <sys/stdtypes.h>
#include <math.h>
#include <time.h>
#define INFILE "busroute.ppm"
#define OUTFILE "newmap.ppm"
#define CKPTFILE "ckpoint.dat"
#define MINDIST 250
#define MPERDEGLAT 110860.5
#define MPERDEGLON 95900.4
#define GIFCMD "/usr/bin/X11/ppmtogif
  newmap.ppm > bustmp.gif"
#define MVCMD "mv bustmp.gif busmap.gif"
#define MVINCLUDEF "mv timeinfo.tmp
  timeinfo.html"
#define INCLUDETMP "timeinfo.tmp"
#define MINX -91.19517
#define MAXX -91.16549
#define MINY 30.42373
#define MAXY 30.38526
#define BOXSIZE 4

#define SAVEMIN 180

struct imginfo {
     int cols;
     int rows;
     int maxval;
     unsigned char *imager;
     unsigned char *imageg;
     unsigned char *imageb;
};

struct ckptinfo {
     int nockpt;
     double ckptxs[10];
     double ckptys[10];
     double ckptxe[10];
     double ckptye[10];
     double starttime[10];
     double endtime[10];
```

```
    double newstarttime[10];
};

struct coordinfo {
    int nocoord;
    double utmtime[10];
    double coordx[10];
    double coordy[10];
    double speed[10];
    int type[10];
};

char getfilech(FILE *file);

unsigned char getrawbyte(FILE *file);

int getint(FILE *file);

void ssort(int n,double a[],double b[],double
  c[],double d[],int e[]);
double calcdist(double lat1, double lon1,
  double lat2, double lon2);
void readcoord(char *fname,int curdaysec,
  struct coordinfo *coords);
void readckpt(char *fname, struct ckptinfo
  *ckpts);
void calckpts(struct ckptinfo *ckpts, struct
  coordinfo *coords);
void writeckpt(char *fname, struct ckptinfo
  *ckpts);
void readimage(char *fname,struct imginfo
  *img);
void htmlinclude(struct imginfo *img, time_t

  cursec, struct coordinfo *coords,
  struct ckptinfo *ckpts);
void plotcoord(struct coordinfo *coords,
struct
  imginfo *img);
void writefile(struct imginfo *img);
main (int argc, char *argv[])
{
struct imginfo img;
struct ckptinfo ckpts;
struct coordinfo coords;
int nockpt,i;
double
ckptx[10],ckpty[10],starttime[10],lasttravtim
e[10];
time_t cursec;
long int curdaysec;
struct tm *curtime;

  cursec=time(NULL);
  curtime=gmtime(&cursec);
  curdaysec=(curtime->tm_hour)*3600+(curtime-
    >tm_min)*60+curtime->tm_sec;

  printf("Current time is %d:%d:%d %ld\n",
    curtime->tm_hour,curtime->tm_min,
    curtime->tm_sec,curdaysec);

  coords.nocoord=0;
  for (i=1;i<argc;i++) {
    readcoord(argv[i],curdaysec,&coords);
  }

  ssort(coords.nocoord,coords.utmtime,
    coords.coordx,coords.coordy,
    coords.speed,coords.type);

  readckpt(CKPTFILE, &ckpts);
  calckpts(&ckpts, &coords);
  writeckpt(CKPTFILE, &ckpts);
  readimage(INFILE,&img);

/*  Convert the coords to image coords */

  htmlinclude(&img,cursec,&coords,&ckpts);

  plotcoord(&coords,&img);

/*  Write out the new image  */

  writefile(&img);
  free(img.imager);
  free(img.imageg);
  free(img.imageb);

/* Convert it and put it in place  */

  system(GIFCMD);
  system(MVCMD);
  system(MVINCLUDEF);
}

/******************************************/
/* Subroutine to calculate checkpoint info  /
/******************************************/
void calckpts(struct ckptinfo *ckpts, struct
  coordinfo *coords)
{
int i,j,foundone;
double mindist;
double measdist;
  for (i=0;i<ckpts->nockpt;i++) {
    mindist=MINDIST;
    for (j=0;j<coords->nocoord;j++) {
      measdist=calcdist(ckpts->ckptys[i],
        ckpts->ckptxs[i],coords->coordy[j],
        coords->coordx[j]);
      printf("Start: Meas dist %lf, min dist
        %lf\n",measdist,mindist);
      if (measdist < mindist) {
        ckpts->newstarttime[i]=coords-
          >utmtime[j];
        mindist=measdist;
      }
    }

    if (ckpts->newstarttime[i] != 0) {
      foundone=0;
      for (j=0;j<coords->nocoord;j++) {
        measdist=calcdist(ckpts->ckptye[i],
        ckpts->ckptxe[i],coords->coordy[j],
        coords->coordx[j]);
        printf("End: Meas dist %lf, min dist
          %lf\n",measdist,mindist);
      if ((measdist < mindist) && (coords-
        >utmtime[j] >ckpts-
>newstarttime[i])){
        if (!foundone) {
          ckpts->starttime[i]=ckpts-
            >newstarttime[i];
          foundone=1;
        }
      ckpts->endtime[i]=coords->utmtime[j];
      ckpts->newstarttime[i]=0;
      mindist=measdist;
      }
    }
  }
  }
}

/*******************************************/
/*  Subroutine to write output image       */
/*******************************************
*/
void writefile(struct imginfo *img)
{
FILE *outfile;
int x;
  if ((outfile=fopen(OUTFILE,"wb")) == NULL)
{
  printf("File %s cannot be
opened!",OUTFILE);
  exit(1);
  }
```

```
  printf("Rows: %d Cols: %d Maxval:
%d\n",img-
      >rows,img->cols,img->maxval);
  fprintf(outfile,"P6\n%d %d\n%d\n",img-
>cols,
      img->rows,img->maxval);
  for (x=0;x<img->rows*img->cols;x++) {
  fprintf(outfile,"%c%c%c",img-
>imager[x],img-
      >imageg[x],img->imageb[x]);
  }
  fclose(outfile);
}
/*******************************************/
/* Subroutine to place coordinates on image*/
/*******************************************/
void plotcoord(struct coordinfo *coords,
struct
  imginfo *img)
{
double slopex,slopey;
int i,newx,newy,boxl,boxr,boxt,boxb,size,x,y;

  slopex=(img->cols-1)/(MAXX-MINX);
  slopey=(img->rows-1)/(MAXY-MINY);

  for (i=(coords->nocoord)-1;i>=0;i--) {
    newx=slopex*(coords->coordx[i]-MINX);
    newy=slopey*(coords->coordy[i]-MINY);
    printf("Old coord: x: %lf y: %lf  New
      coords: x:  %d  y:  %d\n",
      coords->coordx[i],coords-
      >coordy[i],newx,newy);

    size=BOXSIZE;


/*  Make a box  */
    if (i == 0) {
      if ((boxl = newx-size*2) < 0)
        boxl=0;
      if ((boxr=newx+size*2) > (img->cols-
1))
        boxl=img->cols-1;
      if ((boxt=newy-size*2) < 0)
        boxt=0;
      if ((boxb=newy+size*2) > (img->rows-
1))
        boxb=img->rows-1;
      }
    else {
      if ((boxl = newx-size) < 0)
        boxl=0;
      if ((boxr = newx+size) > (img->cols-
1))
        boxl=img->cols-1;
      if ((boxt = newy-size) < 0)
        boxt=0;
      if ((boxb = newy+size) > (img->rows-
1))
        boxb=img->rows-1;
      }

    printf("Bounding box: L: %d  R: %d  T: %d
  B: %d\n",boxl,boxr,boxt,boxb);
  if (i == 0) {
    for (y=boxt;y <= boxb;y++)
      for (x=-1;x<=1;x++) {
        img->imager[y*img->cols+newx+x]=img-
          >maxval;
        img->imageg[y*img->cols+newx+x]=0;
        img->imageb[y*img->cols+newx+x]=0;
      }
      for (x=boxl;x<=boxr;x++)
        for (y=-1;y<=1;y++) {
          img->imager[(newy+y)*img->cols+x]=
            img->maxval;
          img->imageg[(newy+y)*img-
>cols+x]=0;
          img->imageb[(newy+y)*img-
>cols+x]=0;
        }
    }
  else {
    for (y=boxt;y <= boxb;y++)
      for (x=boxl;x<=boxr;x++) {
        img->imager[y*img->cols+x]=img-
>maxval
          -i*img->maxval/(2*coords-
>nocoord);
        img->imageg[y*img->cols+x]=0;
        img->imageb[y*img->cols+x]=0;
      }
    }
  }
}

/*******************************************/
/*Subroutine to print out html include file*/
/*******************************************/
void htmlinclude(struct imginfo *img, time_t
  cursec, struct coordinfo *coords,
  struct ckptinfo *ckpts)
{
int i;

FILE *htmlfile;

struct tm *curtime;

long locdaysec,locstart,locend,locnew;
char *typestr[]={"2-D GPS","3-D GPS","2-D
  DGPS","3-D DGPS"};

/* Open the html file for blocks and times
*/
  if ((htmlfile=fopen(INCLUDETMP,"w"))==NULL)
{
    printf("File %s cannot be opened!",
      INCLUDETMP);
    exit(1);
  }
  fprintf(htmlfile,"<TABLE
BORDER><TR><TD><TD>
    Time<TD>Speed<TD>Data type<TD></TR>\n");
  for (i=coords->nocoord-1;i>=0;i--) {
    curtime=localtime(&cursec);
    locdaysec=coords->utmtime[i]+curtime-
      >tm_gmtoff;
    fprintf(htmlfile,"<TR><TD
ALIGN=CENTER><IMG
    SRC=box%d.gif><TD>%2d:%02d:%02d %s<TD>
      %5.1lf MPH<TD>%s</TR>\n",
      i+1, locdaysec/3600,(locdaysec %
      3600)/60,locdaysec % 60,
      curtime->tm_zone,coords-
      >speed[i],typestr[coords->type[i]]);
    }
  fprintf(htmlfile,"</TABLE><P>\n");

/* Print travel data in another table  */

  fprintf(htmlfile,"<TABLE BORDER><TR><TD>
    Segment<TD>Start Time<TD>Travel
    Time</TR>\n");
  for (i=0;i<ckpts->nockpt;i++) {
    if (ckpts->starttime[i]==0) {
      fprintf(htmlfile,"<TR><TD>%d<TD>N/A
        <TD>N/A</TR>\n",i+1);
    }
    else {
      curtime=localtime(&cursec);
      locstart=ckpts->starttime[i]+curtime-
        >tm_gmtoff;
      locend=ckpts->endtime[i]+curtime-
        >tm_gmtoff;
      locnew=ckpts->newstarttime[i]+curtime-
        >tm_gmtoff;
      fprintf(htmlfile,"<TR><TD>%d<TD>%2d:
```

```
          %02d:%02d %s<TD>%2d:%02d</TR>\n",
          i+1, locstart/3600,
          (locstart %3600)/60,locstart%60,
          curtime->tm_zone,(locend-
locstart)/60,
          (locend-locstart) % 60);
     }
   }
   fprintf(htmlfile,"</TABLE><P>\n");
   fclose(htmlfile);
}

/*******************************************/
/* Subroutine to read in the coordinate    */
/*                files                    */
/*******************************************/
void readcoord(char *fname,int curdaysec,
   struct coordinfo *coords)
{
FILE *coordfile;
double lat, lon, btime, curspeed;
int curtype;
   if ((coordfile=fopen(fname,"rb")) == NULL)
{
     printf("File %s cannot be
opened!",fname);
     exit(1);
   }

/* Read coordinates */
   if ((fscanf(coordfile,"%lf %lf %lf %lf %d",
     &btime,&lat,&lon,
     &curspeed,&curtype)) != 5) {
     printf("Couldn't read coordinate file %s
        !\n",fname);
     fclose(coordfile);
   }
   else {
     fclose(coordfile);
     printf("UTM:  %lf Lat:  %lf  Lon:
%lf\n",
        btime,lat,lon);
     if (curdaysec-btime <= SAVEMIN*60) {
       if (( lon < MINX) || (lon > MAXX) ||
          (lat < MAXY) || (lat > MINY)) {
         printf("Coordinates out of map
            range\n");
       }
       else {
         coords->utmtime[coords-
           >nocoord]=btime;
         coords->coordx[coords->nocoord]=lon;
         coords->coordy[coords->nocoord]=lat;
         coords->speed[coords-
           >nocoord]=curspeed;
         coords->type[(coords->nocoord)++]=
         curtype;
         }
       }
     else {
       printf("Coordinates too old...\n");
     }
   }
}


/*******************************************/
/*Subroutine to read in the checkpoint     */
/*                files                    */
/*******************************************/
void readckpt(char *fname, struct ckptinfo
   *ckpts)
{
FILE *ckptfile;
int i;
   if ((ckptfile=fopen(fname,"rb")) == NULL) {
     printf("File %s cannot be
opened!",fname);
     exit(1);
   }

   if ((fscanf(ckptfile,"%d",&(ckpts-
>nockpt)))
      != 1) {
      printf("Couldn't read checkpoint file %s
        !\n",fname);
      fclose(ckptfile);
      exit(1);
      }
   for (i=0;i<ckpts->nockpt;i++) {
      if ((fscanf(ckptfile,"%lf %lf %lf %lf %lf
        %lf %lf",&(ckpts->ckptxs[i]),
        &(ckpts->ckptys[i]),&(ckpts-
>ckptxe[i]),
        &(ckpts->ckptye[i]),
        &(ckpts->starttime[i]),&(ckpts-
        >endtime[i]),&(ckpts-
>newstarttime[i])))
        != 7) {
        printf("Couldn't read checkpoints\n");
        fclose(ckptfile);
        exit(1);
      }
   }
}

/*******************************************/
/* Subroutine to write out checkpoint files*/
/*******************************************/
void writeckpt(char *fname, struct ckptinfo
   *ckpts)
{
FILE *ckptfile;
int i;
   if ((ckptfile=fopen(fname,"wt")) == NULL) {
     printf("File %s cannot be
opened!",fname);
     exit(1);
   }
   if ((fprintf(ckptfile,"%d\n",ckpts-
>nockpt))
     == EOF) {
     printf("Couldn't write checkpoint file %s
     !\n",fname);
     fclose(ckptfile);
     exit(1);
   }
   for (i=0;i<ckpts->nockpt;i++) {
     if ((fprintf(ckptfile,"%lf %lf %lf %lf
%lf
       %lf %lf\n",ckpts->ckptxs[i],
       ckpts->ckptys[i],ckpts-
>ckptxe[i],ckpts-
       >ckptye[i],ckpts->starttime[i],
       ckpts->endtime[i],ckpts-
       >newstarttime[i])) == EOF) {
       printf("Couldn't write
checkpoints\n");
       fclose(ckptfile);
       exit(1);
     }
   }
   fclose(ckptfile);
}
/*******************************************/
/*      Subroutine to read input image     */
/*******************************************/
void readimage(char *fname,struct imginfo
*img)
{
FILE *infile;
int x,y;


   if ((infile=fopen(fname,"rb")) == NULL) {
   printf("File %s cannot be opened!",fname);
   exit(1);
   }
```

```
/*  Read the magic number */
  if ((getfilech(infile) != 'P') ||
     (getfilech(infile) != '6')) {
     printf("Bad magic number!\n");
     exit(1);
  }

/* Read number of rows and columns and max
  value */
  img->cols=getint(infile);
  img->rows=getint(infile);
  img->maxval=getint(infile);
  printf("Rows: %d  Cols: %d  Max value:
     %d\n",img->rows,img->cols,img->maxval);

/* Put aside some space and read an image */
  if (((img->imager=(unsigned char *)
     malloc(img->rows*(img->cols))) == NULL)
||
     ((img->imageg=(unsigned char *)
malloc(img-
     >rows*(img->cols))) == NULL) ||
     ((img->imageb=(unsigned char *)
malloc(img-
     >rows*(img->cols))) == NULL)) {
     printf("Could not allocate enough
     memory\n");
     exit(1);
  }

  for (y=0;y<img->rows;y++) {
     for (x=0;x<(img->cols);x++) {
        img->imager[y*(img->cols)+x]=
           getrawbyte(infile);
        img->imageg[y*(img->cols)+x]=
           getrawbyte(infile);
        img->imageb[y*(img->cols)+x]=
           getrawbyte(infile);
     }
  }

  fclose(infile);
}

/*******************************************/
/*   Subroutine to calculate the distance  */
/*            between 2 points             */
/*******************************************/
double calcdist(double lat1, double lon1,
  double lat2, double lon2)
{
double diffx,diffy,result;
  diffx=MPERDEGLON*(lon1-lon2);
  diffy=MPERDEGLAT*(lat1-lat2);
printf("X1 %lf Y1 %lf X2 %lf Y2 %lf diffx %lf
  diffy %lf\n",lon1,lat1,lon2,lat2,diffx,
  diffy);
  result=sqrt(diffx*diffx+diffy*diffy);
  printf("Result is %lf\n",result);
  return(result);
}

/*******************************************/
/*Subroutine to get character from image   */
/*                  file                   */
/*******************************************/
char getfilech(FILE *file)
{
  register int ich;
  register char ch;

  ich = getc( file );
  if ( ich == EOF ) {
    printf( "EOF / read error" );
    exit(1);
  }
  ch = (char) ich;
  if ( ch == '#' ) {
    do
       {
       ich=getc( file );
       if (ich == EOF) {
         printf( "EOF / read error" );
         exit(1);
       }
       ch = (char) ich;
     }
     while ( ch != '\n' && ch != '\r' );
  }
  return ch;
}

/*******************************************/
/* Subroutine to get the next byte from    */
/*                image                    */
/*******************************************/
unsigned char getrawbyte( FILE *file )

{
  register int iby;
  iby = getc( file );
  if ( iby == EOF ) {
    printf( "EOF / read error" );
    exit(1);
  }
  return (unsigned char) iby;
}

/*******************************************/
/* Subroutine to get an integer from image */
/*******************************************/
int getint( FILE *file )
{
  register char ch;
  register int i;

  do
    {
    ch = getfilech( file );
    }
  while ( ch == ' ' || ch == '\t' || ch ==
'\n'
    || ch == '\r' );
    if ( ch < '0' || ch > '9' ) {
      printf( "junk in file where an integer
        should be" );
      exit(1);
      }
    i = 0;
    do
      {
      i = i * 10 + ch - '0';
      ch = getfilech( file );
    }
  while ( ch >= '0' && ch <= '9' );
  return i;
}

/*******************************************/
/* Subroutine to sort coordinates by time  */
/*******************************************/
void ssort(int n,double a[],double b[],double
  c[],double d[],int e[])
{
int i,j,y;
double v,u,w,x;

  for (j=2;j<=n;j++) {
    v=a[j-1];
    u=b[j-1];
    w=c[j-1];
    x=d[j-1];
    y=e[j-1];
    i=j-1;
    while (i > 0 && a[i-1] < v) {
      a[i]=a[i-1];
      b[i]=b[i-1];
    c[i]=c[i-1];
    d[i]=d[i-1];
```

```
      e[i]=e[i-1];
      i--;
      }
    a[i]=v;
    b[i]=u;
    c[i]=w;

    d[i]=x;
    e[i]=y;
    }
}
```

APPENDIX F: CASE STUDIES

GPS Data Collection Summary - Baton Rouge

Notes:

1. Traffic type: AMP (am peak); PMP (pm peak); OP (off peak).
2. .ndc refers to the filename.ndc input data file containing both DGPS data and data that could not be differentially corrected.
3. .dc refers to the filename.dc containing only DGPS data.
4. DGPS (%) is the ratio of .dc to .ndc file sizes. It is a measure of the proportion of GPS points that were differentially correct.
5. Driver init. refers to the initials of the student who made the run in the field.
6. .nas refers to the filename.nas file that results after data reduction.
7. No. rec. refers to the number of records contained in each filename.nas file.
8. Data red. refers to the initials of the student who did the data reduction.

GPS DATA COLLECTION SUMMARY - BATON ROUGE

Date	Raw Data						Reduced Data			Comment
	Traffic type	File Name	.ndc file (bytes)	.dc file (bytes)	DGPS %	Driver init.	.nas file (bytes)	No. of Records	Data Reduction	
950516	OP	05162251	136,400	0	0	JM	20,088	372	MS	
	OP	05170125	146,916	0	0	JM	21,995	415	MS	
950530	OP	05292055	211,948	0	0	JM	28,938	546	MS	
950601	OP	04101705	240,768	0	0	JM	29,044	548	MS	
950605	OP	04141730	188,892	0	0	JM	22,631	427	MS	
	OP	04142132	145,200	0	0	JM	16,589	313	MS	
950606	OP	04151813	129,140	0	0	JM	16,536	312	MS	
	OP	04152049	192,236	0	0	JM	23,108	436	MS	
	OP	06052134	162,052	0	0	JM	15,105	285	MS	
	OP	06060110	119,944	0	0	JM	13,992	264	MS	
950607	OP	04161734	213,356	0	0	JM	21,995	415	MS	
	OP	04162146	135,652	0	0	JM	14,204	268	MS	
	OP	06062047	145,816	0	0	JM	22,580	426	MS	
	OP	06070016	179,740	0	0	JM	26,661	503	MS	
950608	OP	04172211	171,468	0	0	JM	23,320	440	MS	
	OP	06072036	215,072	0	0	JM	31,802	600	MS	
	OP	06080106	128,876	0	0	JM	13,409	253	MS	
950609	OP	04181727	222,024	0	0	JM	31,111	587	MS	
	OP	04182145	152,152	0	0	JM	13,727	259	MS	
	OP	06082044	133,716	0	0	JM	13,358	252	MS	
	OP	06090028	146,432	0	0	JM	13,621	257	MS	
950613	AMP	04221503	221,144	0	0	JM	29,258	552	MS	
	PMP	04222327	193,380	0	0	JM	24,804	468	MS	
950614	AMP	04231514	177,012	0	0	JM	20,193	381	MS	
	AMP	06131835	170,764	0	0	JM	24,910	470	MS	
	PMP	06140352	211,816	0	0	JM	25,546	482	MS	
950615	AMP	04241458	224,004	0	0	JM	27,191	513	MS	
	PMP	04242316	234,520	0	0	JM	17,914	338	MS	
	AMP	06141822	212,256	0	0	JM	19,769	373	MS	
	PMP	06150237	231,616	0	0	JM	28,567	539	MS	
950616	PMP	06160239	187,264	107,932	57.6	MA	23,400	468	MS	
	PMP	06162320	216,832	60,324	27.8	MA	19,150	383	MS	
950619	PMP	06191442	234,208	192,412	82.2	JM	27,250	545	MS	
	AMP	06191508	217,932	200,376	91.9	JM	27,878	526	MS	
950620	PMP	06200239	221,188	113,300	51.2	MA	25,758	486	MS	
	AMP	06200638	184,404	104,236	56.5	JM	24,168	456	MS	
	PMP	06201447	179,784	100,496	55.9	JM	24,857	469	MS	
	AMP	06201835	237,952	214,412	90.1	MA	26,924	508	MS	
950621	PMP	06210231	218,504	43,076	19.7	JM	20,034	378	MS	
	AMP	06210643	193,512	84,788	43.8	MA	27,927	527	MS	
	PMP	06211447	223,212	26,752	12.0	MA	27,196	523	MS	
	AMP	06211819	177,144	104,016	58.7	JM	20,193	381	MS	
950622	PMP	06220309	134,904	93,016	68.9	MA	14,628	276	MS	
	PMP	06221507	172,436	0	0	JM	21,677	409	MS	
	AMP	06221825	160,248	8,228	5.1	MA	21,677	409	MS	
950623	AMP	06230624	145,464	54,604	37.5	MA	22,578	426	MS	
	PMP	06231505	166,848	116,380	69.8	MA	22,790	430	MS	
950625	AMP	06250615	137,720	117,216	85.1	JM	19,750	395	MS	
	AMP	06251826	105,336	85,624	81.3	MA	13,900	278	MS	
950626	AMP	06260633	114,664	62,172	54.2	JM	20,140	380	MS	
	PMP	06261542	138,732	77,924	56.2	JM	20,460	387	MS	
950627	PMP	06270231	189,948	0	0	JM	22,419	423	MS	
	OP	06270824	198,396	4,092	2.1	MA	17,967	339	MS	
	PMP	06271447	179,388	0	0	MA	16,748	316	MS	
	OP	06272013	170,412	42,196	24.8	JM	17,596	332	MS	
950628	OP	06280846	221,408	0	0	JM	28,249	533	MS	
	OP	06281309	205,480	0	0	JM	26,712	504	MS	

GPS DATA COLLECTION SUMMARY - BATON ROUGE

Date	Raw Data						Reduced Data			Comment
	Traffic type	File Name	.ndc file (bytes)	.dc file (bytes)	DGPS %	Driver init.	.nas file (bytes)	No. of Records	Data Reduction	
950629	OP	06290056	279,180	53,988	19.3	JM	18,762	354	MS	
	AMP	06290702	180,488	106,744	59.1	MA	23,426	442	MS	
	OP	06291340	157,520	129,844	82.4	MA	17,543	331	MS	
	AMP	06291850	136,708	29,832	21.8	JM	21,783	411	MS	
950630	OP	06301337	263,692	237,248	90.0	MA	18,921	357	MS	
	OP	06301340	157,520	129,844	82.4	MA	17,543	331	MS	
950702	AMP	07020628	104,852	22,264	21.2	JM	16,700	334	MS	
	AMP	07021820	90,024	51,656	57.4	MA	16,052	321	MS	
950703	PMP	07030234	189,156	39,776	21.0	MA	18,815	355	MS	
	AMP	07030627	193,424	0	0	JM	19,239	363	MS	
	PMP	07031500	199,144	48,488	24.3	JM	19,186	362	MS	
	AMP	07031833	181,236	134,156	74.0	MA	17,914	338	MS	
950705	OP	07051351	139,568	28,160	20.2	MA	21,783	411	MS	
950706	AMP	07060616	169,136	73,260	43.3	JM	21,306	402	MS	
	PMP	07062004	189,948	157,564	83.0	MA	20,087	379	MS	
950707	AMP	07071059	227,040	208,296	91.7	MA	29,044	550	MS	
	PMP	07071920	457,908	447,524	97.7	MA	29,044	548	MS	
950709	PMP	07090234	189,156	39,776	21.0	JM	18,301	366	MS	
950711	OP	07111433	312,444	246,180	78.8	JM	27,772	524	MS	
	PMP	07111945	337,436	76,076	22.5	MA	27,984	529	MS	
950712	OP	07121348	356,268	163,768	45.9	JM	25,228	476	MS	
	PMP	07122148	166,452	0	0	MA	7,791	147	MS	
950713	AMP	07131120	215,072	0	0	JM	18,444	348	MS	
	OP	07131916	285,208	284,504	99.7	MA	22,633	427	MS	
950714	AMP	07141129	372,504	372,504	100	MA	18,785	383	MS	
	PMP	07142005	363,132	363,132	100	JM	12,857	262	MS	
950717	OP	07171225	363,572	363,572	100	JM	22,184	451	MS	
	OP	07171809	360,932	359,304	99.5	JM	22,067	450	MS	
950718	AMP	07181103	364,320	364,188	100	JM	22,104	450	MS	
	PMP	07181921	408,144	408,100	100	JM	22,550	451	MS	
950719	AMP	07191104	342,716	342,716	100	JM	23,005	460	MS	
	PMP	07191912	399,080	399,080	100	JM	22,959	459	MS	
950720	AMP	07201104	320,012	319,660	100	JM	24,750	495	MS	
	PMP	07201914	364,628	364,188	100	JM	24,500	490	MS	
950723	AMP	07231107	87,956	87,956	100	JM	6,350	135	MS	
	AMP	07231200	88,572	88,572	100	JM	6,900	138	MS	
950724	OP	07241357	373,824	373,824	100	MA	20,029	400	MS	
	OP	07241811	397,584	397,584	100	MA	19,700	394	MS	
950725	AMP	07251127	330,132	0	0	JM	25,150	503	MS	def. cable to PC
	OP	07251300	534,248	533,676	99.9	MA	54,204	1084	MS	
	OP	07251835	344,432	343,772	99.8	MA	35,750	715	MS	
	PMP	07251929	330,088	328,768	99.6	JM	26,053	521	MS	
950726	AMP	07261122	527,692	527,208	99.9	MA	54,450	1089	MS	
	PMP	07261921	388,828	388,828	100	MA	19,750	395	MS	
950727	OP	07271358	278,520	277,860	99.8	MA	18,600	372	MS	
	PMP	07271915	454,652	454,476	100	JM	26,850	537	MS	
950728	OP	07281259	330,924	330,880	100	JM	26,850	537	MS	
	PMP	07281914	372,680	372,680	100	JM	21,450	429	MS	
950730	AMP	07301035	98,736	98,736	100	JM	8,450	169	MS	
	AMP	07301124	88,880	88,880	100	JM	9,300	186	MS	
950731	OP	07311327	310,508	307,384	99.0	JM	24,351	487	MS	
	PMP	07311917	309,056	309,056	100	JM	21,001	420	MS	
950801	AMP	08011112	331,188	330,308	99.7	JM	26,850	537	MS	
	PMP	08011911	397,452	397,452	100	JM	26,000	520	MS	
950802	OP	08021317	287,452	287,452	100	JM	21,450	429	MS	
	OP	08021837	431,244	430,012	99.7	JM	27,352	547	MS	
950803	AMP	08031104	338,844	338,712	99.9	JM	25,650	514	MS	

GPS DATA COLLECTION SUMMARY - BATON ROUGE

Date	Raw Data						Reduced Data			Comment
	Traffic type	File Name	.ndc file (bytes)	.dc file (bytes)	DGPS %	Driver init.	.nas file (bytes)	No. of Records	Data Reduction	
	PMP	08031951	319,396	317,724	99.4	JM	20,000	400	MS	
950806	AMP	08061121	99,704	99,704	100	MA	7,250	145	MS	
	AMP	08061208	86,504	86,504	100	MA	7,050	141	MS	
950807	OP	08071310	420,244	418,044	99.4	JM	27,950	560	MS	
	PMP	08071926	431,508	431,332	99.9	JM	26,350	527	MS	
950808	AMP	08081121	395,296	393,888	99.6	JM	28,150	563	MS	
	OP	08081805	344,960	344,960	100	MA	24,300	486	MS	
	OP	08081926	352,132	351,824	99.9	JM	20,250	405	MS	
950809	AMP	08091120	407,132	407,132	100	JM	26,602	532	MS	
	AMP	08091124	355,564	355,564	100	MA	19,750	395	MS	
	PMP	08091930	409,772	0	0	MA	19,550	391	MS	
950810	AMP	08101105	310,024	309,936	99.9	JM	20,151	404	MS	
	OP	08101316	197,428	0	0	MA	18,251	365	MS	Prob with port
	OP	08101530	188,452	0	0	MA	18,250	365	MS	Prob with port
	OP	08101827	323,268	322,784	99.8	JM	20,050	401	MS	
	PMP	08101910	388,740	0	0	MA	18,700	374	MS	Prob with port
950811	AMP	08111117	301,620	0	0	MA	20,950	419	MS	Prob with port
	OP	08111809	302,324	0	0	MA	20,500	410	MS	Prob with port
950813	AMP	08130422	126,896	126,676	99.8	MA	7,050	142	MS	
	AMP	08130520	120,208	120,208	100	MA	6,100	122	MS	
	AMP	08131116	64,152	63,624	99.2	JM	7,150	143	MS	
	AMP	08131149	64152	64152	100	JM	7,300	147	MS	
950814	AMP	08140408	289,256	288,640	99.7	JM	20,600	412	MS	
950815	AMP	08150414	365,376	364,056	99.6	JM	19,400	388	MS	
	OP	08151228	95,612	89,892	94.0	JM	5,650	113	MS	Def. PC pow box
	OP	08151311	249,480	248,820	99.7	MA	20,900	418	MS	
	PMP	08151945	353,452	352,616	99.7	MA	18,150	363	MS	
950816	AMP	08160422	261,668	261,492	99.9	MA	21,889	413	MS	
	OP	08161101	370,084	366,740	99.1	MA	23,849	463	MS	
950817	OP	08170609	361,152	361,152	100	MA	18,150	363	MS	
	PMP	08171241	377,212	376,992	99.9	MA	36,273	717	MS	
	PMP	08171924	354,816	353,936	99.8	JM	38,160	720	MS	
950818	OP	08181359	203,060	201,916	99.4	MA	17,649	333	MS	
	OP	08181805	244,112	243,540	99.8	MA	11,289	213	MS	
	OP	08181814	374,440	374,220	99.9	JM	26,924	508	MS	
950820	AMP	08201104	188,144	0	0	MA	18,550	350	MS	
	AMP	08201107	86,680	86,108	99.3	JM	8,350	167	MS	
	AMP	08201149	97,460	97,240	99.8	JM	9,000	181	MS	
	AMP	08201231	178,508	0	0	MA	9,593	181	MS	
950821	PMP	08211953	361,196	360,052	99.7	MS	26,978	509	MS	
	PMP	08211956	208,736	207,856	99.8	JM	15,794	298	MS	
950822	AMP	08221106	373,472	372,504	99.7	JM	19,717	372	MS	
	PMP	08221853	544,456	543,708	99.9	MS	24,221	457	MS	
	PMP	08222003	371,668	368,940	99.3	JM	22,313	421	MS	
950823	AMP	08231202	431,508	431,288	99.9	MS	38,584	728	MS	
950824	AMP	08241119	362,032	362,032	100	JM	19,928	376	MS	
	OP	08241349	361,812	360,536	99.6	MS	26,288	496	MS	
950928	AMP	09281115	370,436	369,952	99.9	BM	31,148	599	DBH	
950929	AMP	09291153	202,928	0	0	BM	17,109	329	DBH	
951002	AMP	10021048	351,560	351,560	100	BM	31,304	602	DBH	
951003	AMP	10031110	143,220	143,220	100	BM	13,468	259	DBH	
	AMP	10031118	323,928	323,884	100	MS	22,516	433	DBH	
951004	AMP	10041128	324,588	324,324	99.9	BM	31,304	602	DBH	
951006	AMP	10061111	146,080	145,948	99.9	BM	13,208	254	DBH	
951010	AMP	10101136	204,556	204,380	100	BM	22,828	439	DBH	
951012	AMP	10121100	372,372	371,888	99.9	BM	16,536	312	MS	
	PMP	10122147	141,768	141,020	99.5	BP	7,844	148	MS	

GPS DATA COLLECTION SUMMARY - BATON ROUGE

Date	Raw Data						Reduced Data			Comment
	Traffic type	File Name	.ndc file (bytes)	.dc file (bytes)	DGPS %	Driver init.	.nas file (bytes)	No. of Records	Data Reduction	
951013	AMP	10131103	211,816	211,728	100	BM	7,420	140	MS	
	PMP	10132003	204,072	204,028	99.9	BP	13,104	252	DBH	
951016	AMP	10161100	348,832	348,436	100	BM	17,160	330	DBH	
	PMP	10162149	156,772	156,596	99.9	PT	11,284	217	DBH	
951018	AMP	10181052	392,656	392,524	99.9	BM	19,968	384	DBH	
	PMP	10182023	265,320	0	0	PT	22,308	429	DBH	
951019	AMP	10191129	270,160	269,852	99.9	MS	15,444	297	DBH	
	PMP	10192210	75,636	75,636	100	BP	6,760	130	DBH	
951020	AMP	10201054	385,924	385,924	100	BM	29,900	575	DBH	
	PMP	10202044	366,740	366,476	100	MS	27,560	530	DBH	
	PMP	10202049	199,760	199,760	100	BP	12,480	240	DBH	
	PMP	10202232	44,484	44,484	100	BP	2,288	44	DBH	
951023	AMP	10231106	319,748	319,748	100	BM	22,048	424	DBH	
	PMP	10232007	364,584	364,540	99.9	PT	33,540	645	DBH	
951024	AMP	10241105	328,064	328,020	99.9	MS	29,796	573	DBH	
	AMP	10241111	384,076	382,624	99.6	BM	29,120	560	DBH	
951025	AMP	10251051	327,668	327,668	100	BM	26,780	515	DBH	
	PMP	10252045	328,592	328,592	100	PT	20,852	401	DBH	
951026	AMP	10261111	310,420	310,288	99.9	BM	19,292	371	DBH	
	AMP	10261121	327,448	325,908	99.5	MS	30,056	578	DBH	
	OP	10261812	228,096	227,964	99.9	PT	22,464	432	DBH	
	PMP	10262032	187,440	187,440	100	BP	6,656	128	DBH	
951027	PMP	10272036	304,656	304,348	99.9	MS	15,964	307	DBH	
	PMP	10272046	339,944	339,944	100	BP	13,728	264	DBH	
951030	AMP	10301343	167,948	168,036	99.9	BM	14,508	279	DBH	
	PMP	10302120	333,652	332,288	99.6	PT	18,616	358	DBH	
951031	AMP	10311207	234,212	234,212	100	BM	22,620	435	DBH	
	AMP	10311352	109,164	108,988	99.8	MS	8,216	158	DBH	
951101	AMP	11011219	189,464	189,464	100	BM	17,680	340	DBH	
	PMP	11012131	274,604	274,560	99.9	PT	10,608	204	DBH	
951102	AMP	11021151	276,232	274,296	99.3	BM	22,984	442	DBH	
	AMP	11021222	303,468	303,160	99.9	MS	15,808	304	DBH	
	OP	11021924	168,124	168,124	100	PT	13,364	257	DBH	
	PMP	11022109	217,448	216,480	99.6	BP	10,504	202	DBH	
951103	AMP	11031153	198,176	196,724	99.3	BM	6,084	117	DBH	
	PMP	11032150	344,577	343,963	99.8	MS	15,964	307	DBH	
951106	AMP	11061210	245,432	245,432	100	BM	13,936	268	DBH	
951107	AMP	11071150	191,092	191,092	100	BM	16,900	325	DBH	
951108	PMP	11082111	343,684	341,704	99.4	PT	17,888	344	DBH	
951109	AMP	11091229	369,732	369,732	100	BM	22,100	425	DBH	
	AMP	11091249	54,340	54,340	100	MS	1,040	20	DBH	
	AMP	11091321	95,920	95,876	99.9	MS	2,704	52	DBH	
	PMP	11092217	184,272	183,876	99.8	BP	6,760	130	DBH	
951110	AMP	11101230	276,012	276,012	100	BM	16,900	325	DBH	
	PMP	11102136	237,424	237,336	99.9	BP	14,560	280	DBH	
	PMP	11102140	338,326	337,239	99.7	MS	18,824	362	DBH	
951113	AMP	11131236	413,468	413,380	99.9	BM	29,328	564	DBH	
	PMP	11132124	447,591	443,500	99.1	PT	21,736	418	DBH	
951114	AMP	11141232	315,436	315,392	99.9	BM	19,240	370	DBH	
951115	AMP	11151218	409,728	409,728	100	BM	29,172	561	DBH	
	PMP	11152055	358,600	357,984	99.8	PT	27,508	529	DBH	
951116	AMP	11161237	343,552	343,552	100	MS	26,728	514	DBH	
	AMP	11161239	182,380	182,380	100	BM	14,716	283	DBH	
	PMP	11162114	219,516	218,240	99.5	BP	12,480	240	DBH	
951117	AMP	11171223	183,040	182,952	99.9	BM	22,568	434	DBH	
	OP	11171920	207,504	206,844	99.7	PT	11,232	216	DBH	
	PMP	11172120	366,725	366,596	99.9	MS	27,612	531	DBH	

GPS DATA COLLECTION SUMMARY - BATON ROUGE

Date	Raw Data						Reduced Data			Comment
	Traffic type	File Name	.ndc file (bytes)	.dc file (bytes)	DGPS %	Driver init.	.nas file (bytes)	No. of Records	Data Reduction	
951120	AMP	11201235	86,460	86,460	100	BM	8,892	171	DBH	Battery ran out
	PMP	11202129	188,936	188,100	99.9	PT	8,112	156	DBH	
951121	AMP	11211223	190,960	190,740	99.9	BM	22,464	432	DBH	
	AMP	11211226	327,228	327,228	100	MS	26,364	507	DBH	
951122	AMP	11221216	326,260	324,148	99.4	BM	19,916	383	DBH	
951127	AMP	11271229	277,596	277,244	99.9	BM	19,916	384	DBH	
	PMP	11272119	239,888	239,888	100	PT	19,292	371	DBH	
951128	AMP	11281230	257,444	257,136	99.9	MS	16,120	310	DBH	
	AMP	11281238	380,952	380,908	100	BM	29,172	561	DBH	
	PMP	11282038	459,461	459,461	100	PT	25,428	489	DBH	
951129	AMP	11291215	333,784	333,520	99.9	BM	22,100	425	DBH	
951130	AMP	11301239	275,440	275,000	99.8	BM	19,760	380	DBH	
951201	AMP	12011218	282,832	282,304	99.8	BM	18,876	363	DBH	
	PMP	12012143	308,725	308,373	99.9	PT	12,688	244	DBH	
	PMP	12012208	42,944	42,944	100	MS	1,404	27	DBH	Batt. disconnect
	PMP	12012229	264,687	264,515	99.9	MS	8,424	162	DBH	Reconnect batt.
951204	AMP	12041229	303,468	303,380	100	BM	17,108	329	DBH	
	PMP	12042041	459,903	459,771	99.9	PT	20,176	388	DBH	
951205	PMP	12052144	238,392	238,260	99.9	PT	29,432	566	DBH	
951206	PMP	12062131	181,896	181,632	99.9	PT	8,060	155	DBH	
951207	AMP	12071228	137,192	137,192	100	BM	6,292	120	DBH	
	AMP	12071229	347,908	347,776	99.9	MS	19,552	376	DBH	
	PMP	12072134	319,711	319,366	99.9	BP	12,720	240	MS	
951208	AMP	12081234	179,916	179,916	100	BM	14,628	276	MS	
	PMP	12081942	530,640	530,640	100	BP	29,786	562	MS	
	PMP	12082151	351439	351,307	99.9	MS	20,020	385	DBH	
951211	AMP	12111216	141,460	141,460	100	BM	13,312	256	DBH	
	PMP	12112146	292,675	292,675	100	PT	16,692	321	DBH	
951212	AMP	12121228	218,152	218,152	100	BM	19,604	377	DBH	
	PMP	12122047	349,184	349,184	100	PT	18,252	351	DBH	
951213	AMP	12131220	302,456	301,444	100	BM	18,980	365	DBH	
	PMP	12132107	294,800	294,800	100	BP	22,828	439	DBH	
	PMP	12132111	316,712	316,668	99.9	PT	20,540	395	DBH	
951214	AMP	12141224	144,364	144,364	100	BM	6,240	120	DBH	
	PMP	12142203	263,723	261,694	99.2	BP	8,528	164	DBH	
951218	OP	12181440	214,632	214,368	99.8	MA	22,672	436	BP	
	OP	12181844	305,668	305,284	99.9	MA	19,610	370	MS	
	PMP	12182104	129,976	129,976	100	MA	8,586	162	MS	
951219	AMP	12191223	263,472	263,384	99.9	MA	13,727	259	MS	
	AMP	12191232	498,256	498,256	100	BM	38,796	732	MS	
	OP	12191840	178,552	178,508	99.9	BP	13,992	153	MS	
	OP	12191438	252,560	252,472	99.9	MA	8,109	264	MS	
	OP	12191901	169,708	169,708	100	MA	12,720	240	MS	
	PMP	12192122	187,440	187,088	99.8	MA	12,985	245	MS	
	PMP	12192123	235,312	235,092	99.9	BP	22,101	417	MS	
951220	OP	12201539	249,788	249,040	99.7	MA	12,243	231	MS	
	OP	12201608	163,460	163,460	100	MS	16,430	310	MS	
	OP	12201837	285,604	283,888	99.4	MS	10,653	201	MS	
	OP	12201838	256,608	256,608	100	MA	12,243	231	MS	
951221	OP	12211603	132,176	132,176	100	MA	10,335	195	MS	
	OP	12211632	193,952	193,952	100	MS	9,858	186	MS	
	OP	12211824	214,544	214,544	100	MS	11,130	210	MS	
	OP	12211854	250,448	250,404	99.9	MA	18,815	355	MS	
951222	OP	12221453	126,544	126,500	99.9	MA	7,950	150	MS	
951227	OP	12271523	182,116	180,576	99.1	MA	11,766	222	MS	
	OP	12271834	244,684	244,684	100	MA	17,119	323	MS	
951228	OP	12281510	221,540	221,540	100	MA	14,098	266	MS	

GPS DATA COLLECTION SUMMARY - BATON ROUGE

Date	Raw Data						Reduced Data			Comment
	Traffic type	File Name	.ndc file (bytes)	.dc file (bytes)	DGPS %	Driver init.	.nas file (bytes)	No. of Records	Data Reduction	
	OP	12281838	263,692	263,472	99.9	MA	23,055	435	MS	
960104	AMP	01041219	489,060	489,016	100	BM	28,461	537	MS	
960105	AMP	01051214	281,028	280,852	99.9	BM	22,154	418	MS	
	OP	01051423	138,424	138,424	100	BM	11,183	211	MS	
960108	AMP	01081229	329,384	329,384	100	BM	37,047	699	MS	
960109	AMP	01091217	287,496	287,496	100	BM	28,726	542	MS	
960111	AMP	01111207	307,252	306,768	99.8	BM	38,054	718	MS	
	PMP	01112151	350,986	350,810	99.9	BP	31,747	599	MS	
960112	AMP	01121215	386,144	385,704	99.9	BM	25,758	486	MS	
960116	AMP	01161223	281,204	281,204	100	BM	20,723	391	MS	
	AMP	01161240	35,948	35,948	100	MS	795	15	MS	Cas rec. not used
960117	AMP	01171211	280,412	280,324	100	BM	24,380	460	MS	
960118	AMP	01181239	169,884	169,884	100	BM	17,278	326	MS	
	AMP	01181301	150,876	150,788	99.9	MS	9,858	186	MS	
	PMP	01182207	264,683	264,121	99.8	BP	18,020	340	MS	
960119	AMP	01191216	365,992	365,948	99.9	BM	29,892	564	MS	
	PMP	01192324	95,165	95,165	100	MS	5,512	104	MS	
960122	AMP	01221219	306,064	306,064	100	BM	22,313	421	MS	
960123	AMP	01231234	258,544	258,500	99.8	BM	21,571	407	MS	
	AMP	01231245	212,124	212,124	100	MS	19,292	364	MS	
	PMP	01232135	251,108	250,360	99.7	BP	13,356	252	MS	
960124	AMP	01241210	186,912	186,648	99.9	BM	22,843	431	MS	
960125	AMP	01251211	148,940	148,852	99.9	BM	14,840	280	MS	
	PMP	01252157	161,216	161,040	99.9	BP	10,176	192	MS	
960129	AMP	01291212	361,284	361,284	100	BM	29,468	556	MS	
960130	AMP	01301220	308,880	308,880	100	BM	22,154	418	MS	
	PMP	01302229	228,519	228,255	99.9	BP	10,229	193	MS	
960131	AMP	01311211	306,856	306,856	100	BM	28,514	538	MS	
960201	AMP	02011222	241,208	241,208	100	BM	22,207	419	MS	
	PMP	02012242	365,953	365,821	100	BP	18,391	347	MS	
960205	AMP	02051223	241,076	241,076	100	BM	23,002	434	MS	
960206	AMP	02061228	241,824	241,164	99.7	BM	30,475	575	MS	
	PMP	02062200	372,964	327,876	100	BP	21,677	409	MS	
960207	AMP	02071212	383,680	383,592	99.9	BM	29,468	556	MS	
960208	AMP	02081240	209,924	209,704	99.9	BM	14,734	278	MS	
	PMP	02082258	215,092	215,006	100	BP	17,278	326	MS	
960209	AMP	02091222	345,400	328,636	95.1	BM	34,291	647	MS	
	PMP	02092047	67,276	67,276	100	MS	4,664	88	MS	
	PMP	02092122	359,492	357,784	99.9	MS	41,393	781	MS	
960212	AMP	02121222	315,744	315,744	100	BM	24,963	471	MS	
960213	AMP	02131208	238,744	238,744	100	BM	21,677	409	MS	
	PMP	02132229	287,846	287,846	100	BP	18,497	349	MS	
960215	AMP	02151210	249,436	248,600	99.6	BM	30,528	576	MS	
	PMP	02152159	75,108	75,108	100	BP	5,777	109	MS	
	PMP	02152258	236,715	236,715	100	BP	20,511	387	MS	
960216	AMP	02161230	280,764	280,764	100	BM	15,264	288	MS	
960222	AMP	02221215	278,960	272,052	97.5	BM	20,072	386	DBH	
	AMP	02221245	243,804	229,988	94.3	MS	15,288	294	DBH	
	PMP	02222208	323,224	318,340	98.5	BP	16,484	317	DBH	
960223	AMP	02231207	323,224	318,340	98.5	BM	21,788	419	DBH	
	PMP	02232217	250,223	249,348	99.7	MS	18,824	362	DBH	
960226	AMP	02261210	133,012	133,012	100	BM	13,364	257	DBH	
960227	AMP	02271208	182,116	181,280	99.5	BM	15,132	291	DBH	
	PMP	02272102	280,632	280,104	99.8	BP	13,052	251	DBH	
960228	AMP	02281214	150,744	150,744	100	BM	12,896	248	DBH	
960229	PMP	02292144	395,610	390,244	98.6	BP	16,748	316	MS	
960301	PMP	03012230	189,364	184,245	97.3	MS	10,229	193	MS	

GPS DATA COLLECTION SUMMARY - BATON ROUGE

Date	Raw Data						Reduced Data			Comment
	Traffic type	File Name	.ndc file (bytes)	.dc file (bytes)	DGPS %	Driver init.	.nas file (bytes)	No. of Records	Data Reduction	
960305	AMP	03051227	333,476	331,760	99.4	MS	17,967	339	MS	
	PMP	03052139	278,640	275,210	98.8	BM	29,574	559	MS	
	PMP	03052231	304,841	302,886	99.4	BP	21,200	400	MS	
960306	PMP	03062132	239,360	237,996	99.4	BM	23,055	435	MS	
960307	PMP	03072140	272,508	272,904	99.9	BM	26,977	509	MS	
	PMP	03072149	359,068	359,068	100	BP	25,069	473	MS	
960308	PMP	03082155	85,976	76,472	88.2	MS	4,264	82	DBH	Prob with power
	PMP	03082247	154,271	152,585	98.7	MS	8,684	167	DBH	Prob with power
960312	AMP	03121252	291,896	286,748	98.2	MS	14,664	282	BP	Prob with power
960314	AMP	03141235	295,900	295,856	100	MS	16,066	309	BP	
960315	PMP	03152242	205,377	202,195	98.5	MS	10,816	208	BP	Prob DCI cord
960319	AMP	03191308	192,236	192,192	99.7	MS	8,424	160	BP	DCI battery prob
	PMP	03192213	177,892	0	0	BP	17,108	329	BP	
960321	AMP	03211241	289,432	289,168	100	MS	13,052	250	BP	
	PMP	03212143	211,596	211,068	99.8	BM	12,948	249	BP	
960322	PMP	03222141	196,372	194,920	99.3	BM	22,620	435	BP	
	PMP	03222146	163,020	162,668	99.8	MS	12,688	244	BP	DCI cord unplug
	PMP	03222303	63,140	63,140	100	MS	4,420	85	BP	
960325	PMP	03252157	153,912	153,912	100	BM	14,508	279	BP	
960326	AMP	03261319	276,452	276,320	99.9	MS	15,912	306	BP	
	PMP	03262139	243,100	242,616	99.8	BM	14,664	281	BP	
	PMP	03262145	320,565	314,317	98.0	BP	21,372	410	BP	
960328	PMP	03282148	153,560	149,908	97.6	BM	10,556	203	BP	Rec. malfunction
	PMP	03282225	263,678	263,415	99.9	BP	22,005	422	BP	
960329	PMP	03292151	192,588	191,004	99.2	BM	21,892	421	BP	
960409	AMP	04091213	284,152	283,712	99.8	MS	12,324	237	DBH	
	PMP	04092118	391,214	391,041	100	BP	21,842	420	BP	
960411	AMP	04111132	305,052	305,008	100	MS	14,404	277	BP	
	PMP	04112202	302,393	301,909	99.8	BP	26,572	511	BP	
960412	PMP	04122113	114,444	114,268	100	MS	9,360	180	BP	Prob with chord
	PMP	04122226	144,672	144,408	100	MS	9,932	191	BP	same
960416	AMP	04161046	294,228	294,228	100	MS	13,936	268	BP	
	PMP	04162056	183,612	183,172	99.8	BM	17,004	327	BP	
	PMP	04162102	381,926	381,926	100	BP	24,076	461	BP	
960417	PMP	04172102	155,100	154,396	99.5	BM	15,184	292	BP	
960418	PMP	04182107	278,168	277,816	99.9	BM	11,388	219	BP	
960419	PMP	04192042	195,184	194,920	99.9	BM	8,112	156	BP	
960423	PMP	04232033	297,132	296,780	99.9	BM	1,924	37	BP	
	PMP	04232215	241,293	239,634	99.3	BP	13,988	389	BP	
	AMP	04231205	80,212	72,644	90.6	MS	20,228	269	BP	Battery ran out
960425	PMP	04252116	246,444	246,268	99.9	BM	6,604	127	BP	
	PMP	04252038	377,036	376,684	99.9	BP	23,036	443	BP	
	AMP	04251150	204,996	198,132	96.7	MS	11,440	220	BP	
960426	PMP	04262018	259,688	259,688	100	MS	16,692	321	BP	
960430	AMP	04301114	229,812	226,952	98.8	MS	8,632	166	BP	
	PMP	04302057	280,896	280,588	99.9	BM	11,544	222	BP	
	PMP	04302144	251,195	249,804	99.4	BP	10,088	194	BP	
960502	PMP	05022158	303,749	303,177	99.8	BP	12,844	247	BP	
960514	OP	05141354	294,624	292,820	99.4	MN	14,352	269	DBH	
	OP	05141832	264,616	264,000	99.8	MN	28,288	544	DBH	
960515	OP	05151506	165,880	164,780	99.3	MN	15,236	293	DBH	
	OP	05151856	117,656	115,368	98.0	MN	9,308	179	DBH	
960516	OP	05161316	315,656	313,808	99.4	MN	22,360	430	DBH	
	OP	05161916	259,248	0	0	MN	6,864	132	DBH	Prob. with DCI
960517	OP	05171456	281,292	280,676	99.8	MN	31,200	600	DBH	
	OP	05171932	118,756	118,580	99.9	MN	7,332	141	DBH	
960520	OP	05201359	87,868	85,844	97.7	MN	8,528	164	DBH	

GPS DATA COLLECTION SUMMARY - BATON ROUGE

Date	Raw Data						Reduced Data			Comment
	Traffic type	File Name	.ndc file (bytes)	.dc file (bytes)	DGPS %	Driver init.	.nas file (bytes)	No. of Records	Data Reduction	
	OP	05201833	381,436	378,400	99.2	MN	9,724	187	DBH	
960521	OP	05211335	176,616	169,840	96.1	MN	18,304	352	DBH	
960522	AM	05221201	186,208	182,908	98.2	MN	17,784	342	DBH	
960614	PM	06141519	228,008	227,128	99.6	MN	17,940	345	DBH	
960617	AM	06170639	217,448	217,096	99.8	MN	23,656	439	DBH	
	PM	06171530	209,924	209,352	99.7	MN	22,672	436	DBH	
960620	AM	06200650	295,196	294,624	99.8	MN	32,768	630	DBH	
	PM	06201625	122,056	121,924	99.8	MN	10,764	207	DBH	
960624	PM	0624m001	180,586	0	0	MN	3,380	65	DBH	Magel rec. used
960625	AM	0625m001	244,446	0	0	MN	22,106	425	DBH	Magel rec. used
	PM	0625m002	246,186	0	0	MN	22,256	427	DBH	Magel rec. used
960701	PM	07011539	210,144	207,504	98.7	MN	21,996	423	DBH	
960710	AM	07100703	206,228	101,024	48.9	MN	21,996	423	DBH	
960712	PM	07121557	163,196	12,980	7.9	MN	10,920	210	DBH	
960716	AM	07160624	56,540	55,792	98.6	MN	6,864	132	DBH	
	AM	07160756	83,336	78,452	94.1	MN	9,724	187	DBH	
	PM	07161518	247,896	247,852	99.9	MN	12,740	245	DBH	
960717	AM	07170630	190,388	189,552	99.5	MN	16,796	323	DBH	
	PM	07171524	195,272	12,540	6.4	MN	21,164	407	DBH	
960718	AM	07180618	296,604	40,964	13.8	MN	32,916	633	DBH	
960722	AM	07220647	249,260	248,820	99.8	MN	29,796	573	DBH	
	PM	07221517	176,660	176,044	99.6	MN	21,788	419	DBH	
960723	AM	07230624	81,356	27,148	33.3	MN	10,348	199	DBH	
	PM	07231554	229,768	229,372	99.8	MN	21,944	422	DBH	
960724	AM	07240619	178,948	178,948	100	MN	21,944	422	DBH	
	PM	07241537	258,808	258,808	100	MN	22,048	424	DBH	
960725	AM	07250638	182,204	182,204	100	MN	21,944	422	DBH	
960726	AM	07260648	313,456	313,456	100	MN	38,220	735	DBH	
	PM	07261532	113,960	113,960	100	MN	10,660	205	DBH	

GPS Data Collection Summary - Shreveport

Notes:

1. Traffic type: AMP (am peak); PMP (pm peak); NP (noon peak); OP (off peak).
2. .asc refers to the filename.asc raw data file sent from Shreveport.
3. .dc refers to the filename.dc after conversion into format suitable for data reduction
4. DGPS (%) is the ratio of .dc to .ndc file sizes (not applicable)
5. Driver init. refers to the initials of the person who made the run in the field.
6. .nas refers to the filename.nas file that results after data reduction.
7. No. rec. refers to the number of records contained in each filename.nas file.
8. Data red. refers to the initials of the student who did the data reduction.
9. Original .asc file did not contain speed data.
10. Segment 68 is located on Spring St between Travis and Fannin. No GPS data was collected for 35 seconds. GPSSPEED was assumed to be the same as AVSPEED because the latter value was considered to be more reliable.
11. For simplicity, 60,000 was subtracted from all time stamp values to make the file consistent with the rest of off-peak runs.
12. Segment 70 is located on Market St between Travis and Texas. No GPS data was collected for 42 seconds. GPSSPEED was assumed to be the same as AVSPEED because the latter value was considered to be more reliable.

GPS DATA COLLECTION SUMMARY - SHREVEPORT

Date	Raw Data						Reduced Data			Comment
	Traffic type	File Name	.asc file (bytes)	.dc file (bytes)	DGPS %	Driver init.	.nas file (bytes)	No. of Records	Data Reduction	
950711	AMP	r071107a	59,585	30,360		CP	3,109	62	BP	
	NP	r071111a	96,985	49,676		CP	1,936	39	BP	
950712	AMP	r071207a	66,215	33,440		CP	3,062	61	BP	
950713	AMP	r071307a	64,260	32,780		CP	2,540	50	BP	
950725	AMP	r072507a	77,435	39,468		CP	2,442	48	BP	
950726	PMP	r072616a	61,030	31,064		CP	1,605	32	BP	
950727	OP	r072722a	60,350	30,404		CP	1,997	40	BP	
	OP	r072722b	80,920	40,876		CP	3,107	62	BP	
	OP	r072723a	88,825	45,140		CP	3,558	71	BP	
	OP	r072723b	55,505	28,336		CP	1,600	32	BP	
950728	OP	r072800a	57,460	28,864		CP	2,356	47	BP	
	OP	r072802a	72,505	37,048		CP	2,527	50	BP	
	OP	r072802b	70,975	36,168		CP	2,527	50	BP	
	OP	r072803a	53,720	27,632		CP	2,197	44	BP	
950822	OP	r082220a	58,735	28,939		CP	3,109	62	BP	
	OP	r082221a	64,600	32,035		CP	1,936	39	BP	
	OP	r082221b	78,540	39,466		CP	1,986	40	BP	
	OP	r082221c	56,015	28,468		CP	1,605	32	BP	
	OP	r082222a	85,000	43,076		CP	3,051	61	BP	
	OP	r082222b	93,670	47,388		CP	3,597	72	BP	
	OP	r082223a	97,410	49,720		CP	3,495	70	BP	
950823	OP	r082300a	44,795	22,396		CP	2,336	46	BP	
	OP	r082300b	73,525	37,620		CP	2,442	48	BP	
	OP	r082301a	87,380	44,484		CP	3,502	70	BP	
950829	AMP	r082907a	62,815	0		CP	0	0	---	note 9
950927	AMP	r092707a	42,160	21,384		CP	2,236	44	BP	
950928	NP	r092812a	81,005	41,140		CP	1,936	39	BP	
951003	AMP	r100307a	72,505	36,652		CP	2,156	43	BP	
951004	AMP	r100407a	62,730	31,988		CP	2,426	48	BP	
951005	AMP	r100507a	76,160	39,028		CP	2,578	51	BP	
951205	AMP	r120507a	90,270	46,244		CP	2,527	50	BP	
951206	AMP	r120607a	78,880	40,304		CP	2,527	50	BP	
951207	AMP	r120707a	117,300	59,532		CP	3,597	72	BP	
951212	AMP	r121207a	98,260	50,116		CP	2,527	50	BP	
951213	AMP	r121307a	95,710	48,840		CP	2,425	48	BP	
951214	AMP	r121407a	134,300	68,640		CP	3,302	66	BP	
960116	PMP	r011616a	90,865	46,464		CP	2,527	50	BP	
960117	PMP	r011716a	83,725	42,680		CP	2,527	50	BP	
960118	AMP	r011807a	58,310	29,612		CP	1,605	32	BP	
960123	AMP	r012307a	97,070	49,104		CP	3,107	62	BP	
960125	PMP	r012516a	60,860	30,932		CP	2,247	45	BP	
	PMP	r012516b	106,930	54,736		CP	2,527	50	BP	
960213	PMP	r021316a	58,225	29,260		CP	2,156	43	BP	note 10
960214	AMP	r021407a	98,345	49,940		CP	3,107	62	BP	
960215	AMP	r021507a	101,490	51,700		CP	3,203	63	BP	
960220	PMP	r022016a	51,340	26,092		CP	2,442	48	BP	
960221	AMP	r022107a	110,330	56,276		CP	3,303	65	BP	
960222	AMP	r022207a	60,690	30,976		CP	1,605	32	BP	
960227	AMP	r022707a	46,835	23,628		CP	2,556	51	BP	
	AMP	r022707b	113,560	57,508		CP	3,595	72	BP	
960228	AMP	r022807a	66,045	33,616		CP	1,600	32	BP	
	AMP	r022807b	50,660	25,520		CP	4,205	52	BP	
	PMP	r022816a	64,515	33,000		CP	1,600	32	BP	
	PMP	r022816b	79,730	40,348		CP	2,136	43	BP	
960229	AMP	r022907a	129,370	65,956		CP	3,597	72	BP	
960305	AMP	r030507a	115,685	58,608		CP	3,452	69	BP	
960306	PMP	r030616a	94,180	47,608		CP	1,936	39	BP	

GPS DATA COLLECTION SUMMARY - SHREVEPORT

Date	Raw Data						Reduced Data			Comment
	Traffic type	File Name	.asc file (bytes)	.dc file (bytes)	DGPS %	Driver init.	.nas file (bytes)	No. of Records	Data Reduction	
960307	AMP	r030707a	42,245	21,472		CP	2,285	45	BP	
960312	AMP	r031207a	130,645	66,352		CP	3,595	72	BP	
960313	AMP	r031307a	69,700	35,640		CP	1,600	32	BP	
960320	PMP	r032016a	55,845	28,116		CP	2,048	41	BP	
	PMP	r032016b	90,185	45,540		CP	3,107	62	BP	
960328	PMP	r032816a	62,135	31,460		CP	2,703	54	BP	
960411	PMP	r041115a	61,285	30,932		CP	2,703	54	BP	
960412	PMP	r041215a	63,580	32,340		CP	2,287	45	BP	
960418	PMP	r041815a	54,825	27,896		CP	2,287	45	BP	
960419	PMP	r041916a	122,740	62,964		CP	3,302	66	BP	
960420	PMP	r042016a	113,050	57,464		CP	3,302	66	BP	
960502	AMP	r050206a	66,045	34,232		CP	2,247	45	BP	
	PMP	r050216a	128,605	65,384		CP	3,597	72	BP	
960503	AMP	r050306a	71,655	36,300		CP	2,247	45	BP	
	NP	r050311a	88,230	45,012		CP	2,036	41	BP	
	PMP	r050315a	120,275	61,336		CP	3,597	72	BP	
960504	AMP	r050406a	70,635	35,904		CP	1,997	40	BP	
	NP	r050410a	86,955	44,572		CP	1,787	36	BP	
960505	AMP	r050506a	64,770	32,868		CP	1,946	39	BP	
960507	PMP	r050715a	62,560	31,944		CP	2,197	44	BP	
	PMP	r050716a	71,145	36,432		CP	2,426	48	BP	
960509	AMP	r050906a	106,250	54,076		CP	2,990	61	BP	
	PMP	r050916a	111,095	56,276		CP	2,527	50	BP	
960510	AMP	r051006a	103,615	52,492		CP	3,051	61	BP	
	PMP	r051016a	94,860	48,312		CP	2,528	50	BP	
960521	PMP	r052115a	54,531	27,563		CP	1,997	40	JG	
	PMP	r052116a	69,275	35,376		CP	2,476	49	BP	
960522	PMP	r052215a	94,350	47,520		CP	3,051	61	BP	
960528	PMP	r052815a	43,990	22,145		CP	2,340	46	JG	
960530	OP	r053012a	63,325	31,988		CP	2,427	49	BP	note 11
	PMP	r053015a	92,225	47,212		CP	1,897	39	BP	
	PMP	r053016a	84,830	43,164		CP	2,036	41	BP	
960604	PMP	r060415a	98,600	50,072		CP	3,107	62	BP	note 12
	PMP	r060416a	64,770	33,088		CP	1,600	32	BP	
960613	PMP	r061316a	94,945	48,356		CP	3,051	61	BP	
	PMP	r061316b	63,080	31,906		CP	2156	43	JG	
960619	PMP	r061915a	105,659	53,965		CP	3595	72	JG	
960723	PMP	r072316a	113,876	58,007		CP	3595	72	JG	
960730	PMP	r073016a	62,997	32,078		CP	3013	60	JG	
	PMP	r073016b	61,005	31,089		CP	1605	32	JG	
960731	PMP	r073116a	101,094	51,858		CP	3303	66	JG	
960807	PMP	r080716a	62,914	31,863		CP	3013	60	JG	
960821	PMP	r082116a	30,576	15,437		CP	703	14	JG	
960822	OP	r082214a	48,804	25,168		CP	506	10	JG	note 11

GPS Data Collection Summary - New Orleans

Notes:

1. Traffic type: AMP (am peak); PMP (pm peak); OP (off peak).
2. .ndc refers to the filename.ndc input data file containing both DGPS data and data that could not be differentially corrected.
3. .dc refers to the filename.dc containing only DGPS data.
4. DGPS (%) is the ratio of .dc to .ndc file sizes. It is a measure of the proportion of GPS points that were differentially correct.
5. Driver init. refers to the initials of the student who made the run in the field.
6. .nas refers to the filename.nas file that results after data reduction.
7. No. rec. refers to the number of records contained in each filename.nas file.
8. Data red. refers to the initials of the student who did the data reduction.
9. Information about the .dc file size was missing on fax sent by RPC.

GPS DATA COLLECTION SUMMARY - NEW ORLEANS

Date	Raw Data						Reduced Data			Comment
	Traffic type	File Name	.asc file (bytes)	.dc file (bytes)	DGPS %	Driver init.	.nas file (bytes)	No. of Records	Data Reduction	
960611	AMP	06110634	287,980	257,664	89.5		20,035	401	JG	-
	PMP	06111558	170,104	0	0		33,211	657	JG	-
	PMP	06111603	336,908	277,552	82.4		29,627	593	JG	-
	AMP	06111911	180,136	0	0		35,021	692	JG	-
960612	AMP	06120642	92,796	0	0		19,289	386	JG	-
	AMP	06120806	66,044	0	0		12,443	249	JG	-
	PMP	06121606	100,584	86,768	86.3		15,209	301	JG	-
	PMP	06121741	58,652	58,520	99.8		11,655	230	JG	-
960613	AMP	06130631	180,224	60,544	33.6		35,877	709	JG	-
	PMP	06131550	178,244	72,908	40.9		34,911	691	JG	-
960617	PMP	06171616	125,752	4,664	3.7		11,656	229	JG	-
960618	AMP	06180637	145,948	24,992	17.1		16,536	325	JG	-
	PMP	06181604	148,808	24,464	16.4		15,824	305	DBH	-
960619	AMP	06190718	107,140	54,648	51.0		12,365	243	JG	-
960620	AMP	06200658	110,044	0	0		12,357	244	JG	-
	PMP	06201613	139,524	139,436	99.9		15,770	310	JG	-
960624	AMP	06240642	303,204	0	0		19,329	379	JG	-
960625	PMP	06251505	235,532	233,508	99.1		45,637	903	JG	-
	PMP	06251531	254,760	254,232	99.8		16,173	319	DBH	-
	PMP	06251745	28,248	28,248	100		2,293	45	DBH	-
960626	AMP	06260633	70,752	70,708	99.9		14,560	288	DBH	-
	AMP	06260640	299,156	299,156	100		22,380	441	DBH	-
	AMP	06260737	86,196	85,976	99.7		16,267	322	JG	-
	PMP	06261539	165,572	165,528	99.9		31,280	619	JG	-
	PMP	06261604	267,520	264,616	98.9		17,729	349	JG	-
960627	AMP	06270639	276,100	275,440	99.8		18,186	358	JG	-
	AMP	06270642	170,808	170,192	99.6		32,401	641	JG	-
	PMP	06271604	311,080	309,936	99.6		19,116	377	JG	-
	PMP	06271613	155,628	155,188	99.7		29,525	584	JG	-
960628	AMP	06280628	167,376	166,672	99.6		35,913	711	JG	-
	AMP	06280654	60,940	60,940	100		5,088	100	JG	-
	AMP	06280728	200,992	199,760	99.4		13,031	257	JG	-
	PMP	06281540	170,940	169,840	99.4		30,477	603	JG	-
	PMP	06281601	43,516	42,108	96.8		1,914	38	JG	-
	PMP	06281626	252,428				17,153	338	JG	Note 9
960701	AMP	07010643	149,600	148,544	99.3		33,678	674	JG	-
	AMP	07010652	268,268				25,373	502	JG	Note 9
960710	AMP	07100634	119,812				25,437	509	DBH	Note 9
	AMP	07100828	33,660	33,264	98.8		7,696	154	DBH	-
	PMP	07101546	163,416	162,844	99.6		30,034	601	DBH	-
960711	AMP	07110633	166,188	166,144	99.9		34,983	692	JG	-
	AMP	07110644	56,452	56,452	100		4,846	97	DBH	-
	AMP	07110721	224,840	224,708	99.9		23,933	479	JG	-
	PMP	07111548	191,268	190,652	99.7		29,241	579	JG	-
	PMP	07111559	209,264	208,824	99.8		20,485	410	JG	-
960712	AMP	07120627	333,256	330,220	99.1		34,625	693	JG	-
	AMP	07120642	110,308	109,868	99.6		20,468	405	DBH	-
	AMP	07120818	50,512	50,512	100		7,021	139	DBH	-
960715	AMP	07150634	56,188	56,144	99.9		13,093	262	DBH	-
	AMP	07150737	81,048	80,916	99.8		17,292	346	DBH	-
	AMP	07150838	64,680	64,504	99.7		6,301	124	JG	-
	PMP	07151601	135,080	133,408	98.8		28,430	569	JG	-
	PMP	07151627	255,992	255,684	99.9		28,863	571	DBH	-
960716	AMP	07160630	160,116	159,676	99.7		25,686	514	DBH	-
	AMP	07160648	302,940	302,544	99.9		32,513	643	DBH	-
	PMP	07161600	196,592	196,460	99.9		20,689	409	DBH	-
	PMP	07161602	140,052	138,160	98.6		28,895	580	JG	-

GPS DATA COLLECTION SUMMARY - NEW ORLEANS

Date	Raw Data						Reduced Data			Comment
	Traffic type	File Name	.asc file (bytes)	.dc file (bytes)	DGPS %	Driver init.	.nas file (bytes)	No. of Records	Data Reduction	
960717	AMP	07170638	150,260	149,952	99.8		33,935	679	JG	
	PMP	07171547	188,144	187,704	99.8		35,380	700	JG	-
	PMP	07171611	285,032	281,996	98.9		12,481	257	DBH	-
960718	AMP	07180641	142,604	141,504	99.2		19,188	384	DBH	-
	PMP	07181542	157,036	153,076	97.5		21,391	428	DBH	-
	PMP	07181559	323,840	321,596	99.3		21,391	428	DBH	-
960719	AMP	07190631	318,428	317,768	99.8		19,737	395	DBH	-
	PMP	07191553	108,592	107,492	99.0		18,089	362	DBH	-
	PMP	07191731	52,448	50,556	96.4		10,746	215	DBH	-
	PMP	07191745	76,780	76,516	99.7		7,744	155	DBH	-
960722	PMP	07221558	319,264	317,900	99.6		25,582	512	DBH	-

This public document is published at a total cost of $1269.37. Two hundred copies of this public document were published in this first printing at a cost of $867.37. The total cost of all printings of this document including reprints is $1269.37. This document was published by Louisiana State University, Graphic Services, 3555 River Road, Baton Rouge, Louisiana 70802, to report and publish research findings of the Louisiana Transportation Research Center as required by R.S.48:105. This material was printed in accordance with standards for printing by state agencies established pursuant to R.S.43:31. Printing of this material was purchased in accordance with the provisions of Title 43 of the Louisiana Revised Statutes.

LTRC DISTRIBUTION CHECKLIST

STUDY TITLE DEVELOPMENT OF A CONGESTION MANAGEMENT SYSTEM USING GPS TECHNOLOGY REPORT NO. 299

TYPE OF REPORT ☒ FINAL ☐ INTERIM ☐ TECHNICAL SUMMARY

GROUP MANAGER ART ROGERS

LA DOTD

- ☒ LTRC GROUP MANAGERS ~~(4)~~ 5
- ☒ PROJECT REVIEW COMMITTEE (8)
- ☒ SECRETARY, DEPUTY SECRETARY (2)
- ☒ CHIEF ENGINEER (1)
- ☒ POLICY COMMITTEE (8)
- ☒ TECHNICAL ADMINISTRATORS (28)
- ☒ DOTD CHIEFS (7)
- ☐ DISTRICT CONSTRUCTION ENGINEERS (9)
- ☐ DISTRICT MAINTENANCE ENGINEERS (9)
- ☒ DISTRICT TRAFFIC ENGINEERS (9)
- ☐ DISTRICT DES/WATER RES ENGINEERS (9)
- ☐ DISTRICT LABORATORY ENGINEERS (9)
- ☐ DISTRICT PROJECT ENGINEERS (9)
- ☐ OTHERS ____________

LA UNIVERSITIES

- ☐ DEANS, COLLEGE OF ENGINEERING (7)
- ☒ PRINCIPAL INVESTIGATORS (1)

STATES

- ☐ SASHTO STATE RESEARCH (RAC), 27 COPIES
- ☒ RESEARCH GROUP HEADS (50) ☐ OTHERS ____________