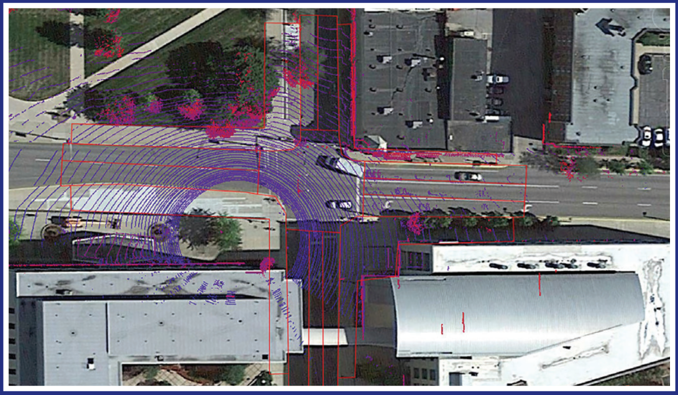
JOINT TRANSPORTATION RESEARCH PROGRAM

INDIANA DEPARTMENT OF TRANSPORTATION AND PURDUE UNIVERSITY

TScan: Stationary LiDAR for Traffic and Safety Studies—Object Detection and Tracking



Andrew P. Tarko, Kartik B. Ariyur, Mario A. Romero, Vamsi Krishna Bandaru, Cristhian G. Lizarazo

SPR-3831 • Report Number: FHWA/IN/JTRP-2016/24 • DOI: 10.5703/1288284316347

RECOMMENDED CITATION

Tarko, A. P., Ariyur, K. B., Romero, M. A., Bandaru, V. K., & Lizarazo, C. G. (2016). *TScan: Stationary LiDAR for traffic and safety studies—Object detection and tracking* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2016/24). West Lafayette, IN: Purdue University. http://dx.doi.org/10.5703/1288284316347

AUTHORS

Andrew P. Tarko, PhD
Professor of Civil Engineering
Lyles School of Civil Engineering
Purdue University
(765) 494-5027
tarko@purdue.edu
Corresponding Author

Kartik B. Ariyur, PhD
Visiting Assistant Professor
School of Mechanical Engineering
Purdue University

Mario A. Romero, PhD
Center for Road Safety Research Scientist
Lyles School of Civil Engineering
Purdue University

Vamsi Krishna Bandaru
Graduate Research Assistant
School of Mechanical Engineering
Purdue University

Cristhian G. Lizarazo
Graduate Research Assistant
Lyles School of Civil Engineering
Purdue University

JOINT TRANSPORTATION RESEARCH PROGRAM

NOTICE

COPYRIGHT

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. FHWA/IN/JTRP-2016/24	**2. Government Accession No.**	**3. Recipient's Catalog No.**
4. Title and Subtitle TScan: Stationary LiDAR for Traffic and Safety Studies—Object Detection and Tracking		**5. Report Date** August 2016 **6. Performing Organization Code**
7. Author(s) Andrew P. Tarko, Kartik B. Ariyur, Mario A. Romero, Vamsi Krishna Bandaru, Cristhian G. Lizarazo		**8. Performing Organization Report No.** FHWA/IN/JTRP-2016/24
9. Performing Organization Name and Address Joint Transportation Research Program Purdue University 550 Stadium Mall Drive West Lafayette, IN 47907-2051		**10. Work Unit No.** **11. Contract or Grant No.** SPR-3831
12. Sponsoring Agency Name and Address Indiana Department of Transportation State Office Building 100 North Senate Avenue Indianapolis, IN 46204		**13. Type of Report and Period Covered** Final Report **14. Sponsoring Agency Code**

15. Supplementary Notes
Prepared in cooperation with the Indiana Department of Transportation and Federal Highway Administration.

16. Abstract

The ability to accurately measure and cost-effectively collect traffic data at road intersections is needed to improve their safety and operations. This study investigates the feasibility of using laser ranging technology (LiDAR) for this purpose. The proposed technology does not experience some of the problems of the current video-based technology but less expensive low-end sensors have limited density of points where measurements are collected that may bring new challenges. A novel LiDAR-based portable traffic scanner (TScan) is introduced in this report to detect and track various types of road users (e.g., trucks, cars, pedestrians, and bicycles). The scope of this study included the development of a signal processing algorithm and a user interface, their implementation on a TScan research unit, and evaluation of the unit performance to confirm its practicality for safety and traffic engineering applications.

The TScan research unit was developed by integrating a Velodyne HDL-64E laser scanner within the existing Purdue University Mobile Traffic Laboratory which has a telescoping mast, video cameras, a computer, and an internal communications network. The low-end LiDAR sensor's limited resolution of data points was further reduced by the distance, the light beam absorption on dark objects, and the reflection away from the sensor on oblique surfaces. The motion of the LiDAR sensor located at the top of the mast caused by wind and passing vehicles was accounted for with the readings from an inertial sensor atop the LiDAR. These challenges increased the need for an effective signal processing method to extract the maximum useful information.

The developed TScan method identifies and extracts the background with a method applied in both the spherical and orthogonal coordinates. The moving objects are detected by clustering them; then the data points are tracked, first as clusters and then as rectangles fit to these clusters. After tracking, the individual moving objects are classified in categories, such as heavy and non-heavy vehicles, bicycles, and pedestrians. The resulting trajectories of the moving objects are stored for future processing with engineering applications. The developed signal-processing algorithm is supplemented with a convenient user interface for setting and running and inspecting the results during and after the data collection.

In addition, one engineering application was developed in this study for counting moving objects at intersections. Another existing application, the Surrogate Safety Analysis Model (SSAM), was interfaced with the TScan method to allow extracting traffic conflicts and collisions from the TScan results. A user manual also was developed to explain the operation of the system and the application of the two engineering applications.

Experimentation with the computational load and execution speed of the algorithm implemented on the MATLAB platform indicated that the use of a standard GPU for processing would permit real-time running of the algorithms during data collection. Thus, the post-processing phase of this method is less time consuming and more practical.

Evaluation of the TScan performance was evaluated by comparing to the best available method: video frame-by-frame analysis with human observers. The results comparison included counting moving objects; estimating the positions of the objects, their speed, and direction of travel; and counting interactions between moving objects. The evaluation indicated that the benchmark method measured the vehicle positions and speeds at the accuracy comparable to the TScan performance. It was concluded that the TScan performance is sufficient for measuring traffic volumes, speeds, classifications, and traffic conflicts. The traffic interactions extracted by SSAM required automatic post-processing to eliminate vehicle interactions at too low speed and between pedestrians – events that could not be recognized by SSAM. It should be stressed that this post processing does not require human involvement. Nighttime conditions, light rain, and fog did not reduce the quality of the results.

Several improvements of this new method are recommended and discussed in this report. The recommendations include implementing two TScan units at large intersections and adding the ability to collect traffic signal indications during data collection.

17. Key Words LiDAR, tracking vehicles, tracking pedestrians, traffic conflicts, traffic measurements	**18. Distribution Statement** No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22161.

19. Security Classif. (of this report)	**20. Security Classif. (of this page)**	**21. No. of Pages**	**22. Price**
Unclassified	Unclassified	94	

Form DOT F 1700.7 (8-69)

EXECUTIVE SUMMARY

TSCAN: STATIONARY LIDAR FOR TRAFFIC AND SAFETY STUDIES—OBJECT DETECTION AND TRACKING

Introduction

The ability to accurately measure and cost-effectively collect traffic data at road intersections is needed to improve their safety and operations. This study investigates the feasibility of using laser ranging technology (LiDAR) for this purpose. The proposed technology does not experience some of the problems of the current video-based technology, but less expensive low-end sensors have limited density of points where measurements are collected that may bring new challenges. In this report a novel LiDAR-based portable traffic scanner (TScan) is introduced to detect and track various types of road users (e.g., trucks, cars, pedestrians, and bicycles). The scope of this study included the development of a signal processing algorithm and a user interface, their implementation on a TScan research unit, and evaluation of the unit performance to confirm its practicality for safety and traffic engineering applications.

Work Done and Findings

The TScan research unit was developed by integrating the Velodyne HDL-64E laser scanner within the existing Purdue University Mobile Traffic Laboratory. The motion of the LiDAR sensor located at the top of the mast was accounted for with the readings from an inertial sensor. The primary research objective was to develop an efficient signal processing method to extract the useful traffic information.

The developed TScan method identifies and extracts the background with a method applied in both the spherical and orthogonal coordinates. The moving objects are detected by clustering data points, tracking clusters, and fitting rectangles to the clusters. Detected moving objects are classified as heavy and non-heavy vehicles, bicycles, and pedestrians. The resulting trajectories of the moving objects are stored for future processing with engineering applications. The developed signal-processing algorithm is supplemented with a user interface for setting, running, and inspecting the results during and after data collection.

In addition, one engineering application was developed in this study for counting moving objects at intersections. Another existing application, the Surrogate Safety Analysis Model (SSAM), was interfaced with the TScan method to allow extracting traffic conflicts and collisions from the TScan results. A user manual was developed to explain the operation of the system and the application of the two engineering applications.

The TScan performance was evaluated by comparing to the best available method: video frame-by-frame analysis with human observers. It was concluded that the TScan performance is sufficient for measuring traffic volumes and speeds, classifying moving objects, and counting traffic conflicts. Nighttime conditions, light rain, and fog did not reduce the quality of the results. Several improvements of this new method are recommended and discussed in this report.

Implementation

Experimentation with the computational load and execution speed of the algorithm indicated that the processing during data collection can be executed in real-time. Implementation of the method to practice must be done through one or two prototypes. The report includes the technical and user's specifications of a trailer-based TScan prototype. The report also provides user manuals for setting and operating the TScan research unit and for counting vehicles, pedestrians, and traffic conflicts.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Safety and operations at intersections remain among the most critical areas of road transportation; therefore, accurate and cost-effective collection of traffic data at intersections is important for identification of the causal factors of crashes and evaluation of safety countermeasures, design, and control treatments. While the value of crashes is generally indisputable, there is considerable concern about the downsides of crash data, including low quality, no reliable insight into the crash-preceding events, and long data collection times of as much as several years. These drawbacks limit the usefulness of crash data for both timely acquisition of new knowledge and evaluation of new safety improvement methods.

The need for quick and accurate estimation of safety has been amplified with the growing presence of modern technology on roads and in vehicles that can change road safety. Traditional crash-based methods of acquiring safety knowledge cannot keep up with these changes; and infrastructure and vehicle modifications are not always sufficiently evaluated from the safety standpoint before being implemented. However, there is a silver lining for safety analysis in the emerging technologies because they are capable of collecting more reliable safety-related data that may provide more insight into the risk of crash.

Using near-crash events, or traffic conflicts, is showing potential for improving safety measurement, but despite the considerable efforts undertaken by researchers in the 1970s and 1980s, the expected breakthrough has yet to occur. Currently, the growing need for efficient safety measurement and the opportunities afforded by modern technologies bring new hope to the safety analysis sector again for improving traffic conflicts techniques (Tarko, Davis, Saunier, Sayed, & Washington, 2009).

A new class of relatively inexpensive Light Detection and Ranging (LiDAR) sensors with multiple beams, such as the Velodyne HDL-64E sensor, provides a promising opportunity for developing new traffic measuring techniques that are sufficiently robust to eliminate the involvement of costly human participation in the post-processing phase, which is the main drawback of all the currently-implemented video-based techniques. The hope this time rests in the way in which the LiDAR scanner works. Unlike the existing video-based techniques, the LiDAR measurements are points on the surface of objects surrounding the scanner. These points are represented by three coordinates in the 3D space. This one-to-one mapping between the measurement and the surrounding world allows avoiding the issues created by video-based measurements that are a projection of 3D objects on a plane.

This report presents the results of a feasibility study of Traffic Scanner (TScan), a portable microscopic traffic data-acquisition system that utilizes LiDAR technology. The TScan concept was jointly supported by the Joint Transportation Research Program of the Indiana Department of Transportation and Purdue University (JTRP) and the NEXTRANS Center at Purdue University to enable collecting microscopic traffic data at road intersections. The vision for this proposed system is to overcome some of the limitations of video cameras by producing 3D point clouds that have a one-to-one correspondence with the environment being sensed. This study focused on developing the LiDAR's tracking algorithm and its implementation to determine whether the technology provides a true opportunity to develop a fully automated system for collecting traffic and safety data at intersections. Although this study focused on the fundamental considerations of signal processing for objects classification and tracking, the computational load and processing efficiency also were addressed to increase the practicality of the developed system by significantly reducing the time required for human processing of the data after collection.

The remainder of this report proceeds as follows. Chapter 2 describes the basic underlying ideas of the system design and the primary components of the system and their functional connections. Chapter 3 provides a detailed description of all the components of the algorithm including background elimination, data points clustering into objects, forward tracking, estimation of object dimensions, correcting the trajectories to reduce the effect of occlusion and incorrect initial clustering, additional trajectory smoothing, and object classification. Chapter 4 describes the research unit, which is a van-based mobile traffic laboratory equipped with a telescoping mast, laser scanner, video-cameras, IMU and GPS units, internal communication between the sensors and computers, and data storage units. Chapter 5 briefly presents two applications which utilized the TScan output files: an application for counting objects developed in this project and the existing application developed by Siemens ITS with FHWA support, the Surrogate Safety Analysis Model (SSAM). Chapter 6 presents the performance evaluation of TScan, including the execution time analysis, an alternative human-supported video-based tracking of objects that served as a benchmark method, and the results of a comparison between TScan and the benchmark method. Chapter 7 discusses the knowledge gathered during the project; and Chapter 8 presents the conclusions drawn regarding the system's performance and future research recommendations.

Several appendices provide important information regarding the components: a description of the research unit (Appendix A), the input and output data formats (Appendices B and C), the user manual for setting and operating the TScan research unit (Appendix D), the user manual for the counting objects application (Appendix E), and the technical and user specifications of the trailer-based TScan prototype (Appendix F). The last Appendix G includes recommendations for online Data Portal – a data source being developed by INDOT. It will become a depository of TScan-collected data once the TScan is implemented in Indiana.

2. CONCEPT

This chapter explains the user-oriented concept of the TScan system and the architecture of the research unit with a focus on the data acquisition and processing modules. This concept and the results of the feasibility study presented in this report were used to develop the specifications for the trailer–based prototype included in Appendix F. The computational method, its components, and their interconnection are presented in Chapter 3.

2.1 General Conditions

The proposed measuring system is applicable to road intersections and relatively short road segments in daylight and nighttime conditions; and even during atmospheric precipitation of intensity (light rain, snowfall, and fog) that allows the light beams to travel without extensive dispersion. A single unit is capable of covering a large range of intersections, is easy to set up by a single operator, and is sustainable for several days with limited supervision. The primary source of information for this system is a single LiDAR unit with 64 laser beams that does not require supplementing the data with video images, which speeds up and simplifies setting up the system in the field. The video cameras included in the design are used only for inspecting traffic in short periods as selected by the user after data collection to confirm the validity of the collected numeric data. The outcome of the data pre-processing provides tracking and classification information for every individual object moving in the field view of the system.

The processing and storing of data are executed in real-time, including classification and tracking objects. Specialized processors for matrix operations and parallel computing are utilized to speed up the calculations, which is an important aspect of the system's design and development due to the massive amount of data it may be required to collect during a period as long as one week. Excessive time for post-processing of the collected data would undermine the practicality of the system.

An acceptable level of accuracy of the results, such as vehicle classification, estimation of the path, and the motion speed of all the objects, as well as detection of dangerous interactions between moving objects is of particular importance in this system. The accuracy is sufficient without human involvement and data processing, which is the primary source of savings and the increased practicality of the proposed TScan.

The TScan output file includes all the traffic characteristics of individual vehicles in a convenient format which makes it appropriate for a variety of engineering studies, such as speed studies, counting turning vehicles, gap acceptance studies, measuring saturation flows, and counting traffic conflicts and measuring their severity. For this purpose, the TScan results include the type, dimensions, and positions of the moving objects at 10 instants per second. Furthermore, the format of the outcome of the classification and tracking follows the SSAM format for simulated traffic with modifications that reflect different sources (measurement vs. simulation). For example, the "wire-based" motion of vehicles in typical simulation is replaced with the realistic two-dimensional motion on a (x, y) plane in the real world.

2.2 Hardware Architecture

The trailer-based TScan unit includes four main components (see Figure 2.1):

1. A trailer with a power supply
2. A telescoping mast
3. Sensors installed on a mount
4. A computer with communication component and data storage

A trailer with stabilizing legs allows transportation of the unit to the data collection site, setting up the equipment in a suitable position, and provides sufficient physical support and protection against tempering. The system can generate a continuous power supply with an option of using electrical energy if available at the site. The pneumatic telescoping mast raises the sensors on a mount with a pant/tilt mechanism at a desired elevation. The mast includes an air compressor and a locking mechanism that keeps the mast unfolded without running a compressor.

A low-end LiDAR sensor, such as the HDL-64E by Velodyne Acoustics, Inc., is used to reduce the cost of the unit while providing sufficiently dense and accurate data. A set of inertial measuring units (IMU) are integrated with the primary sensor to adjust the collected data for the sensor motion. A surveillance video camera

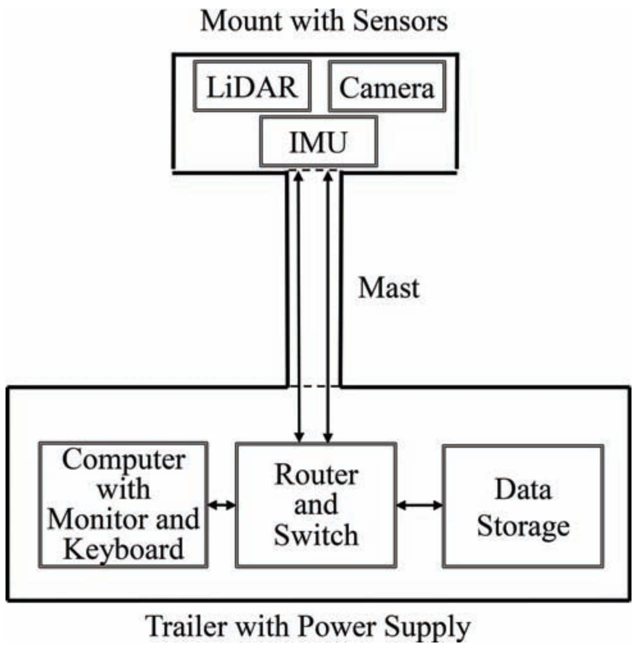


Figure 2.1 A general concept of the portable LiDAR-based system—TScan.

records the traffic flow for later inspection by the user if needed.

The computer with a monitor and a keyboard is used by the operator to supervise setting up the data acquisition component. The computer also facilitates communication between the units, data pre-processing, and data storage during an unsupervised data collection period. The data storage is sufficiently large to allow continuous data collection for at least one week.

2.3 User's Operations and Data Processing

The TScan process is executed in three phases: preparation, real-data processing, and post-processing inspection and engineering applications. The following user and computational tasks are expected during these phases (except one, all the steps are executed with software developed in this study):

1. Preparation
   a. The user prepares an orthogonal image of the intersection with polygons with a TScan application. The intersection should be divided into parts (called polygons), labeled, and classified. This step may be performed in the user's office.
   b. The user aligns the orthogonal image and the TScan frame with TScan software. A snapshot of the TScan data points collected for a short time at the studied intersection must be rotated into an orthogonal position and positioned over an orthogonal image with the supplemented polygons. The two images must be aligned to ensure the same coordinate system of the intersection image and the TScan data.
   c. TScan detects the background during the initial data collection period of several minutes. This step includes the movement stabilization described in the next step. This step produces a set of ground planes estimated for each intersection polygon and are later used to identify object above the plane.
2. Real-time data processing
   a. Movement stabilization. The IMU sensors measure the motion of the LiDAR sensor. The resulted adjustments are continuously used during the data collection to correct the measurements.
   b. Background removal is accomplished by selecting, for further analysis only, points whose positions are sufficiently raised above the estimated ground planes and are moving.
   c. Objects within the polygons are detected by clustering the points that are sufficiently close to each other. These clusters are formed for all points estimated with a single rotation (frame) of the LiDAR sensor.
   d. Objects are tracked between frames by (i) predicting the position of an object in the current frame based on its position in the previous frame and the currently estimated motion, (ii) assigning the TScan-measured cluster in the current frame to the nearest predicted position of the object, and (iii) estimating the new position by combining the predicted and measured positions.
   e. The object dimensions are estimated by overlaying all the clusters by moving them from their estimated positions along the path into a single position. A rectangular box wrapping all the points then is optimized; and its dimensions are assumed to be the dimensions of the actual object.
   f. Object classification is performed based on the object's dimensions and speed and the intersection polygons where the object was detected.
   g. The results include a video and the characteristics of the objects and trajectories, which are stored for later retrieval and processing.
3. Engineering applications
   a. The user may inspect the results during and after data collection with a TScan application. The user displays the objects (rectangles) on the intersection image. Clicking on the object reveals the object's path, speed profile, and acceleration profile together with the recording time. This time can be used to identify the same period and object in the video file for a playback.
   b. Counting turning vehicles is facilitated with an engineering application developed in this project.
   c. Counting traffic conflicts is facilitated with SSAM - an existing public-domain application developed by Siemens ITS.
   d. Other engineering applications can be developed as needed to process the TScan output files.

The next chapter presents the developed TScan algorithm implemented on the research unit.

3. METHOD

This chapter provides a detailed description of all the components of the algorithm including background elimination, data points clustering into objects, forward tracking, estimation of object dimensions, correcting the trajectories to reduce the effect of occlusion and incorrect initial clustering, additional trajectory smoothing, and object classification. This information is provided in a rather informal way to help the user understand the principles upon which the algorithm operates. In justified cases, a more rigorous introduction through mathematical expressions and well-established algorithms of signal processing are provided.

3.1 Setting Up the System

The hardware for data collection resides in a mobile traffic laboratory and consists of a LiDAR HDL64 SE system, three IMUs, and a workstation running the necessary software. Other auxiliary hardware required for collecting data, such as a pan/tilt base, mast, and networking equipment for communication between hardware are discussed in this report.

For TScan to successfully monitor an intersection, a few preparation steps must be performed before the actual data collection begins. Part of this preparation is performed in the user's office (offsite) and the remainder at the intersection (onsite).

During the offsite preparation, the user splits the intersection into approach lanes, intersection exit areas, intersection area, sidewalks, medians, and curb parking lanes by drawing "polygons" that represent the parts of the intersection (see Appendix D). The polygons define the areas that are assumed to be well approximated with planes. These planes, or background planes, are estimated with surface data points that belong to the polygons. Other inputs entered in this phase by the user are later transferred to the corresponding fields in the TScan output files.

The onsite component includes mapping the coordinates of the LiDAR with the Google maps data, selecting the position and orientation of the LiDAR system, and entering the input needed for orientation compensation.

3.2 Background Identification

Once the system is set up, sample data are collected for 15 minutes to identify the background. Two different approaches, depending on the type of polygon, are taken to identify the background. For polygons of a road pavement section that is typically free of vertical obstructions, surface equations are estimated to accurately approximate the background. This approach cannot be used for medians and sidewalks that may contain fixed objects; therefore, a spherical coordinate-based thresholding is applied to those polygons.

3.2.1 Thresholds in Spherical Coordinates

The concept of approximating background using plane equations works for road pavement surfaces it may fail for medians that have fences, poles, plants and other fixed objects. Similarly, sidewalks can also have fixed objects such as the ones mentioned above as well as benches, bicycle racks and other street furniture. These objects belong to the background while planes are not suitable for representing complex surfaces. Hence a different approach is used for these polygons.

The concept of distance-based separation of background and moving objects in spherical coordinates. If one assumes that the LiDAR sensor is not moving (i.e., its motion is negligible or accounted for) then fixed background objects should remain at the same distance from the sensor in all frames. These objects include buildings, road planes, and vehicles parked during the data collection period. Continuous readings in the same direction may be a mixture of measurements of the background and of moving objects if moving objects are expected. It is useful to find a distance threshold separating the background measurements from the moving objects measurements. Moving objects include vehicles, pedestrians, and sometimes trees and other light objects affected by wind.

Let a full rotation of the laser sensor be called the *Frame*. In frame F_1, laser n hits a stationary background object (ground) at horizontal angle α. The distance reported by laser n is D_1. In another frame F_2, laser n hits a moving object when the sensor is again positioned at angle α. The distance reported by the laser is D_2. In the absence of a moving vehicle, the distance reported by the LiDAR would be similar or very close to D_1. Due to the presence a of vehicle, the LiDAR reports a shorter distance. Measurements from certain laser n at certain angle α from a sufficient number of rotations should be considered together to look for a threshold that separates the background from the moving objects.

Algorithm overview

- Collect data for 3000 frames or more, preferably when the traffic is low.
- Distance readings recorded by each laser at each angle are grouped separately.
- For each group, its mean and standard deviation is identified.
- Any point whose distance value is less than the mean and three standard deviations is assumed to belong to a moving object.

3.2.2 Finding Equation of Surfaces

The user provides the coordinates of each polygon in the intersection that is under observation. Polygons may include lanes, sidewalks, medians, crossroads etc. during the setup phase. This information is used in determining the equation of a plane that represents the road pavement.

Algorithm overview

- Convert spherical coordinates to Cartesian coordinates (Section 3.3.1) and at the same time, compensate for the orientation of the sensor (Section 3.3.2). The compensation matrix is pre calculated based on the data from the IMU (which are collected while the initial data are collected) and based on user feedback (Appendix D).
- Remove points that do not belong to any of the polygons.
- Perform triangulation on the resultant point cloud and remove all triangles whose face-normal is beyond a certain threshold. The remaining points belong to road, roofs of vehicles, trees or any other surface that is parallel to the surface of the road.
- In each frame segregate the remaining points based on the background polygon to which it belongs.
- Aggregate the points belonging to the same polygon across all frames.
- For each frame perform plane fitting to obtain a second order polynomial that represents the plane. The plane equation is shown in Equation 3.1.

$$z = a_{11}x^2 + a_{22}y^2 + a_{10}x + a_{01}y + a_{12}xy + a_{00} \qquad (3.1)$$

- After the first initial fit, remove all the points that lie beyond 50 cm from the expected z value in order to remove all points that belong to roofs of vehicles, trees and other objects that are away from the surface.
- Surface fit is then performed again on the remaining points for the polygon and the equations are saved.

3.3 Background Elimination

After the initial setup process and background identification is conducted, the real time data collection and processing module is executed. From this point on, no human involvement is needed for collecting and processing data.

3.3.1 Conversion of Coordinates

The data from the sensor is comprised of three pieces of information:

- the angle about the z-axis of the sensor at which it is oriented
- the distance reading recorded by the laser
- the intensity of return of the laser

These data are of a specific format as described in Section 4.1.2. The data are in spherical coordinates and need to be converted to Cartesian coordinates. Also the data from the LiDAR are segregated into frames, where a *Frame* refers to one rotation of the LiDAR.

Sensor model. The sensor used for the research implementation and recommended for the TScan prototype is the Velodyne HDL 64E sensor, which is equipped with 64 laser diodes. Each of the 64 lasers is individually aimed and, thus, each has a unique set of calibration parameters. An ideal system can be envisioned as follows. The bundle of rays emanating from the 64 lasers lies in a vertical plane and intersects at the origin of the local scanner coordinate frame. The origin of the range measurement for each laser is located at the scanner origin. The manufacturer defines a set of parameters in Table 3.1 for each laser to model the deviations from these ideal conditions (see Table 3.1). Each of these parameters is illustrated in Figure 3.1.

These parameters are determined by the manufacturer and are provided to the end user along with the instructions and sample source code to apply the calibration values to the raw measurements in order to reference the measurements from all the lasers to the local scanner coordinate frame. The computation of the local scanner coordinates (x, y, z) for laser *i* of the Velodyne scanner is given by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (R_i + D_o^i) \times \cos(\delta_i) \times \sin(\varepsilon - \beta_i) - \mathrm{H}_o^i \times \cos(\varepsilon - \beta_i) \\ (R_i + D_o^i) \times \cos(\delta_i) \times \cos(\varepsilon - \beta_i) + \mathrm{H}_o^i \times \sin(\varepsilon - \beta_i) \\ (R_i + D_o^i) \times \sin(\delta_i) + \mathrm{V}_o^i \end{bmatrix} \quad (3.2)$$

Where:

$\mathrm{R_i}$ is the raw distance measurement from laser i;

ε is the encoder angle measurement;

D_0^i, δ_i, β_i, H_0^i are the parameters pertaining to laser i as explained in Table 3.1.

3.3.2 Compensation for Orientation of Sensor

The TScan system has IMUs that report the current orientation of the sensor. The IMUs report three angles, roll (α), pitch (β) and yaw (γ) which represent the rotations about the x, y and z-coordinate axis respectively. If a system is rotated by an angle θ about its x-axis, then the rotation matrix to transform the reference coordinate frame to the new rotated coordinate frame is given by:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \quad (3.3)$$

Similarly, for the y and z axis:

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \quad (3.4)$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3.5)$$

If *xyz* represents a fixed reference frame and *XYZ* represents a frame rotated by *α, β* and *γ* about the x, y

TABLE 3.1
Sensor calibration variables.

Correction Type	Variable	Description
Rotational	β_i	This parameter is the rotational correction angle for each laser, as viewed from the back of the unit. Positive factors rotate to the left, and negative values rotate to the right.
Vertical	δ_i	This parameter is the vertical correction angle for each laser, as viewed from the back of the unit. Positive values have the laser pointing up, and negative values have the laser pointing down.
Distance	D_o^i	Each laser has its own unique distance due to minor variations in the parts used to construct the laser. This correction factor, in centimeters, accounts for this variance. This number should be directly added to the distance value indicated in the packet.
Vertical Offset	V_o^i	This value represents the height of each laser as measured from the bottom of the base. It is a fixed value for all upper block lasers and a different fixed value for all lower block lasers.
Horizontal Offset	H_o^i	This value represents the horizontal offset of each laser as viewed from the back of the laser. It is a constant positive or negative value for all lasers.

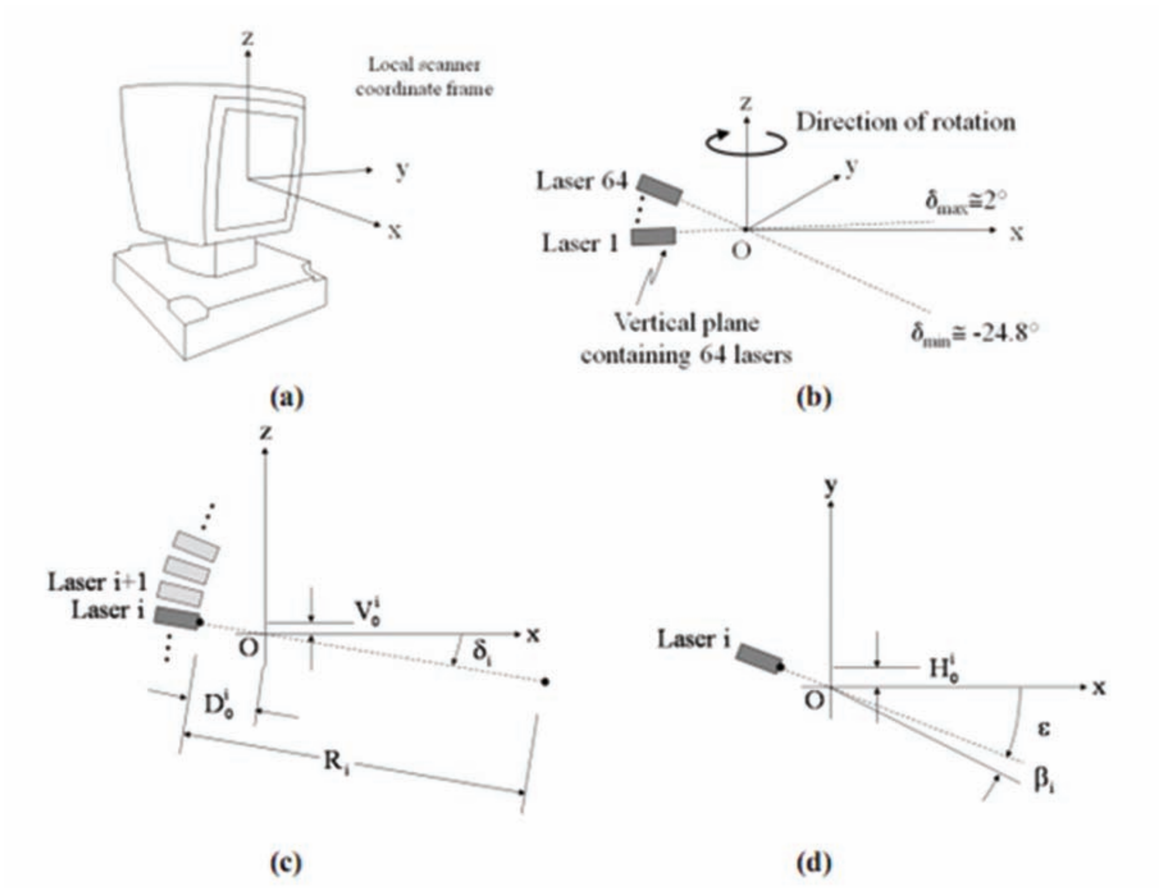


Figure 3.1 (a) Sensor frame axes, (b) sensor layout, (c) scanner parameters in vertical plane, (d) scanner parameters in horizontal plane. (From Glennie, 2010.)

and z-axis of the fixed frame respectively, then the transformation can be represented by Equation 3.7.

$$R_r = R_z(\gamma) \times R_y(\beta) \times R_x(\alpha) \tag{3.6}$$

$$XYZ = xyz \times R_r \tag{3.7}$$

The rotational matrix is orthogonal hence its inverse is its transpose.

$$xyz = XYZ \times R_r{}^T \tag{3.8}$$

3.3.3 Background Elimination

After correcting for the orientation of the sensor, the next step is to remove the background and isolate the points that belong to the objects above the ground. In order to achieve this, the equations of the planes for each of the identified lanes in the previous phase are used.

First, the point cloud that comprises the entire frame is filtered to obtain the points that belong to one polygon. Different background elimination methods are used for polygons that represent road surface and sidewalks.

Spherical coordinates thresholds. For points belonging to polygons other than road pavement, the background cannot be approximated by a single surface equation. Hence for those polygons, spherical coordinates based thresholds are used as explained in Section 3.2.1.

If the reported data point's distance value is greater than a certain cutoff value that is determined by the laser ID and angle of firing, then it is said to belong to the background. Those values that are less than the cutoff value belong to the foreground and are saved for further processing. Refer to Section 4.3.1 for the process of computing the cut-off values and Section 4.4.3 for using those values in removing points that belong to the background.

Equation of surface in XYZ. For the points pertaining to the road pavement, the equation of a surface that was fitted as per the method described in Section 3.2.2 is used. All the points that are less than MAX_THRESH and more than MIN_THRESH above the expected

Z value are the points that belong to the ground (refer to Section 4.3.2 for an explanation of parameters MIN_THRESH and MAX_THRESH). Once the background is eliminated in this fashion, the remaining points are to be further investigated.

This process is repeated for all the frames in the data.

3.4 Clustering Based on Triangulation

After background elimination, the remaining points are grouped into clusters. Points that are close enough to each other are assumed to belong to the same object, and the overlapping bounding boxes of these clusters are assumed to imply that the two clusters belong to the same object. Each sufficiently large cluster then is assumed to represent an object. Points that do not belong to objects but still pass through the background filtering process are considered as "noise." These noisy points can be detrimental as they cause two significant problems:

1. Distortion of the bounding box of a cluster
2. Erroneous clustering of two adjacent objects

Bounding box distortions also lead to erroneous clustering of two adjacent objects. In this process, precautions are taken to reduce the effect of noise. Once the clustering process is finished, an innovative bounding box algorithm is used to find the bounding box of a given point cloud.

3.4.1 Algorithm Overview

This phase consists of the following steps:

- Delaunay triangulation is performed for all the points in the frame post background elimination (Delaunay, 1934).
- The lengths of all the connections obtained from triangulation are computed.
- All connections that are greater than NEIGH_RADIUS are removed. It is assumed that if two points are farther than a certain threshold then those two points belong to different vehicles.
- Those connections that are considered as noise are removed. (Section 3.4.2)
- The remaining connected points are grouped to form clusters.
- A bounding box is computed for each cluster.
- In the case of vehicles similar in size to busses, it is possible that a patch of points is beyond the threshold and yet belongs to the same vehicle. In order to account for this complication, a check is made to see if any two rectangles are intersecting. If they intersect, then they belong to the same vehicle. Hence, the two point clouds are combined and a new bounding box is computed.

3.4.2 Noise Removal

During the clustering process, there may be cases when two vehicles adjacent to each other are clustered together because of (1) noisy points in between the two vehicles that pass through the background filtering process and (2) the noisy points also happen to be close to each other and/or close to clusters representing objects. This means that there is a possibility of two objects (e.g. two vehicles from adjacent lanes) might be clustered together and are given just one ID as illustrated in Figure 3.2.

For each frame, the mean distance between a given point and its neighbors (determined by Delaunay triangulation) is computed along with its standard deviation. All the points that are more than two standard deviations are considered to be noisy connections and are removed.

Noise removal is a conservative process. For the same frame as shown above, once noise removal is applied, the clustering process then identifies them as different objects as shown in Figure 3.3.

One can also note that the vehicles in the coordinates [6000, 1000] in Figure 3.2 are no longer present in Figure 3.3 because the distance between the points among the cluster would have been greater than twice the standard deviation from the mean for that frame. Hence those connections are removed and then the remaining points are less than MIN_PTS_IN_GROUP which represents the number of points that have to be in a cluster for it to be considered an object. Hence the clusters are considered insignificant and are ignored.

3.4.3 minErrorRect

The bounding rectangle that is used in this phase is a modified version of the minimum area bounding rectangle. It is based on the principle that the minimum bounding rectangle has a common edge with the convex hull of the point cloud.

Algorithm

- Compute the convex hull of a given point cloud.
- Calculate the edge angle of each edge of the convex hull.
- Rotate the point cloud such that one of the edges is parallel to the x-axis.
- Compute and total the distance from the parallel edge to all the points in the point cloud, which is the cost of the current rectangle.
- Compute the rectangle using the minimum and maximum X and Y values.
- Rotate the rectangle back to get the coordinates with respect to the input point cloud.
- The rectangle with the minimum cost is the Minimum-Error-Rectangle (minErrorRect).

Comparison with other bounding box methods. Figure 3.4 shows that the box produced by the minErrorRect procedure produces a much better fit to the point cloud than a minimum area rectangle. Since the cost is the Euclidian distance of all the points from one of the edges, this MinErrorRect procedure ensures that an edge of the bounding box is always aligned with the edge with

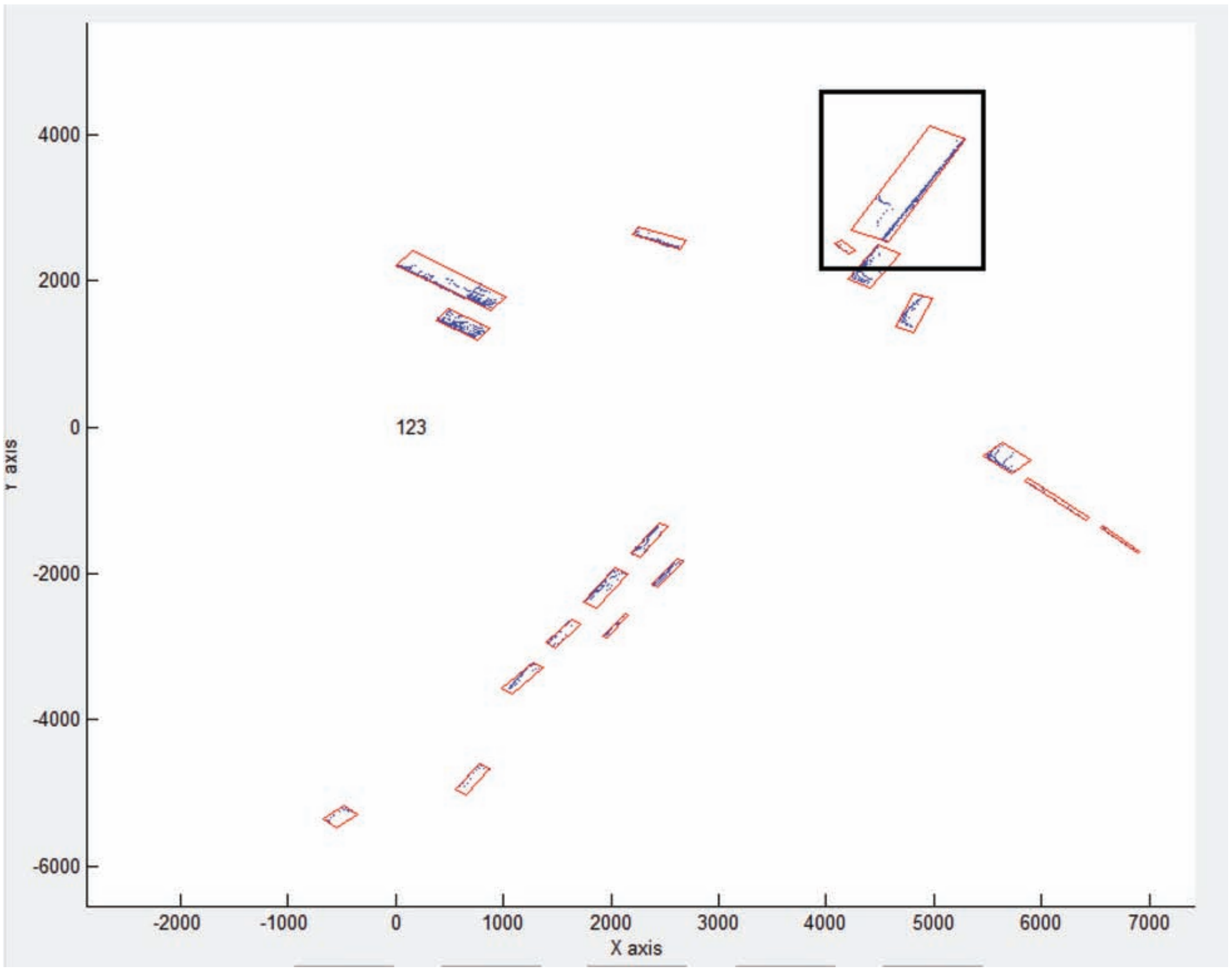


Figure 3.2 Incorrect clustering due to noise.

most of the points in the point cloud. Since predominantly cars, bicyclists, and other road users can be acceptable approximated by a rectangle, this procedure is effective in finding their orientation without knowing their trajectory. The angle that the leading edge (edge with most points near it) makes with the x-axis as reported by this procedure is stored.

3.5 Forward Tracking

Once the point clouds are clustered in each frame, the next step is to the associate point clouds across the frames. Objects are tracked between frames by (i) predicting the position of an object in the current frame based on its position in the previous frame and the currently estimated motion, (ii) assigning the TScan-measured cluster in the current frame to the nearest predicted position of the object and (iii) estimating the new position by combining the predicted and measured positions.

This step is accomplished using a Kalman Filter setup (Kalman, 1960). The motion of vehicles is represented by a constant acceleration model and the centroid of the point cloud is assumed to represent the object as a point mass. Since the dimension of the vehicle is unknown, and the bounding box obtained is different in each frame for the same vehicle, it is unwise to assume the centroid of the bounding box as the point mass that represents the vehicle.

3.5.1 Kalman Filter for Object Tracking

The state vector for the Kalman filter loop is:

$$X_k = (p_x \quad v_x \quad a_x \quad p_y \quad v_y \quad a_y)^T \tag{3.9}$$

Where:

p_x,v_x,a_x represents the position velocity and acceleration along the x axis.

p_y,v_y,a_y represents the position velocity and acceleration along the y axis.

The Kalman filter assumes that the system evolves from time $k-1$ to time k according to the following equation:

$$X_k = F_k X_{k-1} + w_k \tag{3.10}$$

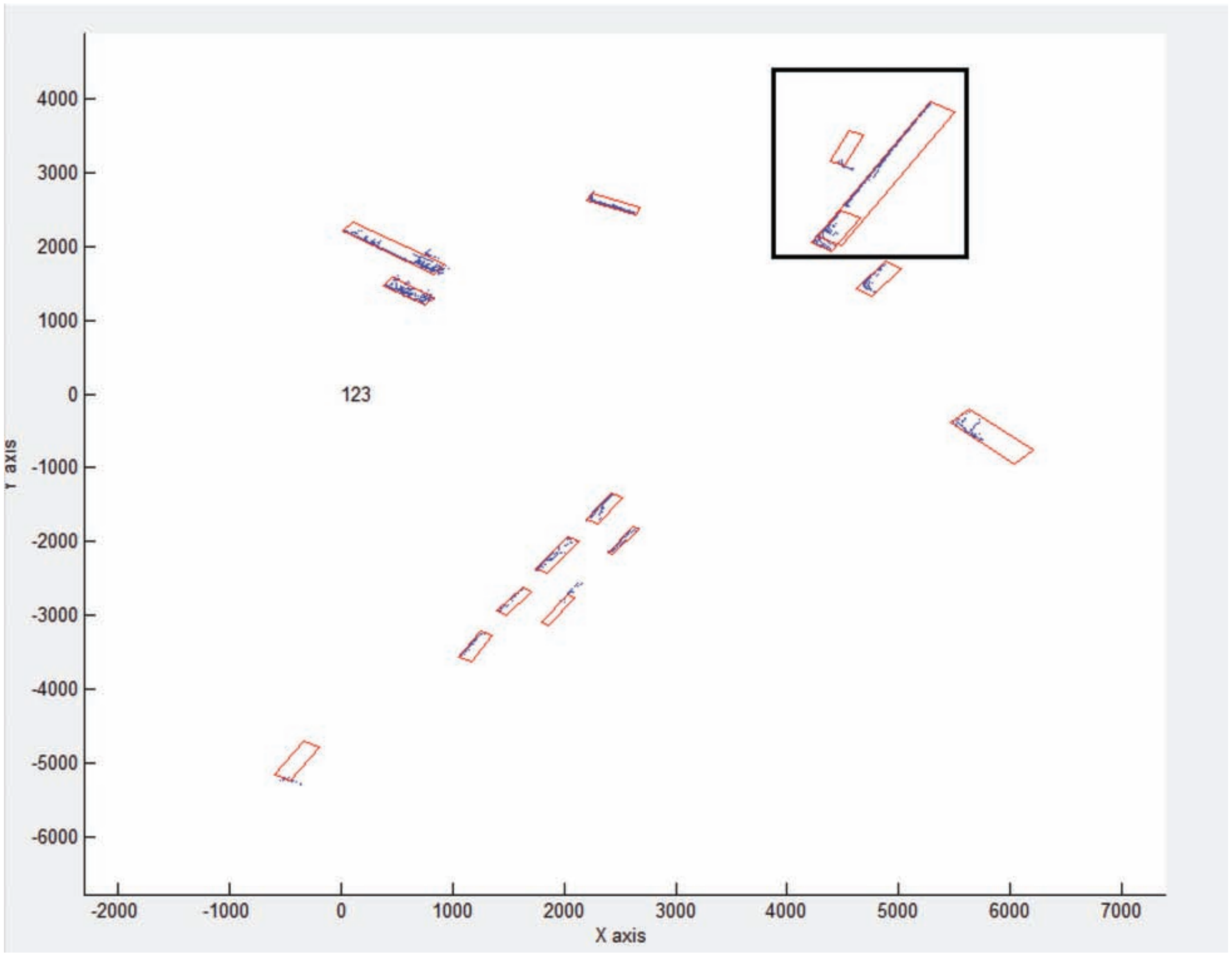


Figure 3.3 Clustering after removing noise.

Where:

F_k represents the state transition model,

w_k represents process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance Q_k.

At time k n observation (or measurement) z_k of the true state x_k is made according to:

$$z_k = H_k X_k + v_k \tag{3.11}$$

Where, H_k represents the measurement matrix which maps the values in the state space to the values in the observed space and v_k presents the observation noise which is assumed to be zero mean Gaussian white noise with covariance R_k. The error covariance matrix is denoted by P_k.

3.5.2 Multiple Object Tracking Using the Kalman Filter

For tracking objects across frames, we use a concept called "tracks." Tracks refer to objects that are currently being tracked. Therefore, in the first frame each object detected in the clustering phase is used to initialize its own track. Tracks represent a concurrent list of objects being tracked. New tracks are added when new objects enter the field of view of the sensor and existing tracks are removed if the objects have not been visible to the LiDAR for a continuous number of frames.

Detection of vehicles. The first step in multiple object tracking process is to detect vehicles in each frame. This is achieved in the previous phase of clustering (Section 3.4). The results of clustering are directly used in this step.

Prediction. The second step is to predict the location of the vehicles in the current frame using the Kalman filter and the following equations:

$$X_{k|k-1} = F_k X_{k-1|k-1} \tag{3.12}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{3.13}$$

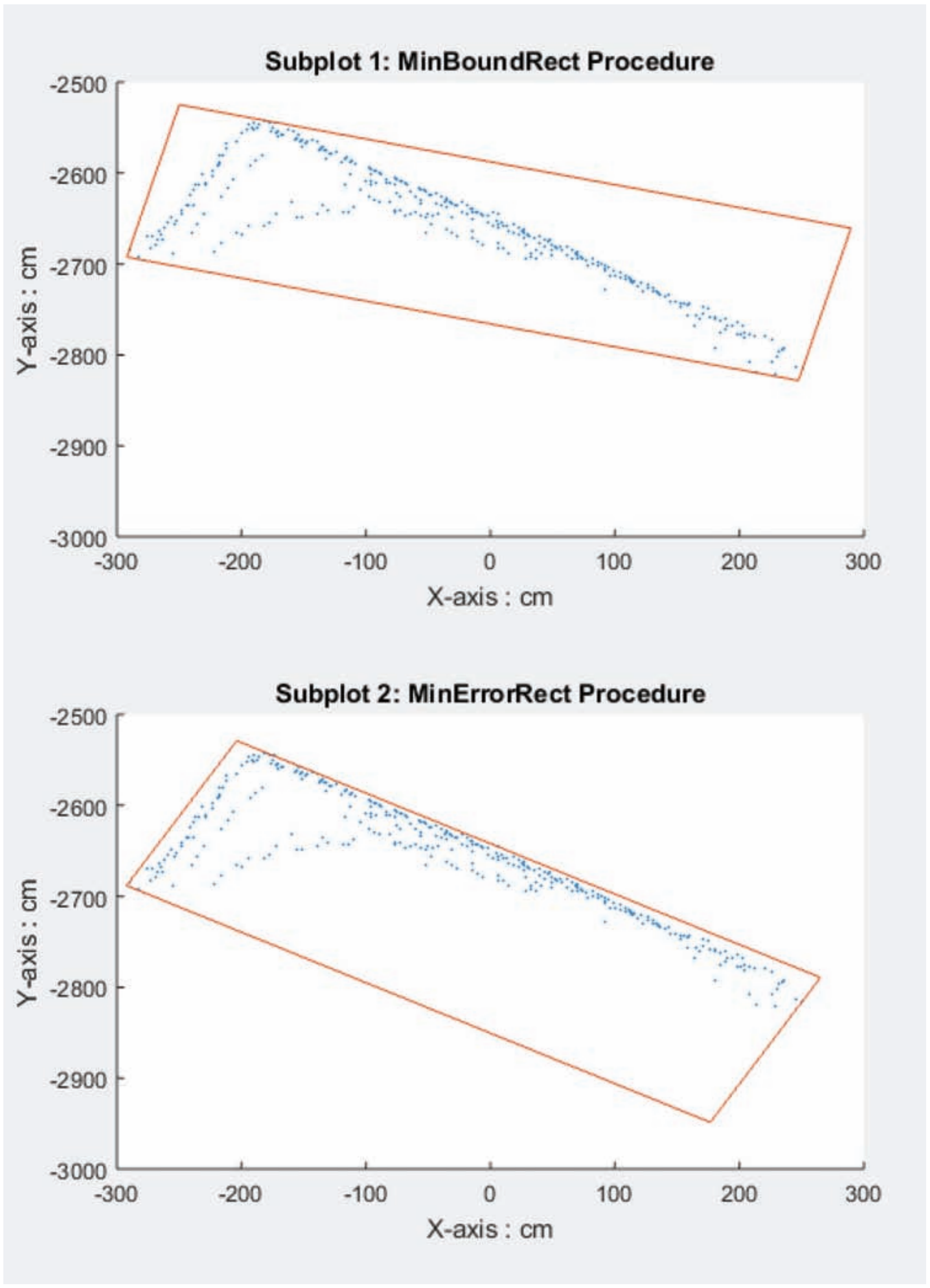


Figure 3.4 Comparison of minimum area rectangle (minBoundRect procedure) and minimum error rectangle (minErrorRect).

Assign detection to tracks. The next step is to compute the Euclidian distance between the predicted centroids (the existing tracks) and the detected centroids (the centroids of the point clouds present in the current frame). This will result in $p \times d$ matrix where p is the number of predictions for the current frame and d is the number of detections in the current frame. Each object currently being monitored is called a track. The Hungarian assignment algorithm (Harold, 1955) is then used to optimally assign the detections to tracks.

Correction. The assignment algorithm assigns detections to most of the tracks, which means that these tracks have detections in the current frame. These detections serve as the measurements that are fed to the Kalman filter routine. The best estimate of the current state is then calculated using the following equations:

$$\tilde{y}_k = z_k - H_k X_{k|k-1} \tag{3.14}$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{3.15}$$

$$K_k = P_{k|k-1} H_k^T + R_k \tag{3.16}$$

$$X_{k|k} = X_{k|k-1} + K_k \tilde{y}_k \tag{3.17}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{3.18}$$

Modify tracks. For tracks to which detections are not assigned, the number of frames since its last detection was assigned is updated.

If the track is invisible for too long (i.e., no detections have been associated with that track for a predefined number of consecutive frames), then it is assumed that the object has moved away from the field of view of the sensor and therefore is removed from the list of tracks.

Detections that are not associated with any existing tracks are assumed to represent new objects and new tracks therefore are created for them.

Each identified object has a record which contains the entire history of the object. As the tracks are updated, so are their records. Tracks only contain the most recent information regarding an object, whereas records contain their entire history. Any record that has persisted less than a second is considered unreliable and the possibility that it represents a physical moving object lessens. Hence, it is removed from the list. Identified objects are given unique IDs, called the cluster ID, which are stored in their records.

3.6 Dimension Estimation and Box Placement

Since the directions of motion of objects are unknown, the Kalman filter is initialized with zero velocity and zero acceleration. In reality, vehicles enter the field of view of the LiDAR with non-zero velocity and acceleration. In order to estimate the position and velocity of the object in the initial frames more accurately, the Kalman smoothing algorithm is used. Note that the estimated velocity and acceleration are that of the centroid of the point cloud and not that of the vehicle. The smoothed estimates of the centroids of the point clouds of a vehicle across frames better represent the overall trajectory (position and orientation) of the vehicle than the Kalman estimates.

3.6.1 Modified Bryson-Frazier Smoother

For Kalman smoothing, the Modified Bryson-Frazier (MBF) (Bierman, 1977) smoother is used over the Rauch-Tung-Striebel (RTS) algorithm (Rauch, Tung, & Striebel, 1965). The advantage of the MBF algorithm is that the inverse of the covariance matrix need not be computed. It uses a backward pass that processes data saved from the Kalman filter forward pass. The equations or the backward pass involve the recursive computation of data which are used at each observation time to compute the smoothed state and covariance.

The recursive equations are:

$$\tilde{\Lambda}_k = H_k^T S_k^{-1} H_k + \hat{C}_k^T \hat{\Lambda}_k \hat{C}_k \tag{3.19}$$

$$\hat{\Lambda}_{k-1} = F_k^T \hat{\Lambda}_k F_k \tag{3.20}$$

$$\hat{\Lambda}_n = 0 \tag{3.21}$$

$$\tilde{\lambda}_k = -H_k^T S_k^{-1} y_k + \hat{C}_k^T \hat{\lambda}_k \tag{3.22}$$

$$\hat{\lambda}_{k-1} = F_k^T \tilde{\lambda}_k \tag{3.23}$$

$$\hat{\lambda}_n = 0 \tag{3.24}$$

Where S_k is the residual covariance and $\hat{C}_k = I - K_k H_k$. The smoothed state and covariance can be found by substitution in these equations:

$$P_{k|n} = P_{k|k-1} - P_{k|k-1} \tilde{\Lambda}_k P_{k|k-1} \tag{3.25}$$

$$x_{k|n} = x_{k|k-1} - P_{k|k-1} \tilde{\lambda}_k \tag{3.26}$$

3.6.2 Orientation of Vehicle from Centroid Trajectory

The smoothed out centroids from the MBF smoother are used to calculate the orientation of the vehicle at each time step. The orientation of the vehicle at time step k is given by the angle that the vector connecting the centroids at time steps $k - 1$ and k makes with the x-axis. When the vehicle is stationary, the centroid of the vehicle still appears to move based on the distribution of points in the point cloud. In order to correct for this, only the centroids from the frames in which the vehicle has a velocity greater than a certain threshold are used to calculate instantaneous orientation.

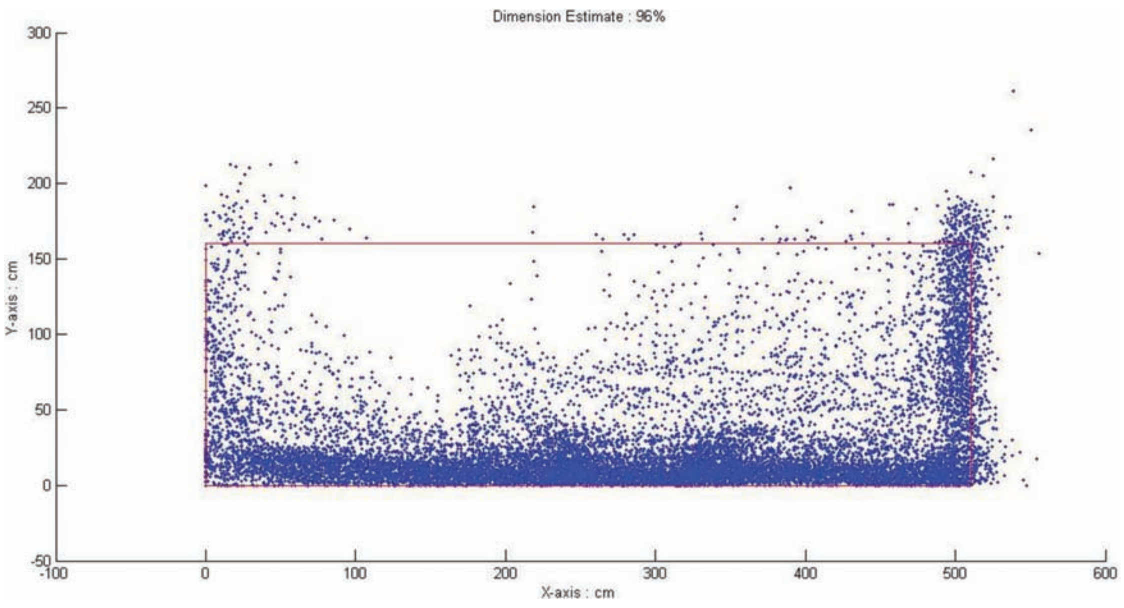


Figure 3.5 Dimension estimation.

3.6.3 Dimension Estimate Using edgeAngles from minErrorRect

In order to best estimate the orientation of the vehicle, all the point clouds that represent the vehicle across the frames need to be used. This is achieved by the following steps:

- Rotate the vehicle by the negative edgeAngle, which is the angle that is obtained from the minErrorRect procedure. The resultant point clouds have the leading edge of the vehicle parallel to the x-axis.
- Compensate for the position of the vehicle by subtracting the coordinates of the smallest vertex of the bounding box.
- Now a new bounding box that envelopes a certain percentage of points given by the parameter PERCENT_BOUND (Section 4.7) is computed. This bounding box is assumed to represent the dimension of the vehicle.

Figure 3.5 shows the estimated dimension box (in red) along with the point clouds of the vehicle collected over all the frames it was present in compensated for orientation (blue dots).

3.6.4 Placement of the Estimated Box on the Point Cloud

Once the dimension of the underlying vehicle, which is represented by a series of point clouds over time, is estimated, a box with that dimension must be placed back on top of the point cloud.

Note that the centroid of the point cloud is not representative of the centroid of the vehicle because the point cloud represents different parts of the vehicle at different points in time in its trajectory based on its orientation. The most reliable feature of the vehicle that is represented in the point cloud is the nearest corner of the vehicle; however, the nearest corner visible to the LiDAR changes along the trajectory of the vehicle. A mechanism to overcome this issue was devised. The principle driving this method is that by property, LiDAR returns points from the nearest surface of any object (only the parts that are "visible"). The following steps are used to place the box on top of the point cloud:

- Rotate a box of size of the estimated dimensions of the object by its instantaneous angle of motion.
- One of the four corners of the point cloud must align with the four corners of the box of the estimated dimensions.
- A cost metric is computed for each of the four configurations. The cost metric is defined as the product of the sum of the Euclidian distances from the origin to the four corners of a given configuration and percentage of the points in the point cloud it encloses.
- The configuration with the maximum cost is the desired box placement.

Figure 3.6 illustrates the four possible box placements for a sample point cloud. Configuration d (Figure 3.6d) is the best representation and the chosen one.

3.7 Refining Cluster IDs

When there is a shadow in the scene, a vehicle passing through the shadow is split into two pieces. The clustering process will consider both pieces as belonging to the same vehicle only if the size of the shadow is small, such that the length of the connection is not considered as noise. When the size of the shadow is big enough, the two pieces of the vehicle are assigned different IDs. Since the intent is to associate each vehicle to just one ID, these instances where the break up happens must be investigated and the pieces that belong to the same vehicle need to be combined and its properties need to be re-estimated.

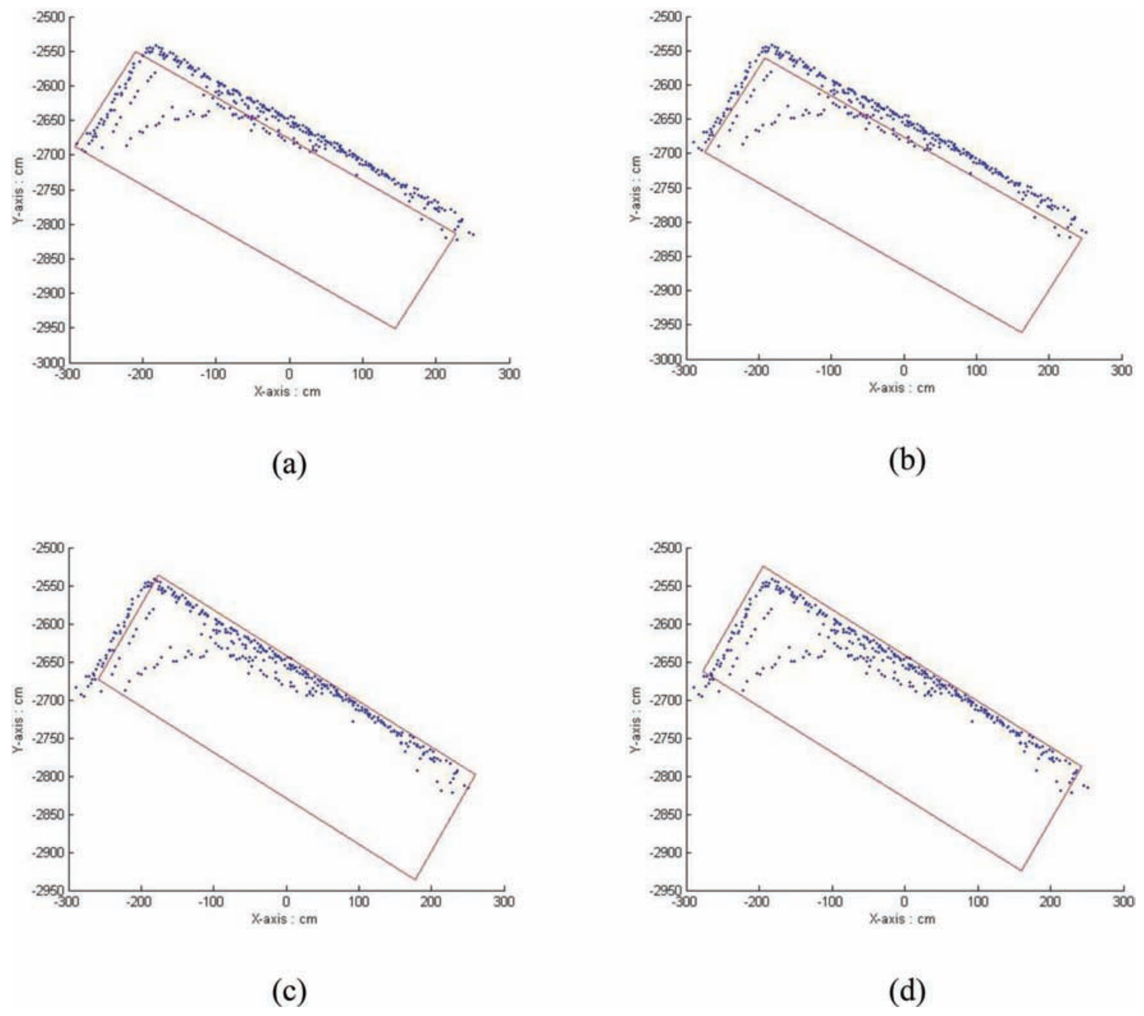


Figure 3.6 Four configurations during box placement.

There also is a possibility of wrong association of vehicles due to the shadows cast by moving vehicles. In this case, the ID of one vehicle is wrongly associated with another vehicle in mid- trajectory. Essentially, what this means is that a vehicle ID may contain one part of the trajectory from vehicle A and another part from a nearby vehicle B. In this case, both IDs need to be analyzed and then separated.

The process of splitting/merging is an iterative one. Convergence is achieved when there are no more objects left to split and/or merge.

3.7.1 Merging IDs That Belong to the Same Vehicle

Need for merging cluster trajectories. When there is a shadow in the scene cast by a pole, a vehicle passing through the shadow will be represented by two sets of dense points with a region of empty space (shadow) between them. The clustering process considers both these pieces as belonging to the same object only if the size of the shadow is small such that the length of the connection at the end of triangulation is not considered as noise (refer to Section 3.4 for clustering process). When the size of the shadow is big, the two sets are considered as two different clusters representing two vehicles and hence are assigned different IDs. Since the goal is to associate each vehicle with one ID, these instances where break ups happen must be investigated and pieces (clusters) that represent the same vehicle need to be combined and its properties re-estimated. When the underlying vehicles represented in the data can be correlated to cluster IDs generated by the algorithm on a one to one basis, further analysis, such as backtracking, is more reliable and accurate.

The principle guiding this procedure is based on the reasoning that two boxes representing two vehicles overlap only as the result of an uncommon collision thus the overlapping case is most likely an indication of a single vehicle.

Due to the nature of LiDAR and the presence of shadows in the data, cluster splits result in multiple boxes around each piece and these boxes tend to overlap. Also due to the nature of LiDAR, a true collision/near miss means that the TScan program will report it as a temporary increase in the size of one of the vehicles. Hence the supposed overlaps of boxes need to be investigated further.

Identifying candidates for merging. Two clusters are good candidates for merging if the two bounding boxes of these clusters overlap at some instances. This overlapping indicates that the two clusters might represent one vehicle whose corresponding cluster was split due to shadows or missing measurements.

Thus the following procedure is used to identify intersecting boxes:

1. After estimating the dimension of each identified vehicle in the dataset, the estimated boxes are then placed back on top of the point cloud.
2. In each frame, it is checked whether or not any two boxes representing a vehicle overlap. If they overlap, then the two IDs of the objects are noted.
3. A table is then developed, which represents the combination of vehicle IDs that have overlaps and also the frames in which they overlap.

Analysis of candidates. Once the lists of candidates have been identified, each pair must be evaluated on an individual basis to check whether or not they can be merged together. Two tests were devised for this purpose.

Distance between centroids. This test analyzes the distance between the centroids of the two point clouds in question when they are present in the same frame. The hypothesis is that two clusters with considerably varying distance between their centroids cannot represent the same object. Thus checking for relative distance and its variance between the centroids of two candidate clusters provides an indication of whether or not they belong to the same physical object.

For instance, in Figure 3.7, two objects exist, whose bounding boxes (which represent the estimated dimension of the point cloud over its entire trajectory) overlap within the area highlighted by a black box. Those two objects are the front "tractor" unit and a trailer which belong to the same semi-trailer type vehicle.

Change in dimension. The second test is to analyze the change in dimension when the two clusters are combined. The hypothesis is that the combined cluster's dimension would be at best the same as one of the two clusters and at worst its length would be the sum of the lengths of the two individual clusters.

First, the frames in which both the clusters are present are identified. Then for each frame, the point clouds that represent these clusters are combined. The tightest bounding box that contains both point clouds is then estimated (minErrorRect procedure explained in Section 3.4.3). This estimation procedure also computes the angle that the leading edge makes with the x-axis of the coordinate frame. This angle indirectly represents the orientation of the combined object. Finally all the point clouds across the frames are overlapped accounting for their orientation and position in space in order to estimate the dimension of this combined object.

If the combined dimension is within a certain threshold (which is a function of lengths of the individual cluster) then the two objects are combined.

3.7.2 Splitting Up Wrong Associations

Need for breaking up trajectories. When there is a shadow in the scene, a vehicle passing through the shadow is split into two clusters. These two clusters are given different IDs. Shadows can be cast by large moving vehicles too.

Possibility of incorrect association. Consider a scene involving two vehicles (A, B) that are in close proximity to each other. These vehicles are represented by clusters M and N. When object A enters a shadow, its corresponding cluster M is also occluded. Meanwhile, vehicle B is partially occluded by a pole (or tree trunk). The corresponding cluster N is split into two (N_1 and N_2). There are instances where cluster N_1 is mistaken to represent vehicle A and is assigned the ID of cluster M. Cluster N_2 retains the ID of cluster N but once cluster N_2 completely disappears, the particular ID associated with it ceases to exist. We end up with an ID whose first part belongs to vehicle A and the second part belongs to vehicle B. Figure 3.8 shows an example of one such case.

In order to account for the possibility of such events, the trajectories of each of the cluster IDs must be analyzed. When the actual vehicles in the data can be correlated to the cluster IDs generated by the algorithm on a one to one basis then further analysis like backtracking is much more reliable and accurate.

Analysis of trajectories to identify incorrect associations. The following steps are performed to analyze a single cluster ID:

1. Fit a polynomial equation (up to fourth order) that best represents the trajectory of the cluster. Here "best" implies the least Root Mean Square Error (RMSE). Every cluster whose trajectory has an RMSE value of greater than MAX_RMSE is assumed to have a likelihood of false association and is considered for further analysis.
2. Identify regions where there is a break in the trajectory. A break in trajectory is characterized by two features.
    a. A set of frames in which the object is "invisible." From analyzing data, it is evident that incorrect associations can occur if the object is missing / not enough points are returned from the object for a few frames and new candidates are close by.
    b. An incorrect association due to the splitting of clusters. This and any other incorrect association without any missing frames for the cluster is characterized by a sudden jump in acceleration.
3. A polynomial equation is then fit (as mentioned in step 1) to the trajectory of the cluster until the first break (frame wise, indirectly time wise).
4. If the RMS value of the fit is greater than 25, then the piece of the trajectory might have to split off from the rest. This break is noted down.
5. If not, then repeat step 3 by considering up to the next identified break.
6. Steps 3, 4 and 5 are repeated again but this time, in the opposite direction of time.

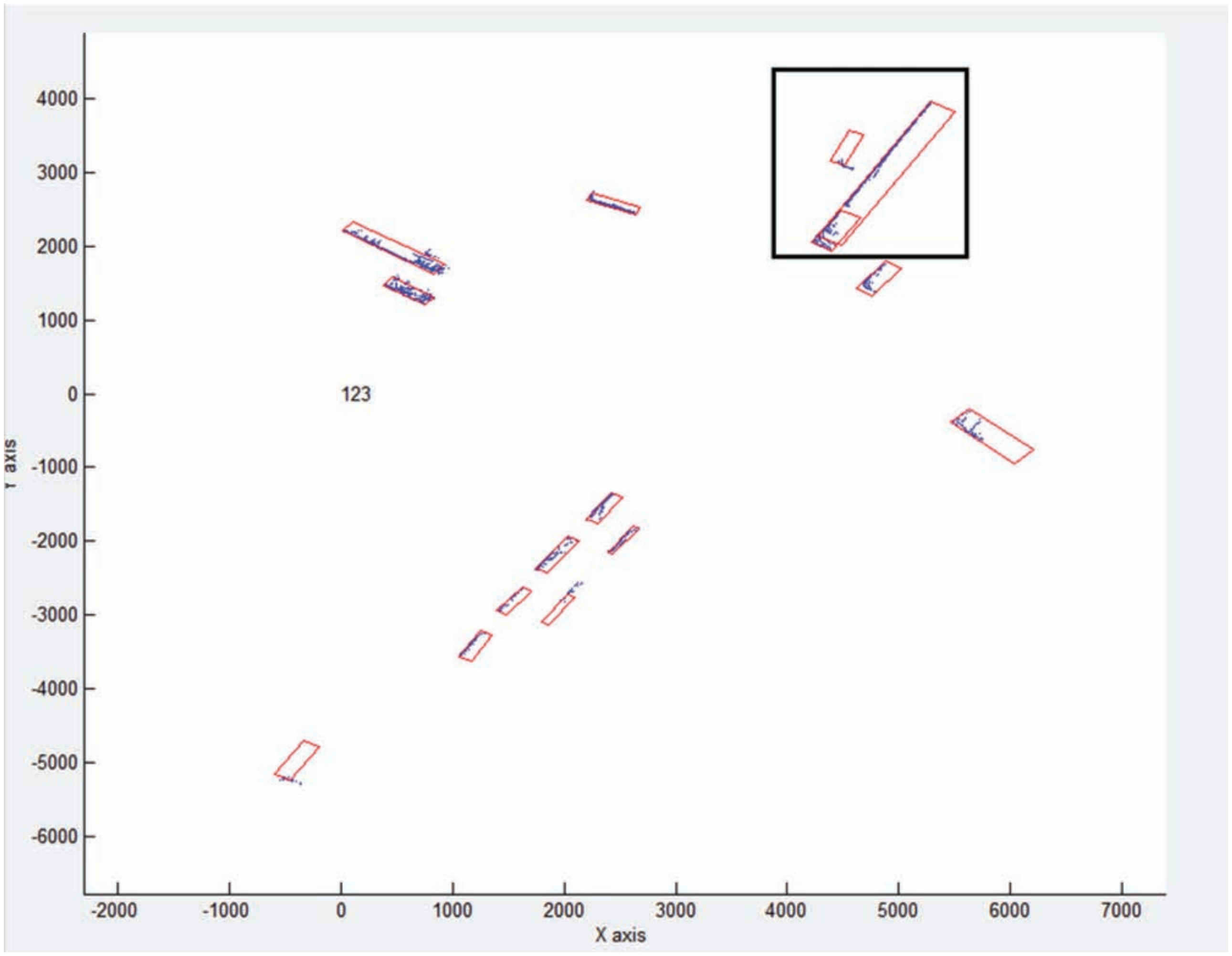


Figure 3.7 Example of overlapping boxes representing same physical object.

7. The trajectory is then broken at the points in time where both the forward and reverse iterative method suggested the same break point.

Note that a single cluster trajectory might have to be broken at multiple points in time. The algorithm assumes that such a case is possible.

3.8 Re-Compute Dimensions and Final Smoothing

After the trajectories are split and joined as explained in Section 3.7, it is believed that most of the clusters have one to one correspondence with actual vehicles and that each individual trajectory is representative of the complete motion of the vehicle within the LiDAR's field of view.

3.8.1 Re-Compute Dimension

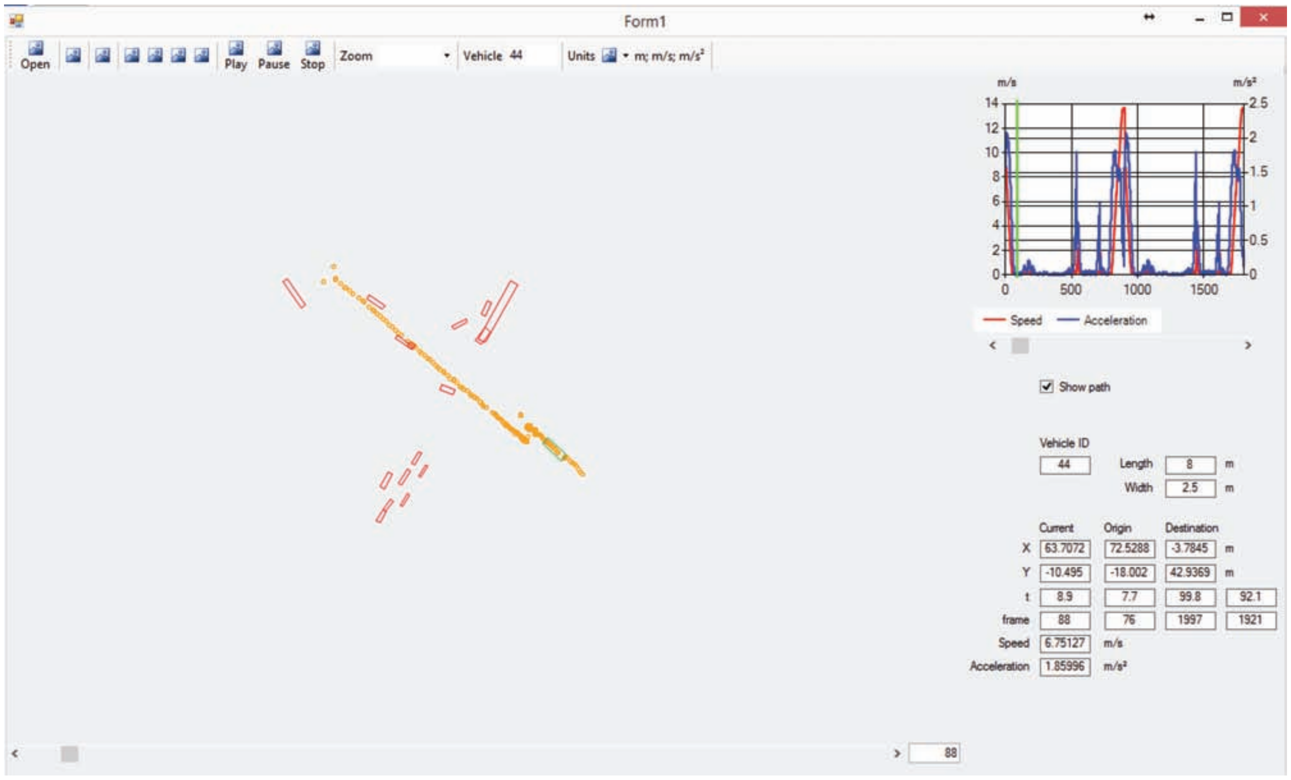
First, the Kalman filter with the modified Bryson Frazier smoother (explained in Section 3.6.1) is used to obtain a smoothed trajectory of the centroids of the point clouds for each ID. From the smoothed centroids, instantaneous angle of orientation is obtained (explained in Section 3.6.2). Using the orientation information, the box is placed as shown in Section 3.6.4.

3.8.2 Final Smoothing

Since the angle of motion estimation is based on the estimated point cloud centroids it does not accurately represent the vehicle. Once the newly estimated box is placed back on top of the point cloud, the centroid of the boxes is known. This centroid represents the vehicle more accurately. A local linear second order regression based smoothing is used to smooth out the centroids. The boxes are then placed based on the smoothed centroids. Thus the final trajectory estimate of the vehicle is obtained.

3.9 Classification of Objects

TScan produces the characteristics of the detected moving objects that are helpful in the classification of these objects. The most promising information for classification objects are speed, dimensions, and parts of intersection (polygons) where the objects were detected. This section describes the method of object classification in this study.



Vehicle 44

Units m; m/s; m/s²

☑ Show path

Vehicle ID	44		
	Length	8	m
	Width	2.5	m

	Current	Origin	Destination		
X	63.7072	72.5288	-3.7845	m	
Y	-10.495	-18.002	42.9369	m	
t	8.9	7.7	99.8	92.1	
frame	88	76	1997	1921	
Speed	6.75127	m/s			
Acceleration	1.85996	m/s²			

88

Figure 3.8 Incorrect association example; orange dots represent position of centroid of bounding box over the trajectory.

There were several candidate methods that might be useful for the classification task: discrete outcome models, discriminant analysis, and decision/regression trees. A discrete outcome model was selected, namely, a multinomial logit (MNL) model. A multinomial logit model was estimated with the data collected by TScan (speeds, dimensions, and polygons) and the data collected by a human observer (object type: pedestrian, bicycle, heavy vehicle, non-heavy vehicle. The probability of an object type i was evaluated with the multinomial logit model Equation 3.27:

$$P(i) = \frac{\exp(\beta_i + \beta_{i1} X_1 + \ldots + \beta_{ik} X_k)}{\sum_{j=1}^{J} \exp(\beta_j + \beta_{j1} X_1 + \ldots + \beta_{jk} X_k)} \tag{3.27}$$

Where (i) denotes the probability of an object being of category i among I, X were the object's attributes, and β_i were the model parameters for category i. One of the categories was selected as a reference with all the β values set at 0. The following model variables were tested for inclusion in the model; the first and last polygons where an object was detected; the 75th, 85th and 95th speed percentiles; and the object's width and length. The speed percentile was preferred over the average speed to reduce the effect of stops on the speed attribute.

Difficulties with the convergence of the estimates led to splitting the sample objects into two groups based on polygons. Otherwise, a quasi-complete separation of the data points detected with the statistical software SAS would prevent the estimation process from converging. Some of the binary variables that represented the first and last polygons were found to correctly allocate most of the observations into subsets of the response groups. This difficulty revealed that if a sidewalk or median was a starting or ending polygon, the object was almost always a pedestrian or bicycle. Thus, the dataset was split into two subsamples:

(1) Objects starting or ending in a sidewalk or median polygon,
(2) Objects starting and ending in a polygon of another type.

The first subsample included mainly pedestrians or bicycles. Estimating the MNL model for this group failed again for the same problem as for the entire sample. A quasi-complete separation of the data points when the 75th speed percentile was included in the model. Thus, the 75th percentile of the speed was used with a threshold of 3.51 m/s to separate pedestrians from bicycles.

The second subsample included all possible types of objects except bicycles. The MNL model was estimated for this subsample. The non-heavy vehicles category was used as a reference. The most useful and significant variable for distinguishing between objects turned out to be the length of the object. The final model is shown in Table 3.2. All the signs were in line with the expectations. In general, a length increase reduced the probability of the object being a pedestrian (negative beta)

TABLE 3.2
Multinomial logit model objects belonging to the second subsample.

Variable	Parameter Estimate	Std. Error	Wald Chi-Square	p-value
Pedestrian				
Constant (Pedestrian)	11.427	5.320	4.614	0.0317
Length of object (cm)	-0.081	0.040	4.019	0.0450
Heavy Vehicles				
Constant (Heavy vehicles)	-6.762	0.866	61.010	0.0001
Length of object (cm)	6.1×10^{-3}	1.1×10^{-3}	30.582	0.0001
Non-Heavy Vehicles				
Constant (Non-heavy vehicles)	0	–	–	–
AIC at convergence = 76.330		AIC (constants only) = 300.143		
SC at convergence = 91.132		SC (constants only) = 307.544		

and increased the probability of the object being a heavy vehicle (positive beta).

A complete procedure includes two binary splits of the objects based on the first/last polygon and the 75th speed percentile, and the maximum probability calculated with the MNL model:

```
IF Starting Polygon = Sidewalk or Median OR Ending
   Polygon = Sidewalk or Median
THEN    IF 75th Speed Percentile > 3.51 m/s
        THEN Object = Bicycle
        ELSE Object = Pedestrian
ELSE MNL Model
Calculate Probability of Pedestrian
Calculate Probability of Non-Heavy Vehicle
Calculate Probability of Heavy Vehicle
Object = Object with Highest Probability
```

3.10 Output Files

Once the final smoothing is completed, the results are presented in two formats SSAM format presented in Appendix B and a TScan format described in Appendix C. These files act as inputs for various engineering applications such as counting vehicles and SSAM conflict analysis to name a few. Figure 3.8 shows a screenshot of the application developed as part of this project for viewing trajectories.

4. RESEARCH IMPLEMENTATION

The TScan concept described in Chapter 2 and the method described in Chapter 3 were implemented in the research unit.

Two major components were implemented for data collection as shown in Figure 4.1. Each of the components was further divided.

1. The System Preparation by User
   a. Off-site location data preparation
   b. On-site sensor alignment
2. Data Collection Module
   a. Background Identification
   b. Real time data collection

User involvement ends with the background identification phase. Once the background is identified, the user is notified that the system is ready for background removal. The user then can start the actual data collection and processing. During this process, user intervention is not required. The real time data collection and processing procedures are depicted in Figure 4.2.

For the research unit, all the algorithms were prototyped in MATLAB (Mathworks, n.d.). Using MATLAB allowed modification of the algorithm and visualization of the results much faster than implementing it in a lower level language like C++. Since the current setup is geared toward rapid prototyping and emphasizes lessening the time taken to implement the changes, the real time module was not a part of the TScan software and additional software was used to collect the data. The Data Collection and Processing module provided the functionality for the additional software. The final version of the system will be using the TScan software alone.

The algorithms were designed to achieve modularity and parallelism wherever possible. The final production code were written in C++ and integrated with the TScan software (Appendix D).

The current method in the research unit requires additional software to collect data. The data were collected first using the DSR viewer for LiDAR (from Velodyne) and Sensor Explorer for the IMUs (from Vectornav). Then, the algorithms were run offline after data collection to produce the necessary output. For the prototype unit, the TScan software was used, which can collect data and process it in real time.

4.1 Research Unit Hardware

The Purdue MTL (Mobile Traffic Laboratory) is built based on a Chevy Express 3500 van and equipped with a 42-foot pneumatic mast that can be operated from inside the van. It includes two PTZ IP dome cameras and two flat screen monitors, a computer, an eight-channel video recorder, and a one-gigabit Ethernet network. The equipment is powered by a heavy-duty inverter and almost all the equipment is rack-mountable for safe transportation.

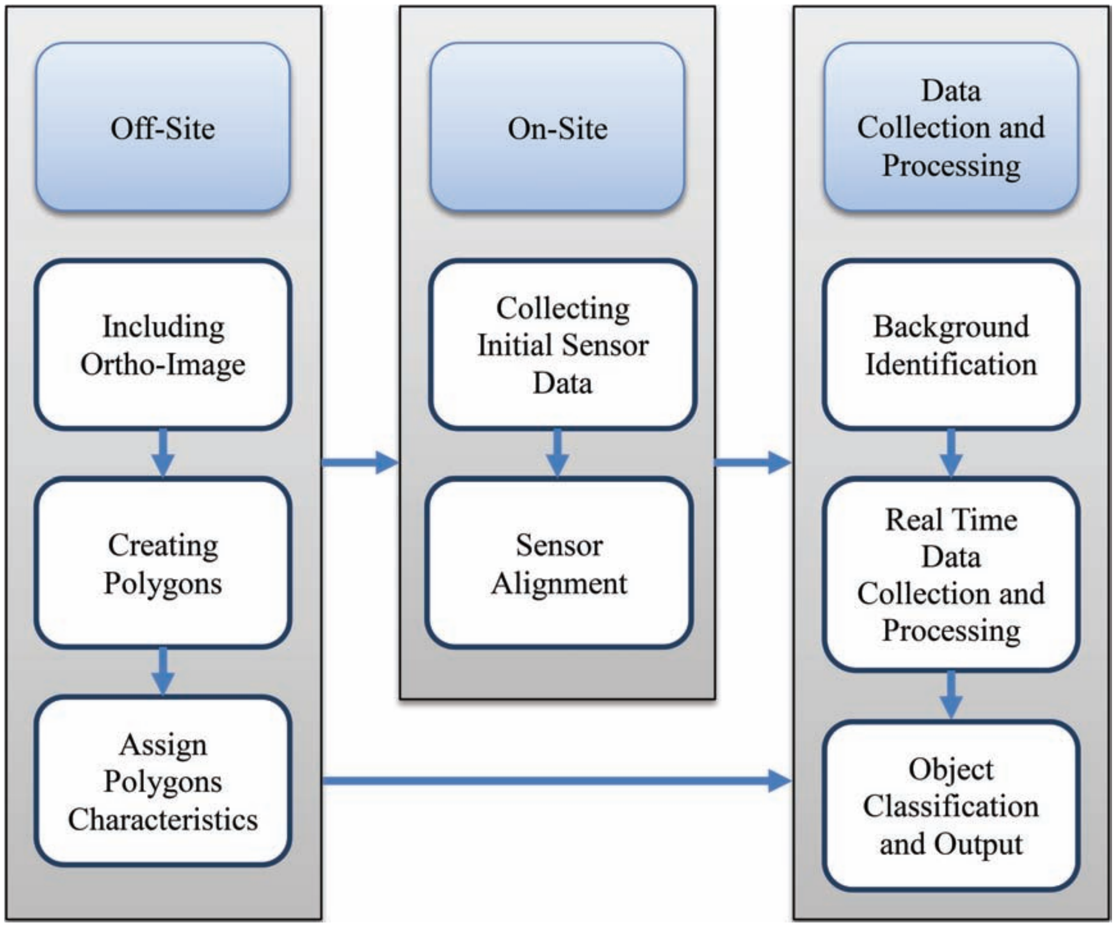

Off-Site
Including Ortho-Image
Creating Polygons
Assign Polygons Characteristics
On-Site
Collecting Initial Sensor Data
Sensor Alignment
Data Collection and Processing
Background Identification
Real Time Data Collection and Processing
Object Classification and Output

Figure 4.1 TScan data collection process overview

For TScan, LiDAR three-dimensional laser scanning technology was integrated into the MTL with three IMUs to track the orientation of the LiDAR in real time. The viewing position of the LiDAR can be further adjusted with a pan and tilt base controlled from inside the van. For a complete specification list of the research unit, refer to Appendix A.

4.1.1 LiDAR Unit

The Velodyne HDL-64E LiDAR sensor that was used in the prototype unit includes a compact sensor pod with 64 laser line scanners. The line scanners produce 64 laser beams arranged vertically inside a 27^o vertical angle. The unit rotates to give a full 360^o by 26.8^o FOV. The Velodyne sensor has been used in autonomous vehicle applications, such as the DARPA Grand Challenge (Cheung, 2007). The specifications for the Velodyne scanner are shown in Table 4.1 (Velodyne, n.d.b).

The HDL-64E spins at rates ranging from 300 RPM (5 Hz) to 900 RPM (15 Hz). The default is 600 RPM (10 Hz). Changing the spin rate does not change the data rate. The unit sends out the same number of packets (at a rate of 1.3 million data points per second) regardless of the spin rate. The image resolution will increase or decrease depending on the rotation speed.

4.1.2 Data Structure of LiDAR Unit

The HDL-64E data are presented as distances and intensities only. The connection between the LiDAR and the computer is similar to a two-way LAN setup. The LiDAR constantly sends messages with the fixed IP source and destination addresses. The data collected are packaged in a format called *.pcap*.

The HDL-64E outputs UDP Ethernet packets. Each packet contains a data payload of 1206 bytes that consists of 12 blocks of 100-byte firing data followed by six bytes at the end of each packet that contain the spin counter and firmware version information. Each packet can be either for the 32 upper or the 32 lower laser banks (called laser blocks).

The packet format is as follows:

2 bytes of header information

This header indicates whether the packet is for the upper block or the lower block. The upper block has a header of 0xEEFF and the lower block has a header of 0xDDFF.*

*The hex values shown in the packages are in inverted orders. Therefore, the upper block indicator EE FF is shown as FF EE in any text editor, which is the case for all the other values.

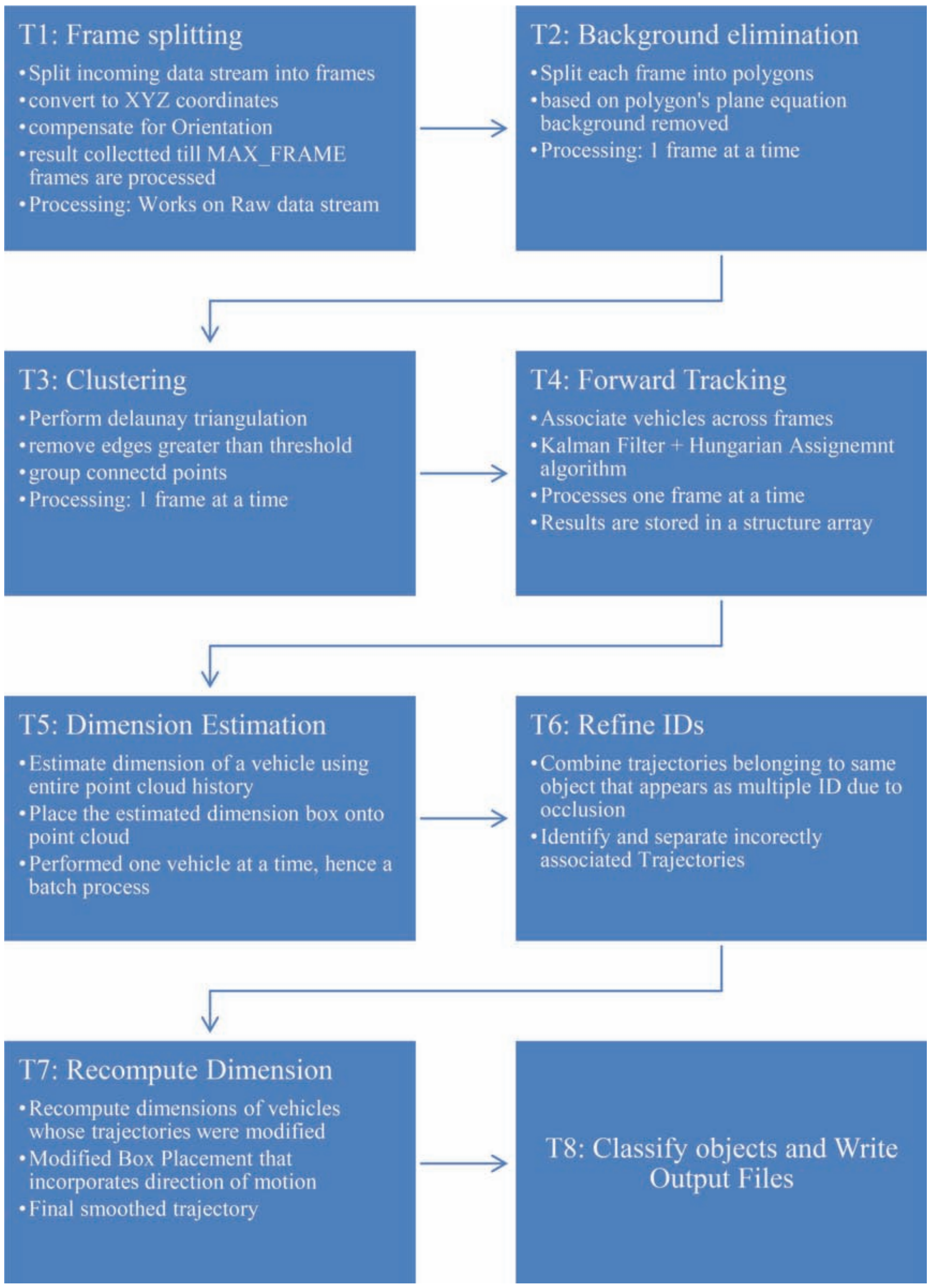

T1: Frame splitting
•Split incoming data stream into frames
•convert to XYZ coordinates
•compensate for Orientation
•result collectted till MAX_FRAME frames are processed
•Processing: Works on Raw data stream
T2: Background elimination
•Split each frame into polygons
•based on polygon's plane equation background removed
•Processing: 1 frame at a time
T3: Clustering
•Perform delaunay triangulation
•remove edges greater than threshold
•group connectd points
•Processing: 1 frame at a time
T4: Forward Tracking
•Associate vehicles across frames
•Kalman Filter + Hungarian Assignemnt algorithm
•Processes one frame at a time
•Results are stored in a structure array
T5: Dimension Estimation
•Estimate dimension of a vehicle using entire point cloud history
•Place the estimated dimension box onto point cloud
•Performed one vehicle at a time, hence a batch process
T6: Refine IDs
•Combine trajectories belonging to same object that appears as multiple ID due to occlusion
•Identify and separate incorectly associated Trajectories
T7: Recompute Dimension
•Recompute dimensions of vehicles whose trajectories were modified
•Modified Box Placement that incorporates direction of motion
•Final smoothed trajectory
T8: Classify objects and Write Output Files

Figure 4.2 Schema for real time processing after initial setup and identification of background.

2 bytes of rotational information

This is an integer between 0 and 35999; dividing this number by 100 produces values in degrees.

32 laser return info of 3 bytes each

Each return contains:

2 bytes of distance information, in .2 centimeter increments.

1 byte of intensity information shown as 0 - 255, with 255 being the most intense return.

A zero return indicates no return up to 65 meters.

TABLE 4.1
Sensor manufacturer specifications for the HDL-64E S2 scanner.

Sensors	64 lasers
	360° (horizontal) by 26.8° (vertical) FOV
	Range: 50 m (10% reflectivity) 120 m (80%)
	1.5 cm range accuracy (1 sigma)
	0.09° Horizontal Encoder Resolution
	>1.33 million points per second
Lasers	Class 1
	905 nm wavelength
	5 nanosecond pulse
	2.0 mrad beam divergence

TABLE 4.2
Sensor manufacturer specifications for the VN-100T IMUs.

Manufacturer and Model	Vectornav VN-100T
Sensors	3 axis accelerometer
	3-axis gyroscopes
	3-axis magnetometer
	1 barometric pressure sensor
Communications	Serial RS-232 & TTL
Angular resolution	<0.05 deg
Output rate	800 Hz

Note: For detailed specification of other hardware being used, refer to Appendix A.

6 status bytes

These status bytes alternate between packets. The end of the packet will show one of the below options:

- A reading showing the internal temperature of the unit in the form, of a "DegC" ASCII string as the last four bytes of the packet. The two bytes before this string are the thermistor's reading in C in hex 8.8 format, which is in the "big endian format" (i.e., the byte immediately preceding the DegC text is the whole degrees, and the byte preceding that is the fraction of a degree in 1/256 increments; for example, c0 1a indicates that the temperature of the thermistor is 26.75 degrees C.
- The version number of the firmware in ASCII character format "Vn.n" where n.n is the version number (e.g., "1.5").

In summary, the total bytes per packet of data is: $1206 = 12 \times (2+2+32 \times (2+1)) + 6$

An image depicting the data structure of a packet is shown in Figure 4.3 for reference. Note that the distance reading is zero, which means that the particular laser fired at that angle never returned, either because it was reflected away or there was no object to reflect off of within the range of the sensor.

4.1.3 IMU

The current research unit has a system of 3 IMUs to accurately measure the orientation of the sensor. The sensor used is VectorNav's VN-100T. The specifications are shown in Table 4.2.

4.2 System Preparation by User

From a user perspective, the TScan process has two distinct phases, an offsite process and an onsite process. These two parts help in setting up the system for the actual data collection.

4.2.1 Off-Site Process

The off-site process allows the user to enter, save and retrieve the intersections characteristics in a graphic environment to transfer the information to TScan. An orthographic-image is used as the base reference to identify areas with similar characteristics. The user draws polylines to create uniform polygons. Characteristics are then assigned to the created polygons such as: polygon type, potential users, type of maneuver, etc. Once the information is complete, a set of files is created in order to transfer the information to the TScan unit.

4.2.2 On-Site Process

The on-site process aims to set a common coordinate system for the TScan sensor and the orthographic-image in order to transfer the polygon characteristics to the TScan processing module. A typical workflow executed by the user on-site can be summarized as follows:

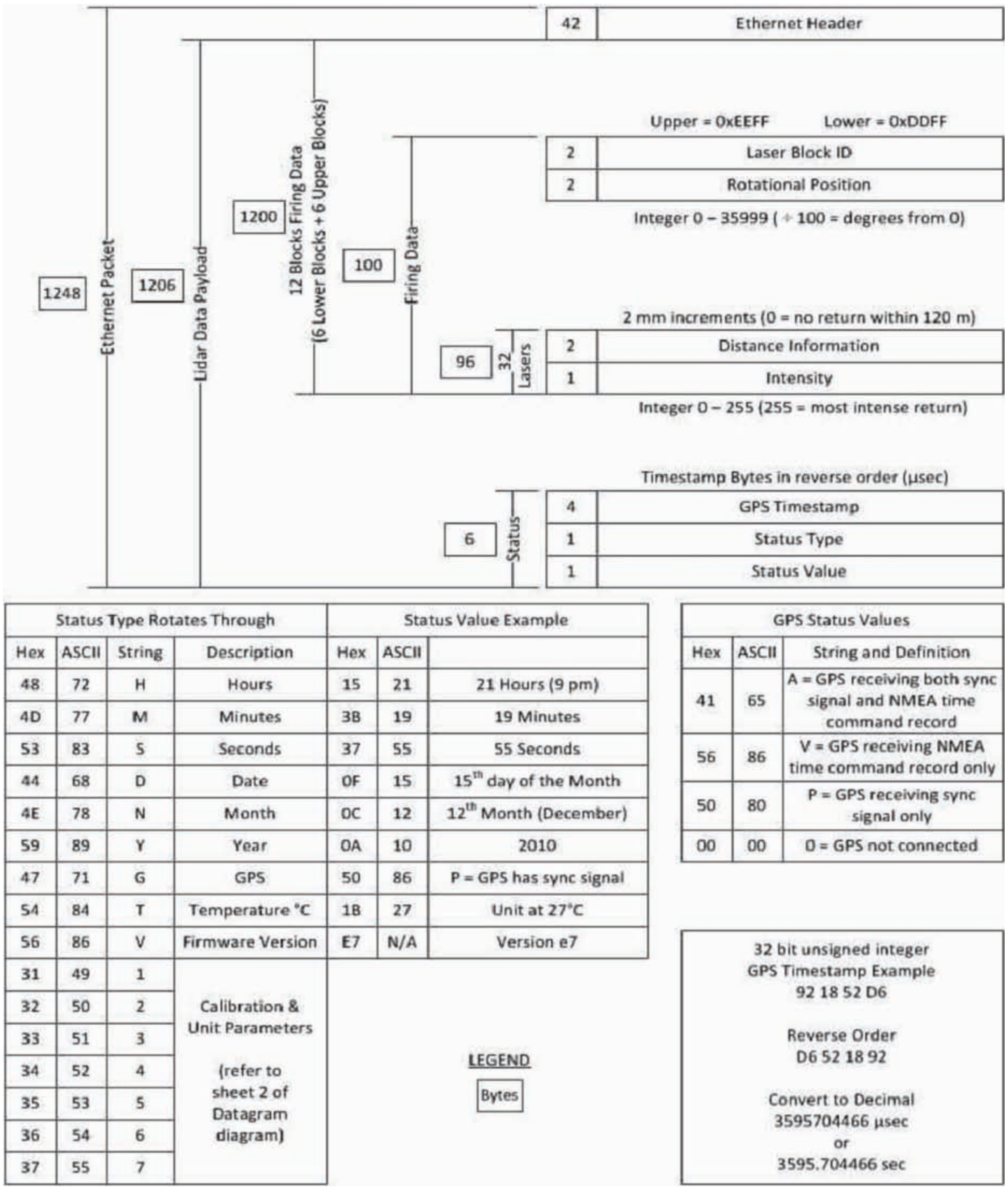
- Information created off-site is retrieved and visualized.
- Initial sensor data are collected.
- Sensor data is visualized side by side with the orthographic-image.
- Two specific features are then selected that are present in both the orthographic-image and the LiDAR sensor data.
- The application's built-in alignment function is used to properly align the two.
- The alignment is further adjusted manually.
- The final alignment information is saved and exported to the Data Collection and Processing Module.

4.3 Background Identification

Once the initial setup information is confirmed, the next step is to identify the background. The background in polygons belonging to the road pavement is approximated by a surface equation in Cartesian coordinates. For polygons other than road pavement, it maynot be possible to characterize the background with a single surface equation because the background might contain poles, plants, road signs, buildings, and other objects. For this reason, spherical coordinate-based background removal is used. The concept is explained in Section 3.2.

4.3.1 Thresholds in Spherical Coordinates

The distance information from the LiDAR must be grouped based on the laser from which the data were obtained as well as the angle of the LiDAR at the time the data is recorded. From Table 4.1 it can be seen that


42
Ethernet Header
1248
Ethernet Packet
1206
Lidar Data Payload
1200
12 Blocks Firing Data (6 Lower Blocks + 6 Upper Blocks)
100
Firing Data
Upper = 0xEEFF Lower = 0xDDFF
2
Laser Block ID
2
Rotational Position
Integer 0 – 35999 (÷ 100 = degrees from 0)
2 mm increments (0 = no return within 120 m)
96
32 Lasers
2
Distance Information
1
Intensity
Integer 0 – 255 (255 = most intense return)
Timestamp Bytes in reverse order (μsec)
6
Status
4
GPS Timestamp
1
Status Type
1
Status Value
LEGEND
Bytes

Status Type Rotates Through				Status Value Example		
Hex	ASCII	String	Description	Hex	ASCII	
48	72	H	Hours	15	21	21 Hours (9 pm)
4D	77	M	Minutes	3B	19	19 Minutes
53	83	S	Seconds	37	55	55 Seconds
44	68	D	Date	0F	15	15th day of the Month
4E	78	N	Month	0C	12	12th Month (December)
59	89	Y	Year	0A	10	2010
47	71	G	GPS	50	86	P = GPS has sync signal
54	84	T	Temperature °C	1B	27	Unit at 27°C
56	86	V	Firmware Version	E7	N/A	Version e7
31	49	1	Calibration & Unit Parameters (refer to sheet 2 of Datagram diagram)			
32	50	2				
33	51	3				
34	52	4				
35	53	5				
36	54	6				
37	55	7				

GPS Status Values		
Hex	ASCII	String and Definition
41	65	A = GPS receiving both sync signal and NMEA time command record
56	86	V = GPS receiving NMEA time command record only
50	80	P = GPS receiving sync signal only
00	00	0 = GPS not connected

32 bit unsigned integer
GPS Timestamp Example
92 18 52 D6

Reverse Order
D6 52 18 92

Convert to Decimal
3595704466 μsec
or
3595.704466 sec

Figure 4.3 User datagram protocol (UDP) Ethernet packet format: HLD-64E.

the data can be split into 64×4000 groups, where 64 represents the number of lasers and 4000 represents the number of possible angles. The encoder of the LiDAR has a resolution of 0.09 degrees; hence there are 4000 different angle readings.

Once the data are grouped, the mean and standard deviation for each of the groups are computed and stored in 64×4000 arrays. Groups having points less than 1% of the number of frames are ignored as the sample size is too small. Next, a minimum standard deviation of 10 cm is assumed because anything less than that is within LiDAR's error range.

$$\mu_{i,\alpha} = \frac{\sum_{n=1}^{n=k} D_{i,\alpha,n}}{b} \tag{4.1}$$

Equation 4.1 refers to computing the mean of each group, where:

$\mu_{i,a}$ is the mean value of a group of readings from laser i fired at angle α

$D_{i,a,n}$ is the distance reading given by laser when fired at angle α in n frame

k is the total number of frames in the batch

b is the number of non-zero distance readings in the batch because a zero distance value means a null or no return.

$$\sigma_{i,\alpha} = \sqrt{\frac{\sum_{n=1}^{n=k} (D_{i,\alpha,n} - \mu_{i,\alpha})^2}{b}} \qquad (4.2)$$

Equation 4.2 refers to computing the standard of each group, where:

$\sigma_{i,a}$ is the standard deviation value of a group of readings from laser i fired at angle α.

The cutoff value $c_{i,\alpha}$ is then given by:

$$c_{i,\alpha} = \mu_{i,\alpha} - m\sigma_{i,\alpha} \qquad (4.3)$$

Where m is a real positive value, from experimental data, setting $m=3$ produces the best results.

4.3.2 Finding Equation of Surfaces

The LiDAR sensor packages the measurements in a specific format as explained in Section 4.1.2. These measurements are in spherical coordinates and have to be converted into XYZ. The conversion process is explained in Section 3.3.1. At the same time, the readings are compensated for the orientation of the sensor as explained in Section 3.3.2. This compensation allows to the assumption that after compensation the ground plane (predominantly the road pavement) is parallel to the XY plane and only has gradients defined by the terrain/road design.

Points that don not belong to any of the polygons of interest are then removed from the data. Delaunay triangulation is then performed on the points belonging to each frame. In each frame, the triangles whose face-normal are not parallel to the z-axis are removed. This assumption holds true due to the compensation to sensor orientation performed earlier. After this step only triangles (and as an extension, the points in the triangle) that are parallel to the ground surface are left.

The remaining points in each frame are then divided into groups based on the polygon in which they are present. Then a surface equation is fit to all the points belonging to a particular polygon. The surface equation is of the form:

$$z = a_{11}x^2 + a_{22}y^2 + a_{10}x + a_{01}y + a_{12}xy + a_{00} \qquad (4.4)$$

Where:

x, y and z represent the coordinate values

a_{ii} represents the coefficients

QR decomposition is used for the fitting process as it provides the best balance between numerical stability and speed.

4.4. Background Elimination

Section 3.3 explains the background elimination procedure. The steps performed in this phase are:

1. Process incoming data and split into frames
2. Compensate for orientation
3. Convert coordinates from spherical to Cartesian
4. Perform background elimination (XYZ and spherical)

4.4.1 Data Processing

The LiDAR information is stored in a specific format as explained in Section 4.1.2. The LiDAR generates a continuous stream of data which has to be broken down into frames. A frame refers to the data collected during one rotation of the LiDAR. This process of breaking down into frames is thus inherently dependent on the following:

- Nature of the sensor
  - Number of lasers
  - Angular resolution of the encoder
- Structure of incoming data

If any of the above changes (due to firmware changes or choosing a different sensor for the prototype) then this part of the algorithm needs to be rewritten accordingly.

4.4.2 Coordinate Conversion and Orientation Compensation

Once the incoming data are split into frames, the data are converted to Cartesian coordinates as explained in Section 3.3.1. Then the data are compensated for the orientation of the sensor as explained in Section 3.3.2.

The current prototype consists of a system of IMUs to track the orientation of the sensor in real time. The sensor has a sampling rate of 100 Hz. The angles reported by the individual IMUs are used to directly compensate for the orientation of the sensor.

IMU data are not being used for real time motion compensation because proprietary software was chosen, which does not allow storage of the synchronization information as needed. The TScan software (Appendix D) overcomes this issue.

After compensating for orientation, the next step is to remove the background from the data by extracting only the points belonging to moving objects. There are two different methods used for this depending on the polygon to which the point belongs as explained in the following two sections.

4.4.3 Spherical Coordinates Based Elimination

For polygons other than road pavement, as explained in Section 3.2.1, spherical coordinates based thresholds are used. First points belonging to these polygons are collected together. Any point that satisfies the following condition is assumed to belong to a moving object.

$$D_{i,\alpha,n} < \mu_{i,\alpha} + 3 \times \sigma_{i,\alpha} \tag{4.5}$$

Where, $D_{i,a,n}$ reporesents the distance value reported by laser i at angle α for frame n, $\mu_{i,a}$ and $\sigma_{i,a}$ represent the mean distance and standard deviation for laser i at angle α respectively. Note that if the lasers have no return (i.e there was no object within the range of the laser), then by virtue of the sensor, it returns a distance of zero. One must ensure that these points are ignored. Three times the standard deviation is used because of the assumption that the majority of the points for any given laser and angle group belong to the background and only the extreme outliers belong to the moving object. Also, of interest here are outliers that report shorter distances because the distance reported will always be smaller when there is an interruption when compared to the normal path of hitting a stationary background.

4.4.4 XYZ-Based Background Elimination

For the polygons that encompass the road pavement, the following method of background elimination is used.

First for a given frame, the points that belong to a particular polygon are segregated. The coordinates of the points are then substituted in the surface equation of that polygon. The surface equation of the polygon is of the form shown in Equation 4.4. The Z value obtained from the equation is referred to as the expected z-value. If the Z coordinate of a point is less than MAX_THRESH and more than MIN_THRESH than the expected Z value, then that point is said to belong to the foreground. These points are saved for further processing.

The process explained in the above two sections (Sections 4.4.3 and 4.4.4) are repeated for each frame in the dataset.

4.4.5 Parameters

Parameters dependent on sensor:

- STEP_ANGLE: one step in the angular encoder of the sensor
- N_LASERS: number of lasers in the sensor
- FPS: number of frames per second

Parameters independent of sensor:

- MAX_FRAMES: number of frames in dataset, if not set, the program reads the entire .pcap file
- MIN_THRESH: the minimum height from the expected Z value above which any point is considered to be part of an object of interest needs further investigation
- MAX_THRESH: the maximum height from the expected value below which any point is considered to be part of an object of interest needs further investigation

4.5 Clustering Based on Triangulation

Once the background is eliminated, the points belonging to a single object must be grouped together. Section 3.4 explains this procedure in detail. There are two main parameters in this phase that influence the outcome of clustering.

NEIGH_RADIUS

If two points are separated by a distance greater than this value, then the points are considered to not belong to the same physical object. The smaller the value, the more conservative the clustering is and may result in a single vehicle being considered as two. For instance, the roof is considered to be separate from the sides. If the value is larger, two adjacent vehicles might be assumed to be the same vehicle. Currently a value 50 cm is used which works well in conjunction with noise removal process (Section 3.4.2).

MIN_PTS_IN_GROUP

MIN_PTS_IN_GROUP represents the number of points that have to be in a cluster for it to be considered an object. The farther the object form the LiDAR the lesser the number of points incident on the object. Hence if this parameter is high we may fail to track the object when it reaches the edges. From the data we found that in order to reliably identify an object, a cluster must have 15 points in it.

4.6 Forward Tracking

A Kalman filter setup is used for tracking objects as explained in Section 3.5. The Kalman filter has some variables that have to be initialized, and a discussion of that is provided in this section. This section also discusses about choice of parameters like nature of point to be used for the point mass model of Kalman Filter, cost function for association and cost of non-assignment in Hungarian assignment algorithm.

4.6.1 Kalman Filter Initialization

Initial state error covariance matrix. Since the first measurement gives us a good estimate of the position but we lack information regarding velocity and acceleration, the state error covariance matrix P_k is initialized to:

$$P_{0|0} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 300 & 0 & 0 & 0 & 0 \\ 0 & 0 & 300 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 300 & 0 \\ 0 & 0 & 0 & 0 & 0 & 300 \end{bmatrix} \quad (4.6)$$

The initial values, shown in Equation 4.6, are based on the covariance matrix after tracking for a sample of 100 vehicles.

Process noise covariance Q. Using Trial and error, the process noise covariance matrix was fixed at:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.5 \end{bmatrix} \quad (4.7)$$

Measurement noise covariance R. The LiDAR has an error of up to 15 cm (at a range of 70 m). The error is less when the object is closer. Experiments have been designed to characterize this error better in the future. The error covariance matrix is currently set to be:

$$R = \begin{bmatrix} 15 & 0 \\ 0 & 15 \end{bmatrix} \quad (4.8)$$

4.6.2 Parameters

Centroids for tracking. The two possible centroids that can be used for forward tracking are listed below. The former is a mean of all the points that constitute the cluster. The latter is the centroid of a rectangle with the least area that can enclose all the points in that cluster.

- Use centroid of point cloud
- Use centroid of bounding box

The centroid of point cloud is less prone to wavering as any additional points revealed in subsequent frames will result in smaller change in the centroid of the point cloud. The downside is that this centroid does not represent the center of the vehicle and more often than not resides close to the face of the vehicle that is visible to the LiDAR.

The centroid of the bounding box represents the center of the vehicle better than the other option. The downside is that even if one point of the vehicle that is farther from the side already revealed to the LiDAR appears, the dimension of the bounding box changes. In other words, it is more prone to errors from frame to frame.

At this point in the process, association takes more importance than accurately representing the trajectory. Centroid of the point cloud in this case provides better association as it is less prone to change in number of points.

Cost function (forward tracking). This parameter determines the cost of associating one cluster in previous frame with another cluster from the next frame.

- Euclidian distance
- Angle between centroids + distance

Currently Euclidian distance is used. The cost involving angle between centroids and distance gave worse performance than Euclidian distance. This is because when a vehicle is stationary, due to the nature of LiDAR, the centroid of the point cloud still shifts. The LiDAR's LASERs do not hit the same physical point every rotation. This results in a change in centroid of the point cloud even if the vehicle is stationary. While the Euclidian distance is small, the angle estimates vary a lot.

Cost of non-assignment. The parameter that determines that the two clusters in question from consecutive frames are definitely far away from each other and are most likely not the same vehicle. Currently 300 cm is used in conjunction to using Euclidian distance as the cost function.

4.7 Dimension Estimation and Box Placement

After associating objects across frames, the next step is to estimate dimension of the object. A critical element of estimating the dimension is the angles reported by minErrorRect procedure (Section 3.4.3). But these angles are not good enough to place the estimated box back on top of the point cloud. In order to do that, we need to know the orientation of the object. To get orientation information, we use modified Bryson – Frazier smoother (Section 3.6.1).

There exists a significant challenge when calculating the orientation of the objects at rest or very low velocity from the smoothed trajectory obtained after applying MBF smoother. The LiDAR has inherent property that it will not hit the same physical point in space every rotation. This will lead to a 'motion' in centroids that does not exist in reality. In other words, the centroid reported is non-stationary.

Figure 4.4 and Figure 4.5 represent the same trajectory. The latter shows the issue of non-stationary centroid when the underlying object in question is actually stationary. To tackle this problem, we introduced an artificial lower bound for reported velocity. If the velocity of the object is less than a certain value, it is assumed to be fixed.

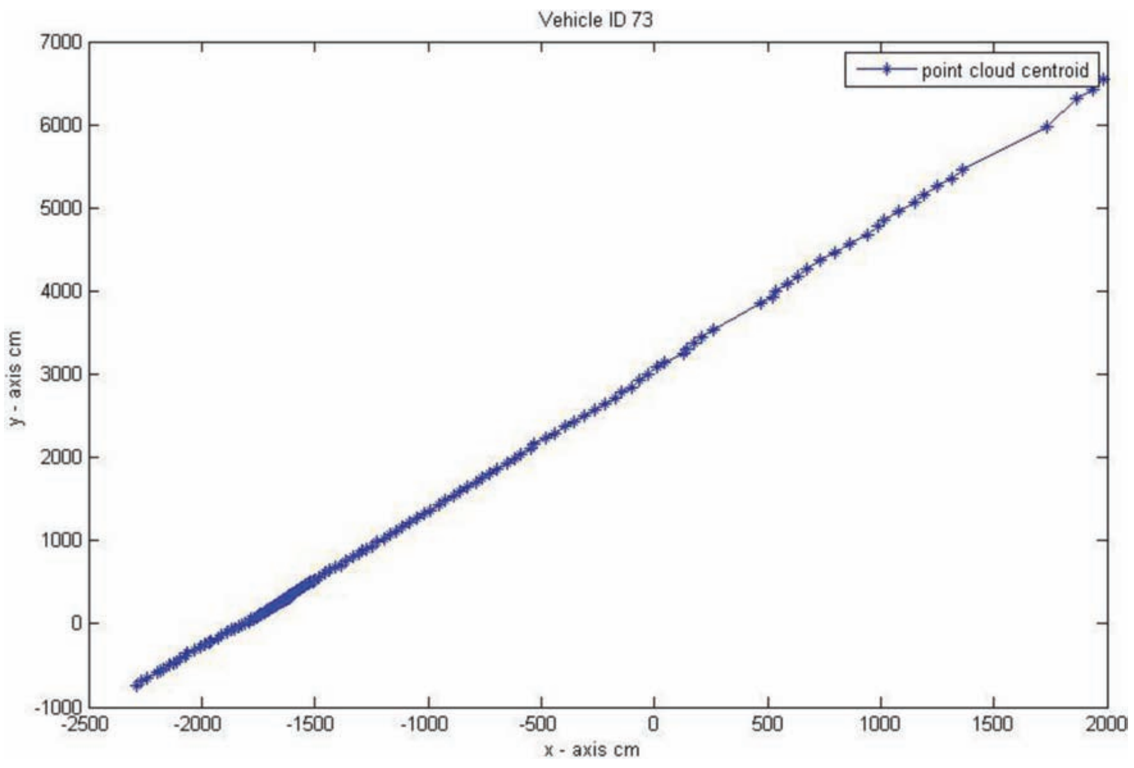


Figure 4.4 Sample trajectory of a moving object.

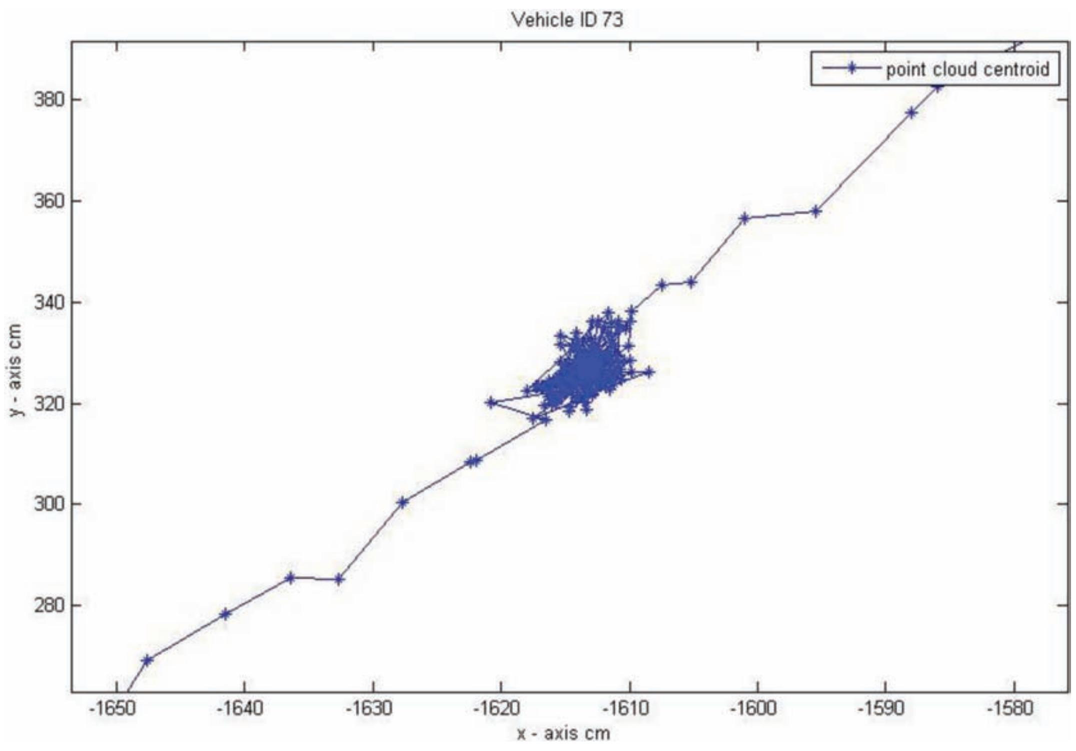


Figure 4.5 Section of trajectory shown in Figure 4.4.

Minimum Velocity

This parameter defined the velocity of the centroid of the cluster below which the cluster is assumed to be in a state of rest. Currently the centroid has to move at a speed of 1 m/s or 2.2 mph. The reason for choosing this value is as follows. The LiDAR unit has a sampling rate of 5-15 Hz. Assuming a median value of 10 (This is the spin rate used for testing and evaluation), the above said threshold of 1 m/s translates to 10 cm per frame or per rotation of the LiDAR. LiDAR has an error of 10 cm at a range of 40 m (going up to 15 cm at a range of 70 m), which is the expected distance between the LiDAR and center of intersection. Thus the sensor is limited in detecting such small movements frame to frame. Therefore, this minimum velocity is to compensate for the fundamental limitation of the sensor.

PERCENT_BOUND

The other important parameter used in this phase is PERCENT_BOUND. This represents the number of points expressed as a percentage that the bounding box has to envelop. Currently it is set at 96%, a 1% noise budget for each of the four sides of the box. The effect of this value can be visualized in Figure 3.5.

4.8 Refining Vehicle IDs

Once the dimension estimates are known the step in the process is to refine the cluster IDs and ensure that there is one to one correspondence with the number of objects. Section 3.7 talks about the need for merging and splitting trajectories and explains the method used to do so.

The process of splitting/merging is an iterative one. Convergence is said to have been achieved if *any* of the following conditions are met:

1. There are no more objects left to split and/or merge.
2. Same set of IDs fall within the ambiguous range where they are repeatedly split and merged in successive iterations.

In Section 3.7, only the first condition is mentioned as that is the most desirable and is the ideal one. In practice, we may not achieve it due to occurrence of condition number two or time limitations. In the second case, the algorithm ends up in an infinite loop if left unchecked. Hence, a manual termination is performed and the clusters are left in a split state.

3. If the process doesn't converge then the number of iterations will be changed based on available time. Once the allocated time budget is used up, the process is forced to stop.

4.9 Re-Compute Dimensions and Final Smoothing

After trajectories are split and joined as explained in Section 3.7, it is believed that most of the clusters have one to one correspondence with actual vehicles and that each individual trajectory is representative of the complete motion of vehicle within the LiDAR's field of view. Dimension of objects are estimated once again (to account for trajectories that might have changed). These estimated dimension boxes are then placed back on top of the point cloud using the algorithm explained in Section 3.6.4. The centroids of these boxes are then smoothed using local linear second order regression. The boxes are adjusted accordingly and the final trajectory estimate of the object is thus obtained.

4.10 Output

Once the trajectory and dimension of objects are estimated, the procedure explained in Section 3.9 is used to classify the objects. Finally, the results are written down in disk. TScan produces two output formats: SSAM format and a custom format. As a result, three output files are created.

4.10.1 SSAM Format

Surrogate Safety Assessment Model (SSAM), was developed by Siemens ITS with FHWA funding. It converts the outcome from micro-simulation models into safety-related output such as traffic conflicts and other risky interactions. The biggest hurdle of implementing SSAM is the lack of trustworthy simulation tools for safety modeling. TScan addresses the above problem by allowing the use of real-world data instead of simulated data.

The SSAM format output produces a trajectory file with extension .trj. It is a binary file that contains the location, speed and acceleration of each vehicle every 0.1 sec. More information can be found in Appendix B.

4.10.2 Custom Format

The custom format divides the output into two categories: time independent values and time dependent output.

The time independent file is in a comma separated value format. It contains object ID and the values that change over time such as location, speed and acceleration of each vehicle every 0.1 sec.

The time independent file is also in a comma separated value format. It contains the object ID and all measured object characteristics that cannot change in over time.

The time independent and the time dependent data is linked by the object ID.

5. ENGINEERING APPLICATIONS

This study developed one engineering application and interfaced the TScan tool with another existing application to demonstrate the new system capabilities. The first application counts the turning vehicles at intersections and the second one, SSAM, extracts the meaningful traffic safety information (conflicts and other risky interactions) from the TScan results.

5.1 Application for Directional Counting Vehicles at Intersections

The traffic volume at intersections is one of the key elements of any traffic or safety study. A wide range of technologies are used to collect traffic volumes at intersections (McShane et al. 2011). They vary from manual and simpler techniques of a human observer aided with handheld or other devices for recording data to advanced sensors based on artificial vision. Accurate estimation of traffic volumes helps estimate the current usage of a road network, predict the future volumes, and measure the risk exposure at intersections.

A TScan engineering application for counting vehicles at intersections (TScan-CV) is one of the engineering applications meant to demonstrate the TScan capabilities. This application utilizes the results of tracking vehicles to return the number of vehicles in each traffic movements.

The user interface allows selecting the TScan output file as well as the data collection settings file to create the origin-destination matrix for directional counting as well as selecting the counting interval and the starting counting period. It also displays the counting results and allows the user to export the results to a Comma Separated Value file. The TScan-CV user manual is provided in Appendix E.

5.2 TScan Interfacing with SSAM

SSAM converts the outcome from micro-simulation models into meaningful safety-related information, such as traffic conflicts and other risky interactions. SSAM was developed to use the results of micro-simulation engineering software. Although SSAM is freely available public domain software, the biggest hurdle of implementing SSAM is the lack of reliable simulation tools for safety modeling.

Interfacing the TScan data processing module with the existing SSAM addressed the above problem by allowing the use of real-world data instead of simulation. In this way, the weakest point of the safety evaluation with SSAM became its strongest point.

The TScan Data Collection and Processing Module gathers the information from the sensors, processes it, and produces the result required to export the information in the SSAM trajectory file format. The SSAM trajectory file format specifications are shown in Appendix B.

The output file was tested using SSAM Version 2.1.3, running in a Windows XP virtual machine. The results reported by SSAM are discussed in Chapter 6.

6. EVALUATION

The following chapter describes the procedure applied to evaluate TScan's performance from the user's perspective as well as for research purposes to improve the TScan algorithm.

The execution time of the algorithm was tested first because it is critical for performing the calculations in real-time during data collection. Online processing of data reduces the time required to run engineering applications after the data collection. As stated in previous sections, the research unit currently uses MATLAB as the programming language as it allows iterating on the algorithm much faster than any other low level language. The current set of algorithms explained in Chapter 3 takes up to three hours to process one hour of collected data. The background elimination is the most time consuming component, accounting for 50%-60% of the execution time. To evaluate the execution time needed in the future when the algorithm is implemented with the C++ language, the coordinate conversion algorithm (Section3.3.1) along with orientation compensation (Section3.3.2), was rewritten in C++. The C++ version executed three times faster than MATLAB which makes the existing algorithm already implementable in real time. The following additional strategies implemented together with the faster language guarantee the real-time execution of the developed algorithm:

- Processing the LiDAR data in batches
- Use of multi-threading or parallel processing features in modern CPUs to perform tasks in parallel
- Better memory management. MATLAB has limited flexibility in this regard when compared to C++
- Use of efficient data structures to transfer data between various sections of the code

It was concluded that the algorithm can be executed in real time.

The evaluation of the TScan results was conducted next by comparing them with the results obtained with an alternative benchmark method. The TScan-produced results were classified for the storage and evaluation purposes as follows:

(1) Time-independent properties of the objects: type, width, and length; the ability of the software to detect objects, properly classify them, and estimate their dimensions were evaluated.
(2) Motion of the objects: position, speed, and heading in time; the discrepancy between the results produced with TScan and with a benchmark method were estimated.
(3) Interaction between objects: conflicts and collisions extracted with SSAM from the TScan motion and dimension results were evaluated and discussed.

Computer-aided processing of video images by human observers was chosen as a benchmark method for its presumed accuracy. The benchmark method required extraction of the objects' trajectories from video images frame by frame. This method was labor-intensive and imposed limitations on the length of the evaluated periods and the number of evaluated objects.

6.1 Data Collection

Data were collected at several locations to evaluate the TScan performance in various conditions. Four-leg and three-leg intersections were included as well as signalized and non-signalized intersections. The data were collected using the TScan research version implemented in the Purdue University Mobile Traffic Laboratory (MTL). The intersections selected for data collection are described in the following sections.

6.1.1 Pedestrian Crossing on Northwestern Avenue

This intersection is located at 504 Northwestern Avenue, West Lafayette, Indiana. It is a signalized pedestrian crossing with high pedestrian volume and a median opening adjacent to the crosswalk to allow access to the Northwestern Parking Garage via a left turn. Most of the vehicles were traveling northwest and southeast. A small number of vehicles turn left towards the parking garage. The posted speed limit was 25 mph. The data were collected on December 8, 2015 from 3:12 pm until 4:27 pm. The weather was partly sunny, the temperature was 41.8 °F, and the mean wind speed was 7.25 mph. Aerial photography of the intersection with the overlapped data from TScan was shown in Figure 6.1.

6.1.2 Intersection on McCormick Road at West State Street

This signalized intersection is located in the southwest part of West Lafayette, Indiana. The intersection experienced mixed traffic with an AADT of 7,200 vehicles on the minor approach and 12,440 vehicles on the major approach (APC of Tippecanoe County, 2012). All the approaches have three lanes: one lane for through movement, another for through and right-turning movements, and an exclusive lane for left turns. The speed limits posted on West State Street and McCormick Road were 35 mph and 40 mph, respectively. The data were collected on December 17, 2015, from 11:42 AM until 12:21 AM. The weather was cloudy with a mean wind speed of 11.28 mph. Aerial photography of the intersection with the overlapped data points from the LiDAR are shown in Figure 6.2.

6.1.3 Intersection on Morehouse Road at West 350 North

Morehouse Road and West 350 North is an urban intersection administered by Tippecanoe County that is located in the northern part of West Lafayette, Indiana. It is a non-signalized intersection with three approaches. The fourth approach is private driveway accessing to a gas station with low traffic volume. The AADT on the minor road was 1,230 vehicles while the AADT on the major approach was 3,900 vehicles. The data were collected on January 26, 2016, from 6:21 PM until 7:25 PM. The time for data collection was selected to test the performance of the LiDAR during night conditions. The mean wind speed during the data

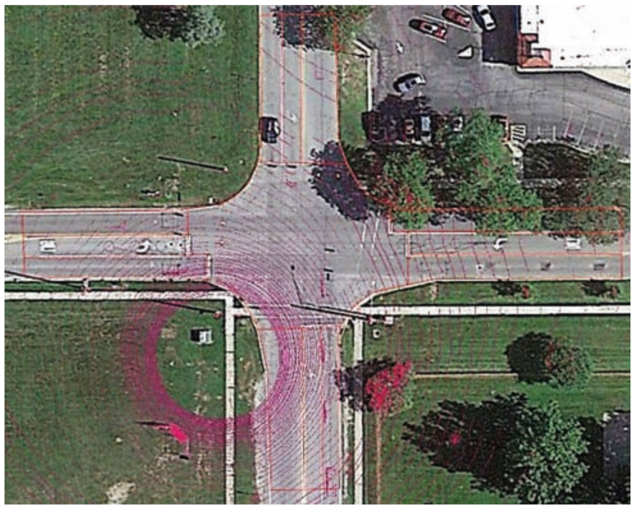


Figure 6.2 Intersection of West State Street and McCormick Road, West Lafayette, Indiana.

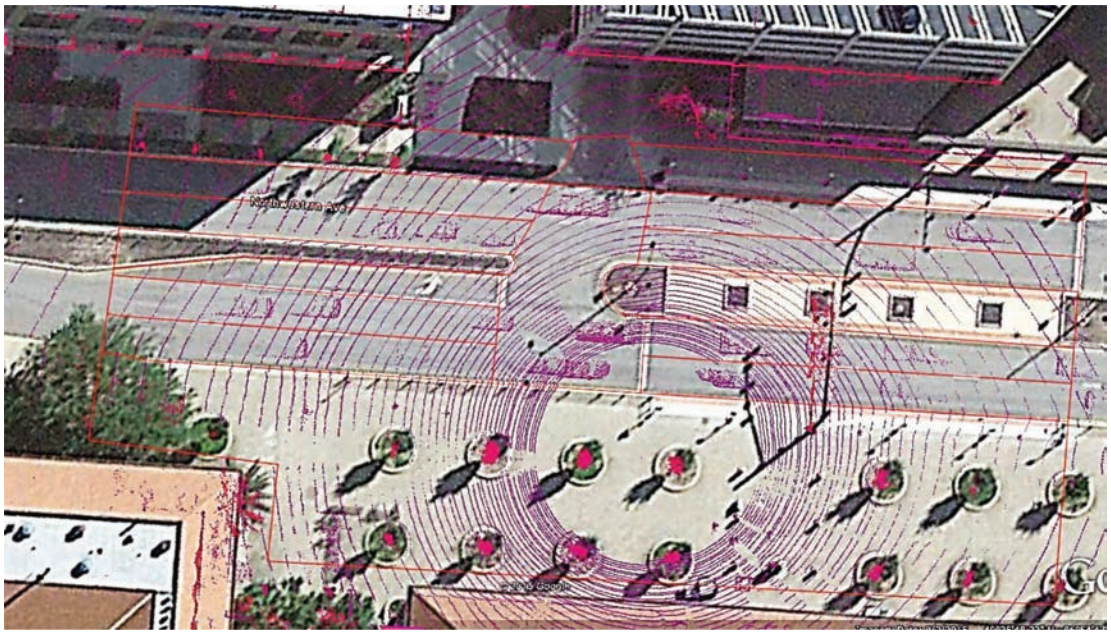


Figure 6.1 Pedestrian crossing at 504 Northwestern Avenue.

collection was 18.36 mph. The configuration of the intersection is shown in Figure 6.3.

6.2 Evaluation Methodology

A common coordinate system for the video and TScan was accomplished through aligning the video and TScan orthographic images. Data from the overlapping area of the two images were used in the evaluation. The time coordination was accomplished by estimating the shift and scale adjustments of the times of events matched in the TScan and video data. Even on the same time scale, the TScan and video-based measurements were not made at the same moments. In order to reconcile the events in time between the two methods, the positions of objects estimated with TScan were interpolated. Then, the TScan and video objects were matched in each frame using the Hungarian assignment algorithm. The methodology for extracting a vehicle's location from the videos is explained below.

6.3 Data Extraction from Video

The trajectories from video were estimated based on a customized vehicle tracking software (VTS), which is described in Romero (2010). The procedure for tracking a vehicle was developed by collecting its position at pre-specified time intervals. VTS stored the monitor coordinates (x, y) of the selected point at a specific time stamp t. Based on a double homology transformation VTS transformed the monitor-based (x, y, t) coordinates into the real-world 3D coordinates. The two consecutive homological transformations avoided estimating the parameters of the mathematical projection formula. According to Tarko, Hall, Romero, and Lizarazo (2016), at least four reference points were required to be known in both coordinate systems. The four known points on the image provided multiple

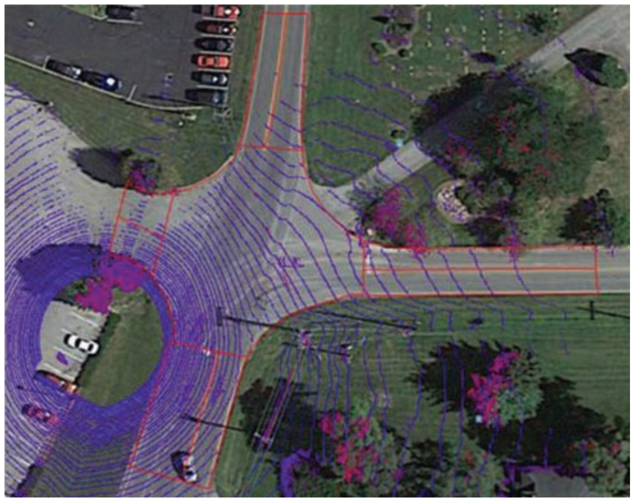


Figure 6.3 Intersection of Morehouse Road and West 350 North in West Lafayette, Indiana.

solutions to the problem. The chosen parameters were carefully selected for simplifying the estimation.

Figure 6.4 graphically shows the transformation process. To transform the coordinates from the screen system to the real world system, two homology axels were defined. The first homology axel corresponded to the relationship between the image and the auxiliary space defined by the two reference points: A′B′. The positions of A″ and B″ were known in the auxiliary drawing. The additional homology axis refers to the relationship between the reality and the auxiliary space and was defined by intercepting the point A and either C or D since A belonged to the two homology axes, A = A′= A″.

Once the homology axes were defined, the C″ location was defined in the auxiliary space. The line in the real world through B and C also was represented in the auxiliary space since B″ was known and the intersection between the free line CB and the homology axel were obtained. Then, a line through the C′D′ in the conic perspective, called the fixed line, intersected the other homology axel with a homologous point of this line into the intermediate space. The intersection of the free line and the fixed line in the auxiliary space was the point C″.

The homologies have been defined, since 4 points in three were as: real, intermediate, and image were obtained. The vertices of either of the homologies were yet to be determined. Therefore, to calculate the actual coordinates of any point, with a known image, the methodology relied on these references by defining the straight lines to a known point and calculating their counterparts in those lines (see Point P″ in Figure 6.4).

The trajectories of the vehicles were estimated by marking the points on the tires along the different video frames, forming a sequence of points that approximately represents the vehicle's trajectories (x, y, t). The video-based width and length were obtained by applying the same methodology. The procedure of marking points on the vehicle is shown in Figure 6.5.

In general, marking the points on the vehicles' tires and on the vehicle was a time consuming manual procedure. This methodology was applied to evaluate the reliability of the trajectories obtained from TScan. However, extensive data processing with this method was not feasible. The number of trajectories and dimensions for evaluating TScan are shown in Table 6.1.

Taking into account that the video information loses one dimension, thereby representing the 3D real world in a 2D conic perspective, a point in the video has infinite possible locations in the real world as shown in Figure 6.6. To perform the conic perspective transformation an additional assumption was required to obtain the point's position in the real world. The assumption used was that the points were located in the same plane. This method had different errors based on whether or not the pavement surface was similar to a plane. At the Morehouse evaluation site, the pavement design is complex in that two of the approaches were designed with a single horizontal curve with a high super elevation

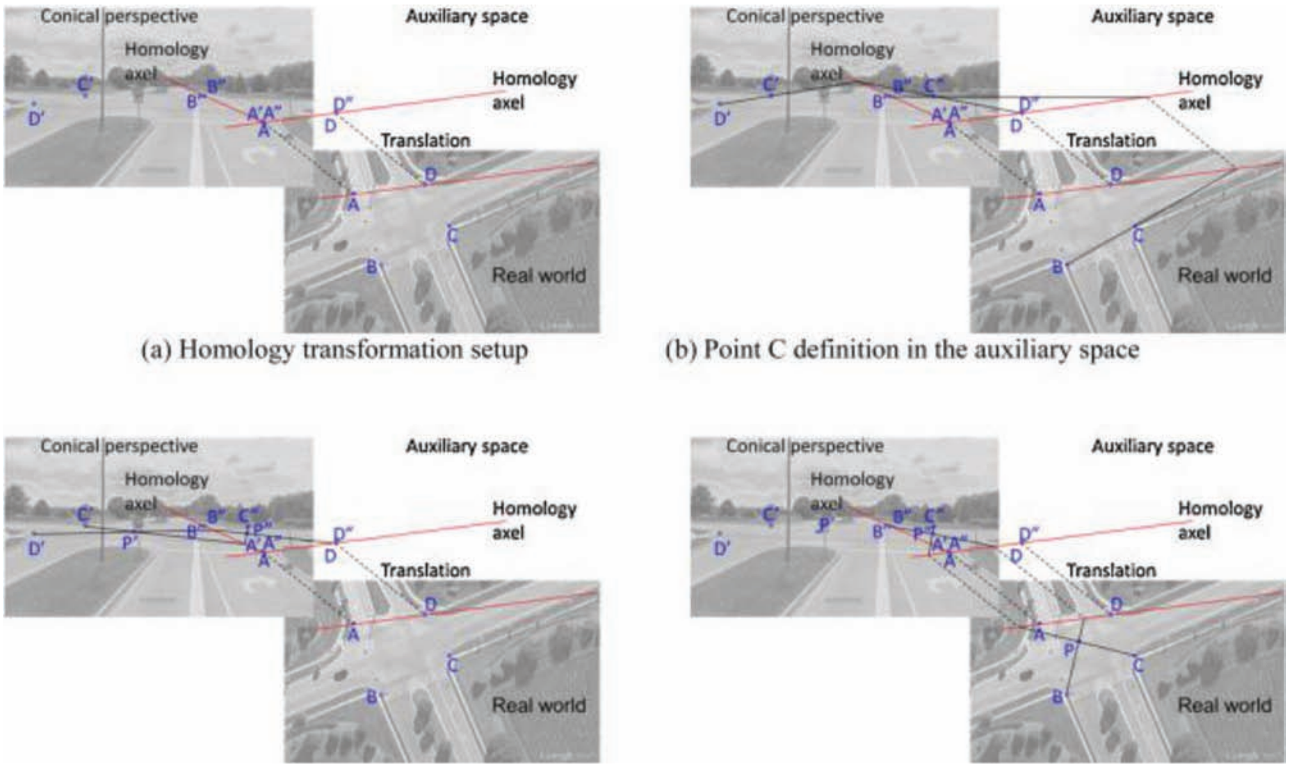

Conical perspective
Auxiliary space
Homology axel
Translation
Real world
(a) Homology transformation setup
(b) Point C definition in the auxiliary space

Figure 6.4 Double homology transformation. Source: Tarko et al. (2016).

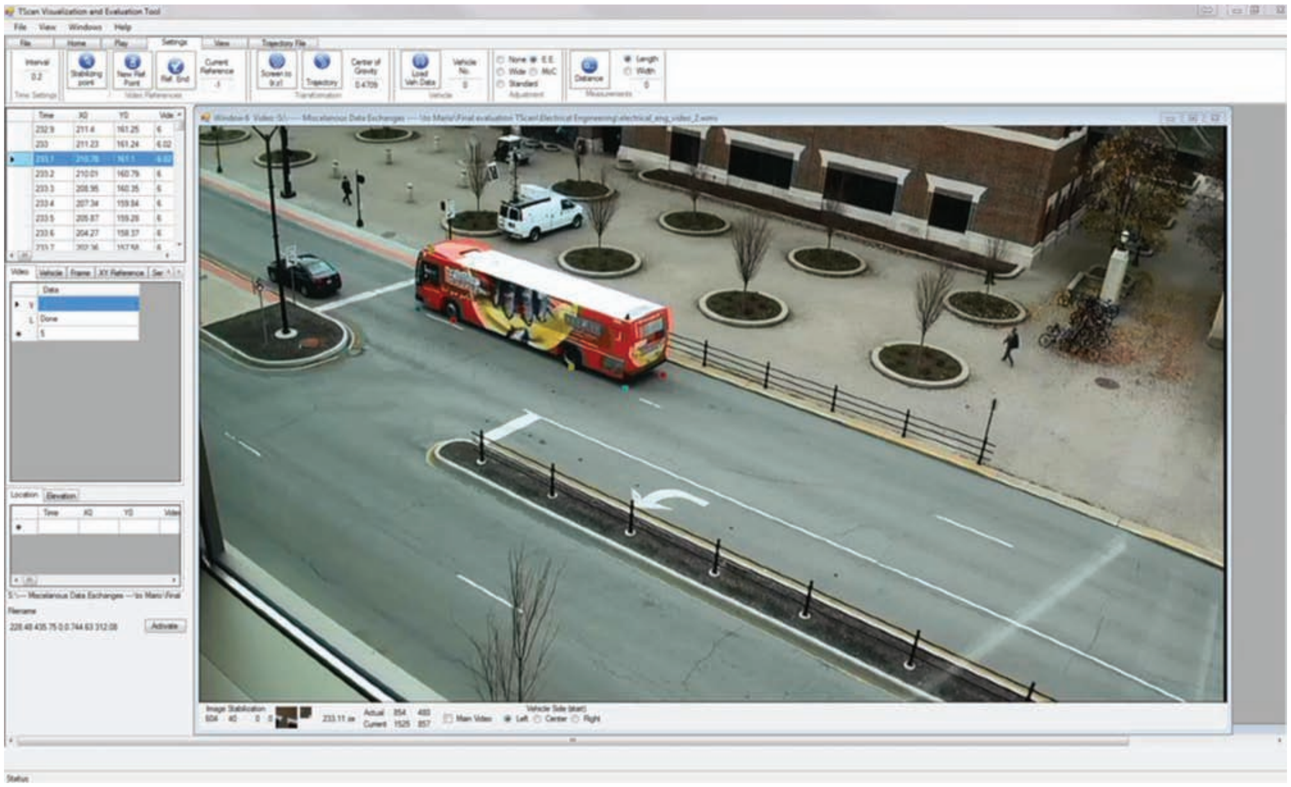


Figure 6.5 Extraction of trajectories from video.

rate and the third approach has two consecutives vertical curves. This complex topology increased the error in the location estimation of the video points.

The location error also depends on the vantage point of the video (i.e., the higher the elevation of the camera the lower the error is in the coordinate's transformation). For videos taken from a lower point, a small change in the screen locations could have a big impact in real world locations.

6.4 Results Comparison

The time-independent properties of motion were evaluated for the objects classified by humans as vehicles. The results for the evaluation of object classification, vehicle detection, vehicle dimensions, and vehicle trajectories are described in the following sections.

It must be stressed that the benchmark method, although the best available to us for a comprehensive evaluation of the TScan results, was not free of measurement errors. Two sources of errors where identified:

1. The marking of vehicles on computer monitors by human observers were not always perfect. The error is particularly considerable when the actual object is far away for the camera and appears small on the monitor.
2. The transformation from the monitor plane to the intersection plane is exact only if the intersection is indeed flat and the coordinates of the reference points perfect. The imperfection of the transformation is particularly considerable if the video camera is positioned at a low vintage point.

TABLE 6.1
Video based trajectories.

Intersection	Number of Vehicles
Intersection at Northwestern Avenue	96
Intersection West State Street and McCormick Road	105
Intersection Morehouse Road and West 350 North	48
Total	249

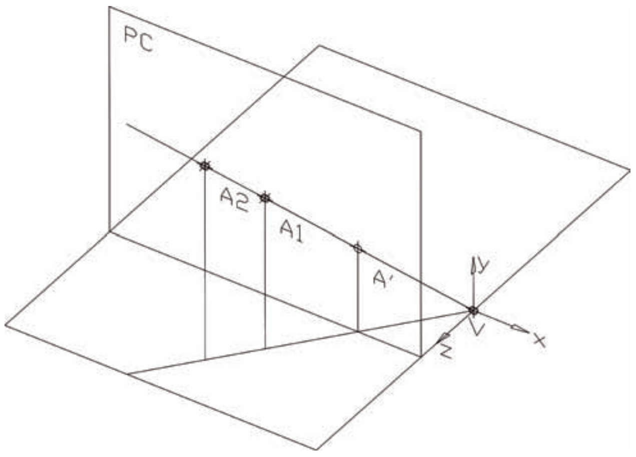


Figure 6.6 Conic perspective transformation for a single point.

The reported discrepancies between the results from the TScan and from the benchmark method provide good information about the TScan measurement error but these discrepancies are reflective also of the imperfections of the benchmark method. Thus, the reported discrepancies are just the upper-bound estimates of the TScan measurement error. This remark applies to the vehicles' positions, speeds, and directions.

Evaluating the traffic conflict counts was even a bigger challenge because there is no alternative technique sufficiently reliable for the purpose. Thus, the evaluation is based on the conflicts detected by SSAM from the TScan data and then confirmed by inspecting the visualized TScan results and the video images.

6.4.1 Objects Classification

The TScan classification of objects was evaluated first. This evaluation was conducted by comparing the object types provided by a human observing video with the object type determined by the TScan classification algorithm.

A total of 504 objects were included in the evaluation of the objective classification from TScan. The final results are shown in Table 6.2. According to the evaluation, the total accuracy of the classification was 96%. Some issues were noted when differentiating between heavy and non-heavy vehicles, which can be explained by the fact that according to the multinomial logit model, the only criterion for differentiating these objects was the vehicle length. The accuracy of classifying objects between three categories (vehicle, bicycle, pedestrian) was 98% accurate.

In general, the length of a pick-up truck is similar to a single unit truck's length. The height and length of 312 vehicle types published by their manufacturers were collected by Urazghildiiev et al (2007). Based on Figure 6.7, the length of the vehicle is not the only consideration for vehicle classification. Rather other dimensions such as height and width should be included. However,

TABLE 6.2
Final results classification of objects from TScan.

		TScan Classification			
Object	**Count**	**Pedestrian**	**Bicyclists**	**Non-Heavy Vehicle**	**Heavy Vehicle**
Pedestrians	220	**215**	5	0	0
Bicycles	11	2	**9**	0	0
Non-heavy vehicles	259	1	2	**253**	3
Heavy vehicles	14	0	0	7	**7**
Total	504	218	16	260	10

inserting them in the multinomial logit model can lead to correlation between the explanatory variables and incorrect parameter estimates. Hence, additional modeling techniques, including discriminant analysis or decision tree regression, are considered future directions of this research.

6.4.2 Vehicle Detection

There are two possible detection errors:

1. Different ID for the same vehicle – Two different IDs were assigned to the same vehicle when the trajectory was split.
2. Two different objects with the same ID – When two trajectories were combined, TScan defines the same ID for two different objects. It can be a vehicle-vehicle joint or pedestrian-vehicle joint.

The statistics related to the vehicle detection issues are shown in Table 6.3. The results report a total of five incorrect detections over the total sample size of 249 vehicles. The location with the highest number of detection errors was the pedestrian crossing at 504 Northwestern Avenue. These discrepancies can be explained by the high volume of pedestrians at this intersection. When a pedestrian walks near a vehicle, the algorithm might associate these two objects leading to an incorrect detection.

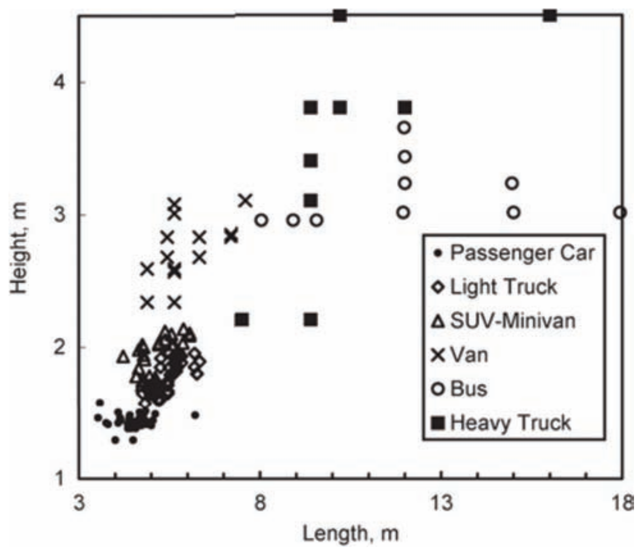


Figure 6.7 Length and width of vehicles by type. Source: Urazghildiiev et al. (2007).

6.4.3 Vehicle Dimensions

The dimensions were evaluated by estimating the discrepancy between the vehicle's width and length reported by TScan and video. The evaluation of the results was performed for each type of traffic maneuver and for each intersection. Three types of maneuvers were defined: vehicles following straight trajectories, vehicles turning left, and vehicles turning right.

The dimension of the vehicles was evaluated based on the difference between the reported vehicle's length and width from TScan and video (see Table 6.4). The length reported by TScan tended to be lower by 38 cm in average compared to video whereas the width was lower by 15 c. The differences and the standard deviations for the intersections located on Morehouse and McCormick were significantly higher, which can be explained by the fact that data extractions from video were more susceptible to error at these two locations. Since the camera locations were lower in these two scenarios, a small movement on the video was translated into a longer distance in real coordinates, which tended to produce bias in the dimensions reported by video that could cause overestimated dimensions when the clicks were not properly placed.

6.4.4 Counting Vehicles at Intersections

Vehicles were counted at intersections by their type of maneuver. The maneuver was determined based on the first and last polygons in which the vehicle was detected. A vehicle was counted in the analyzed period when its centroid crossed the intersection stop-line. The counts were evaluated at each studied intersection in five-minute intervals selected randomly at each analyzed intersection. A comparison of the counts obtained from TScan and video is shown in Table 6.5.

The average counting discrepancy error was 1.3% in all the considered cases. The highest discrepancy was found at the intersection of McCormick Road and West State Street. The three missing vehicles were marked as incomplete trajectories by the counting application. When buses were turning left on the westbound approach (West State Street), they tended to block vehicles going through or turning right on the additional lane. Hence, the vehicle paths started in the middle of intersection and classification of the maneuver types was not possible. The difference in the number of vehicles at the other two intersections was caused by the issues reported in the vehicle's detection.

TABLE 6.3
Detection errors at the analyzed intersections.

Intersection	Northwestern	McCormick	Morehouse
Two different IDs same vehicle	1	1	1
Joint Vehicle-vehicle/Pedestrian-vehicle	2	0	0

TABLE 6.4
Difference in vehicle's width and length.

Intersection	Maneuver Type	Number of Observations	Vehicles' Length (m) (Standard Deviation)	Vehicles' Width (m) (Standard Deviation)
504 Northwestern	Straight	94	-0.250 (0.305)	-0.048 (0.277)
McCormick	Straight	57	-0.123 (0.938)	-0.140 (0.365)
	Left	22	-0.069 (0.458)	-0.143 (0.552)
	Right	22	-0.106 (0.902)	-0.103 (0.463)
Morehouse	Straight	15	-0.440 (0.459)	-0.147 (0.236)
	Left	20	-0.861 (0.878)	-0.456 (0.993)
	Right	11	-0.848 (0.672)	-0.021 (0.216)

TABLE 6.5
Comparison of vehicle counts.

		Eastbound			Westbound			Northbound			Southbound		
Intersection	Method	L	T	R	L	T	R	L	T	R	L	T	R
Northwestern	Video	–	–	–	–	–	–	–	54	–	0	40	–
	TScan	–	–	–	–	–	–	–	53	–	0	40	–
	Difference	–	–	–	–	–	–	–	**-1**	–	**0**	**0**	–
McCormick	Video	4	12	4	7	15	4	1	8	2	4	7	2
	TScan	4	12	4	7	12	4	1	8	2	4	7	2
	Difference	**0**	**0**	**0**	**0**	**-3**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Morehouse	Video	0	1	0	15	0	7	3	13	5	4	10	0
	TScan	0	1	0	14	0	7	3	13	5	4	10	0
	Difference	**0**	**0**	**0**	**-1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

TABLE 6.6
Difference in trajectories estimation between TScan and video based method.

Intersection	Maneuver	Number of Observation	ΔX (m) (Standard Deviation)	ΔY (m) (Standard Deviation)	ΔSpeed (m/s) (Standard Deviation)	ΔHeading (°) (Standard Deviation)
504 Northwestern	Straight	94	0.096 (0.777)	-0.009 (0.774)	0.048 (0.861)	1.366 (6.205)
McCormick	Straight	57	-0.146 (1.404)	0.076 (1.226)	0.159 (1.937)	5.589 (23.502)
	Left	22	-0.346 (1.725)	-0.189 (2.024)	-0.147 (2.109)	2.386 (22.602)
	Right	22	0.365 (1.746)	-0.389 (1.268)	0.032 (2.088)	-6.689 (25.190)
Morehouse	Straight	15	-0.285 (1.664)	-0.238 (1.844)	0.342 (2.158)	-4.793 (18.261)
	Left	20	-0.065 (1.331)	-0.154 (1.331)	-0.676 (1.743)	-4.346 (26.183)
	Right	11	0.330 (1.174)	0.967 (1.801)	-0.825 (1.969)	-1.103 (23.549)

6.4.5 Vehicle Trajectories

The trajectories of the vehicles were evaluated based on the position, speed, and heading of the vehicles during the time when these vehicles were tracked inside the studied field of view and reported by TScan and video (see Table 6.6). The position discrepancy in the x and y coordinates was calculated separately. Higher differences and standard deviations were reported at the intersections on McCormick and Morehouse. The primary source of discrepancies at these sites are associated to the vantage point of the video cameras and the surface complexity.

The position error will be greatly reduced in the next version of the system when direct measurements of time will be conducted with an integrated GPS unit and the obtained time stamps will be embedded in the LiDAR measurements.

6.4.6 Traffic Interactions

The ability of the combined TScan and SSAM to properly detect dangerous interactions was tested by first ensuring that the TScan output data could be read and processed by the existing SSAM application. This test was passed successfully.

The TScan results processed by SSAM included 60 minutes of Northwestern traffic, 30 minutes of McCormick traffic, and 25 minutes of Morehouse traffic. There were 41 interactions extracted by SSAM during these periods. They were analyzed and classified as collisions, conflicts, or none of these two, by applying two criteria: (1) the types of involved objects and (2) the minimum speed. Specifically, in order for an interaction to be considered a collision at least one vehicle should be involved. The interactions between pedestrians were eliminated. There were also interactions

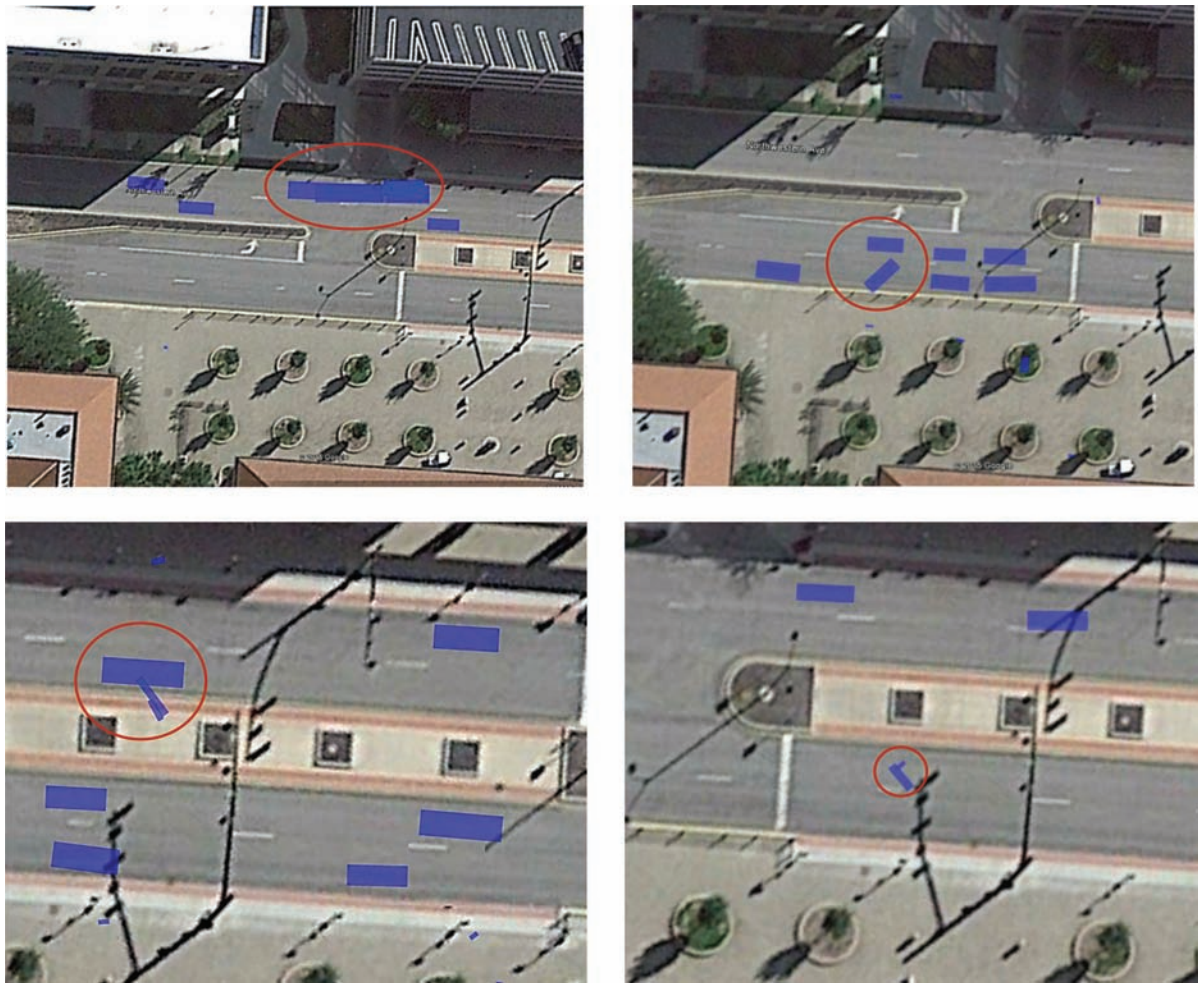


Figure 6.8 Virtual traffic conflict at the pedestrian crossing at 504 Northwestern Avenue: (a) improper clustering, (b) box placement, (c) pedestrian with overestimated dimensions, (d) pedestrian-pedestrian conflict.

between moving objects and fixed objects incorrectly left on the pavement by the background removal module. Another condition was the minimum speed of 3 miles/h of at least one of the involved vehicles. All the remaining interactions were considered collisions if there was a zero time-to-collision (TTC). Otherwise, the interaction was defined as a conflict (TTC<1.5 s). The extracted interactions categorized by the above criteria are presented in Table 6.7. After filtering out events that did not meet the collision and conflict criteria, only three events remained: one conflict (true positive), one conflict (false positive), and one collision (false positive). No false negatives could be confirmed due to the lack of an alternative benchmark method of extracting conflicts.

Although almost all the initial false positives were detected automatically, they were analyzed by inspecting the corresponding video material to identify the sources of the false positives for interactions other than pedestrian-pedestrian interactions. This analysis was believed to help improve the TScan algorithm.

For the pedestrian crossing at 504 Northwestern Avenue, SSAM reported 33 interactions during the analyzed one hour. By inspecting the results with the TScan tool, three primary issues led to the conflicts (see Figure 6.8): 1) detection errors manifested through multiple overlapping boxes representing a single vehicle, 2) vehicle position errors manifested through an unstable position of a vehicle in queue (box incorrectly directed), 3) overestimated pedestrian dimension that produced nonexistent pedestrian-vehicle conflict. The multiple pedestrian-pedestrian interactions indeed occurred, but they should not have been classified by SSAM as dangerous.

At the intersection on West State Street and McCormick Road during 30 minutes of analysis, SSAM extracted a total of six interactions. The six incorrectly "produced" conflicts and collisions were associated with an incorrect placement of boxes when the vehicles were stopped. In a single case, two slowly moving vehicles in adjacent lanes (Figure 6.9) collided "virtually." This event did not occur in reality.

TABLE 6.7
Traffic interactions at the studied intersections.

Case	Northwestern		McCormick		Morehouse	
	Collisions	**Conflicts**	**Collisions**	**Conflicts**	**Collisions**	**Conflicts**
SSAM Extracted						
True positives	0	0	0	0	0	1
False positives	22	11	5	1	1	0
After Filtering						
True positives	0	0	0	0	0	1
False positives	0	1	1	0	0	0

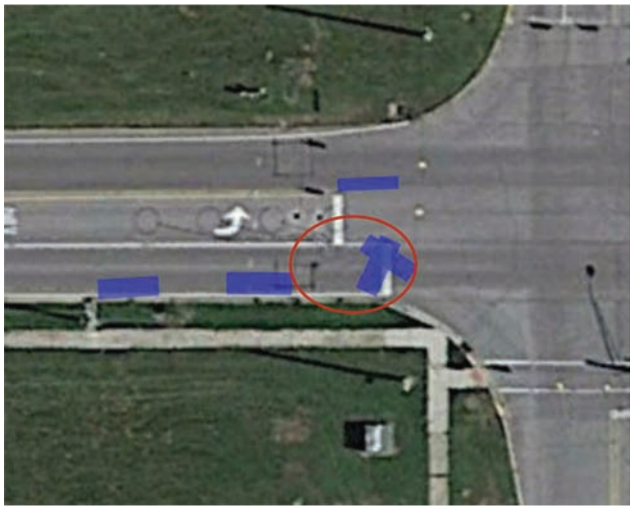


Figure 6.9 Conflict at McCormick intersection because of an incorrect placement of two vehicles.

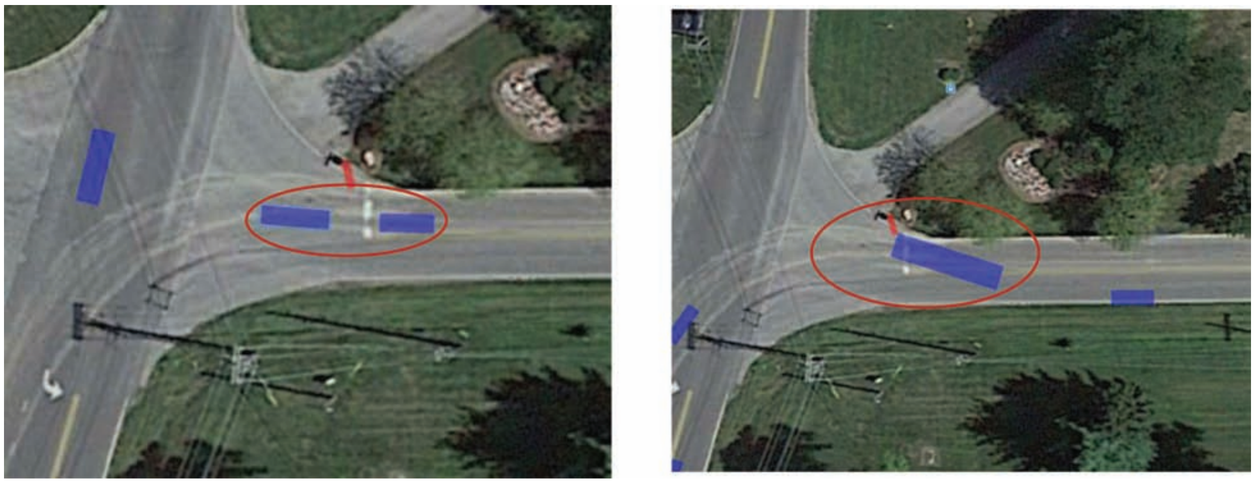


Figure 6.10 Conflicts at Morehouse intersection: (a) real conflict, (b) incorrect background removal.

Finally, at the intersection of Morehouse Road and West 350 North Street, SSAM detected two conflicts. One of them was an actual conflict between two vehicles (Figure 6.10a), and the estimated time to collision in this case was 1.4 seconds. The second one was caused by the failure to remove a part of the background on the westbound approach of the intersection (Figure 6.10b).

A summary of the number of interactions and their sources at each intersection is shown in Table 6.8.

TABLE 6.8
Sources of the false positive traffic interactions.

	Northwestern		McCormick		Morehouse	
Source	**Collisions**	**Conflicts**	**Collisions**	**Conflicts**	**Collisions**	**Conflicts**
Pedestrian-pedestrian interactions	12	8	0	0	0	0
Object position error (in queue)	4	0	4	1	0	0
Background removal error (fixed object)	0	0	0	0	1	0
Detection error (multiple overlapping boxes)	6	2	0	0	0	0
Dimension error (pedestrian)	0	1	0	0	0	0
Object position error (low speed in queue)	0	0	1	0	0	0
Total	22	11	5	1	1	0

7. DISCUSSION

The evaluation results discussed in the previous chapter indicate that TScan in its current version is capable of detecting moving objects, classifying them, and tracking across the field of view at the error acceptable for the envisioned types of traffic studies. The initially high rate of traffic interactions detected by SSAM was reduced to the acceptable level by applying the definition of traffic conflicts that excludes interactions at very low speed and between pedestrians – conditions that are not considered with the SSAM-based processing.

This chapter presents thoughts and recommendations geared towards further improvement of the results. It discusses the knowledge gained during the course of the presented study as well as the challenges and their sources and solutions, which either were applied in the current version of the method or are recommended to be considered in future efforts. Some of the challenges were discovered in the initial stage of the study when the LiDAR sensor was tested and the method was being developed; other challenges became obvious after evaluating the performance of the system. The challenges discovered early were addressed during the study, while for others, the sources of the issues were identified and promising solutions are proposed. This chapter concludes with a summary of the accomplishments of this study.

The main challenge with the LiDAR sensor that was discovered in the initial stages of this study was that the density of the returns was small. The low density of the data points was further reduced by long distances between the sensor and an object, light absorption on dark objects, and light reflection away from the sensor on oblique surfaces. The portion of light returning to the sensor was sometime too small to be detected, which meant that the algorithms had to effectively use all the available points, whether in the background or on the moving object. The algorithms for sparse sensing were developed as a result, with very good results, and they should easily apply to future prototypes that produce denser point clouds. Also, using two units or the partial integration of video with LiDAR can produce a far greater density of points immediately.

The current version of the algorithms does not utilize any feedback from the tracking routine to support more effective clustering of data points to detect moving objects. This feedback did not appear to be necessary during the development of the algorithms. Although this still may be claimed, the subsequent evaluation of the TScan results indicated that such feedback could help avoid incorrect merging of pedestrians and vehicles. The current resolution of LiDAR is insufficient to confidently distinguish between the two when a pedestrian is less than 50 cm away from another object, whether it is another pedestrian or a vehicle. A new method using the classification algorithm after forward tracking to identify pedestrians and then using that information to re-cluster the points after background elimination is being investigated. This method will also help prevent issues regarding vehicle-pedestrian interaction. A quite opposite problem is posed by long vehicles with trailers. In this case, the scarcity of returned light beams may lead to breaking a single object into two or more clusters. The current algorithm requires overlapping between clusters to allow their combination. The feedback from tracking, which indicates similar trajectories of multiple and closely-spaced clusters, may help identify these clusters as a single object. Although these cases were infrequent and not critical for traffic analysis, all of them produced false positives among the traffic interactions reported by SSAM. Since dangerous interactions are infrequent, even a small number of false positives can skew the results considerably. Fortunately, these cases could be easily detected and eliminated automatically by post-processing the SSAM results. Nevertheless, improving the clustering of points with the feedback information in the tracking phase should be considered in future studies.

Another challenge detected in the evaluation phase of the study was the incorrect orientation of rectangular shapes fitted to clusters that represented objects. In several analyzed cases, the incorrect box placement was caused by the combination of a vehicle stopped or moving at a low speed and a number of points varying over time for the same object. This variation was causing movement of the cluster centroid that did not follow the actual motion (or lack of motion) of the object (see Figure 4.5). Currently, the box placement algorithm depends on the orientation information obtained via the Modified Bryson Frazier smoother. The orientation is calculated based on two subsequent centroid values and makes the orientation sensitive to very small shifts in trajectory. Using longer segments of trajectories to identify the orientation of the object can help mitigate this issue.

The algorithm for classifying objects correctly separates motorized and non-motorized objects. However, the multinomial logit model exhibits difficulties in differentiating between heavy and non-heavy vehicles. In order to provide a more accurate classification, the mode should utilize the width and height of the vehicle in addition to the currently used length. Furthermore, other methods, such as discriminant analysis or decision tree regression, should be checked.

The TScan research unit measures the time when data are written to the storage. A small delay may occur in this process. Future versions of the TScan system will be equipped with a GPS that measures the time without the mentioned delay. Accurate time synchronization between the video and LiDAR data is critical when the two data streams are to be combined.

For the research unit, all the algorithms were prototyped in MATLAB (Mathworks, n.d.). Using MATLAB allowed modification of the algorithm and visualized the results much faster than implementing it in a lower level language like C++. Since the current setup is geared towards rapid prototyping and largely emphasizes lessening the time taken to implement changes, the Data Collection and Processing Module (TScan software) was not used. Additional software (Velodyne's Digital Sensor Recorder (DSR) and Vectornav's Sensor Explorer) were used to collect data. The Data Collection and Processing Module encompasses the functionality of the additional software. The final version of the system uses its own software alone.

The developed TScan Traffic Counter counts moving objects and classifies their movements on complete trajectories, which may lead to undercounting objects if some trajectories end in the middle of the intersection. An improvement is possible for exclusive traffic lanes when the traffic movement is known for the lanes.

The SSAM file format does not include types of objects, thus pedestrian-pedestrian interactions are reported as valid conflicts and collisions. This issue is easy to solve through post-processing.

SSAM uses 2D representation of objects projected on the xy plane, which can lead to reporting conflicts and collisions between moving objects and fixed objects such as tree branches intruding in the road clear space.

An extensive evaluation of the testing methods for all the components of TScan was conducted. The evaluation identified two limitations of the benchmark method which prohibit it from being characterized as the ground truth. The first limitation is related to the assumption that all the transformed points are located in the same plane, which deviates from reality, especially at intersections with steep cross-slopes. This can be mitigated by measuring the road surface with TScan to use multiple projection planes that better represent the actual surface. Otherwise, the video-based method cannot be used as a ground truth method. The second limitation is the relation between the error from video and the height where it was recorded. In general, the higher the video camera, the lower the error was in the coordinate's transformation.

TScan's average undercounting error was just 2% for all the considered objects, which could be reduced further. Two types of detection errors were identified. The first error was in assigning two different IDs to the same vehicle, leading to incomplete trajectories that were ignored during counting because the object was occluded for a considerable time. Including an additional LiDAR sensor would reduce or even eliminate this first error. The second error was caused by incorrect clustering, which can be improved by utilizing the feedback from the tracking phase.

The dimensions of vehicles were slightly underestimated as indicated by the mean error. Due to the limitations of the benchmark method, the error of standard deviation was considerable. However, this error did not affect the automatic classification of objects, such as vehicles, pedestrians, and bicycles.

The post-processing of the TScan results with SSAM produced 41 traffic interactions during the period of analysis, which was equal to 115 minutes. The automated post-filtering of the events that did not meet the collision and conflict criteria produced one true positive conflict, one false positive conflict, and one false positive crash. The two false positive detections were attributed to the imperfect clustering of the data points. The method for improving this operation was discussed earlier in this chapter.

Although almost all the false positives were filtered out automatically, they were inspected to determine the origin of the errors. The most common source was pedestrian-pedestrian interactions, which were reported by SSAM as valid traffic interactions. Since SSAM is limited to analyzing only interaction between vehicles, a post-processing method is needed based on the classification algorithm to detect these false positives conflicts. An additional limitation of SSAM arose when including three-dimensional information. SSAM generally reports conflicts based on a two-dimensional location of the object. By including the third dimension, issues related to fixed objects could be eliminated since fixed objects occurring on roads are trees or traffic signals located above the traffic. The remaining false positives were caused by imperfect clustering of data points, and more specifically, they represented large vehicles with multiple clusters. This problem prompts improvement of the clustering method as already discussed.

8. CONCLUSION

The presented research study met all of its objectives. A TScan research unit was successfully developed together with the companion software for setting the system, acquiring the 3D location data of surrounding surface, processing the measurements to successfully detect moving objects, tracking them across the field of view, and classifying objects as heavy vehicles, non-heavy vehicles, pedestrians, and bicycles. The data processing can be performed during data collection; thus, the original large data files are reduced to less than one percent of the original size and the post-processing for engineering studies is quickly conducted. An engineering application for counting moving objects was developed; and one existing application, SSAM, for counting traffic conflicts and collisions, was interfaced with the TScan system.

The TScan research unit was applied at three intersections to collect traffic data for evaluation purposes. Objects could be detected and tracked within 200 feet of the location of the TScan; and the objects could be tracked along paths up to 400 feet depending on the location of the TScan. The evaluation results indicate that the TScan method, in its current version, provides sufficiently accurate counts of vehicles at intersections and measurement of the speeds and paths of vehicles, pedestrians, and bicycles. These measurements do not require human involvement in data processing.

TScan was able to estimate the trajectories of road users at a level of accuracy that allows conflict-based safety analysis. This ability was tested with SSAM in order to extract dangerous interactions. Although the TScan-estimated object trajectories processed with SSAM produced false positives, almost all of them could be easily filtered out by applying a proper condition for a valid conflict or collision. The sources of false positives were identified in this study to help eliminate them by improving the TScan method rather than through post-processing. These measurements do not require human involvement in data processing.

The LiDAR was evaluated in nighttime conditions and during light precipitation and was able to track vehicles without any difficulty. It is hypothesized that as long as the precipitation is not strong enough to disperse the lasers of the sensors, the LiDAR will be able to perform as reported. Further testing is necessary to determine the exact threshold of working conditions of the LiDAR for TScan.

Although the current method devised in this study for tracking vehicles showed acceptable performance, there is still room for improvement in the modules responsible for clustering data points to detect objects and for positioning rectangular shapes as simplified representations of objects.

The algorithms are designed with a sparse density of points as the target. Therefore, any improvement in density either through the addition of another LiDAR sensor or integration with video will yield dividends immediately. Also, the current algorithms are insensitive to the source of the point cloud. In other words, once the various sources of information are converted to 3D point clouds in XYZ Cartesian coordinates, all the developed algorithms can be used with little to no modification.

Integrating LiDAR with video will directly relate depth to the image regardless of the feature density. Even stereoscopic video systems cannot accomplish what can be with TScan. Stereoscopic video systems rely on the difference in the relative positions of the same feature in two separate images. To accomplish this, the feature detection method used in computer vision and image processing was needed. There is no universal or exact definition of what constitutes a feature in systems based purely on video because the densities of features change depending on the image.

The results of this study and the experience gained during its conduct were the basis for developing the specifications in Appendix F, which will be helpful for developing a trailer-based unit.

REFERENCES

APC of Tippecanoe County. (2012). *Seasonally adjusted average daily traffic*. Lafayette, IN: The Area Plan Commission of Tippecanoe County. Retrieved from http://www.tippecanoe.in.gov/documentcenter/view/12538

Bierman, G. J. (1977). *Factorization methods for discrete sequential estimation* (Mathematics in Science and Engineering 128). New York, NY: Academic Press.

Cheung, H. (2007). Spinning LASER maker is the real winner of the urban challenge. *TG Daily*, November 7. Retrieved from http://www.tgdaily.com/trendwatch-features/34750-spinning-laser-maker-is-the-real-winner-of-the-urban-challenge

Delaunay, B. (1934). Sur la sphère vide. *Bulletin de l'Académie des Sciences de l'URSS, Classe des sciences mathématiques et naturelles* issue 6, 793–800. Retrieved from http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=im&paperid=4937&option_lang=eng

Glennie, C., & Lichti, D. D. (2010). Static calibration and analysis of the Velodyne HDL-64E S2 for high accuracy mobile scanning. *Remote Sensing*, *2*(6), 1610–1624. http://dx.doi.org/10.3390/rs2061610

Harold, W. K (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, *2*(1–2), 83–97. http://dx.doi/org/10.1002/nav.3800020109

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*(1), 35–45. http://dx.doi.org/10.1115/1.3662552

MathWorks. (n.d.). MATLAB [Online]. Retrieved May 16, 2016, from http://www.mathworks.com/products/matlab

Rauch, H. E., Tung, F., & Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, *3*(8), 1445–1450. http://dx.doi.org/10.2514/3.3166

Romero, M. (2010). *Modelo de diseño de la longitud de los carriles de deceleración paralelos, basado en el desarrollo y la aplicación de un indicador de la conflictividad de las maniobras de salida* (Doctoral dissertation). Valencia, Spain: Polytechnic University of Valencia. Retreived from https://riunet.upv.es/bitstream/handle/10251/8333/tesisUPV3290.pdf

Tarko, A. P., Davis, G, Saunier, N., Sayed, T., & Washington, S. (2009). Surrogate measures of safety [Unpublished white paper].

Tarko, A., Hall, T., Romero, M., & Lizarazo, C. (2016). Evaluating the rollover propensity of trucks — A roundabout example. *Accident Analysis & Prevention*, *91*, 127–134. http://dx.doi.org/10.1016/j.aap.2016.02.032

Urazghildiiev, I., Ragnarsson, R., Ridderstrom, P., Rydberg, A., Ojefors, E., Wallin, K., ... Lofqvist, G. (2007). Vehicle classification based on the radar measurement of height profiles. *IEEE Transactions on Intelligent Transportation Systems*, *8*(2), 245–253. http://dx.doi.org/10.1109/TITS.2006.890071

Velodyne. (n.d.a). High definition LiDAR—HDL 64E—Datasheet [Online]. Retrieved June 27, 2014, from http://velodynelidar.com/lidar/products/brochure/HDL-64E%20S2%20datasheet.pdf

Velodyne. (n.d.b). *High definition LiDAR sensor HDL-64E S2 and S2.1: User's manual and programming guide*. Morgan Hill, CA: Velodyne. Retrieved June 27, 2014, from http://velodynelidar.com/docs/manuals/63-HDL64ES2h%20HDL-64E%20S2%20CD%20Users%20Manual.pdf

APPENDICES

APPENDIX A. RESEARCH UNIT HARDWARE

The Purdue University Mobile Traffic Laboratory (MTL) is built based on a Chevy Express 3500 van and is equipped with a 42-foot pneumatic mast that can be operated from inside the van. It includes two PTZ IP dome cameras and two flat screen monitors, a computer, an eight-channel video recorder, and a gigabit Ethernet network. The equipment is powered by a heavy-duty inverter and almost all the equipment is rack-mountable for safe transportation.

For the TScan system, a LiDAR three-dimensional laser scanning technology was integrated into the MTL with three IMUs to track the orientation of the LiDAR in real time. The viewing position of the LiDAR can be further adjusted with a pan and tilt base controlled from inside the van.

The characteristics of the research equipment are described in the following sections.

A.1 Sensors and Cameras

The sensors and cameras installed on the research unit are the LiDAR, three IMUs, and two cameras. The Sensors specifications are shown in Table A.1, Table A.2, and Table A.3.

A.2 Computer, Storage Units and Network

The computer and storage units characteristics installed on the research unit are shown in Table A.4, Table A.5, Table A.6, and Table A.7.

A.3 Power Supply

The MTL has an inverter to generate the 110 VAC needed for the electronic equipment. The inverter takes 12 DCV from batteries that are charged using the van's engine. The inverter specifications are shown in Table A.8.

A.4 Mast and Base

The MTL is equipped with a telescopic mast that allows user to rise the sensors to a better vantage point. Then A Pan-Tilt base is used to better orientate the LiDAR. The mast and the Pant-tilt base specifications are shown in Table A.9 and Table A.10, respectively.

TABLE A.1
LiDAR specifications.

Brand and model	Velodyne HDL-64E
Number of lasers	64
Range	Up to 120 m
Horizontal field of view	360 deg
Vertical	26.8 deg
Angular resolution	0.09 deg
Vertical resolution	0.4 deg
Frame rate	From 5 to 15 Hz.

TABLE A.2
Inertial measurement unit specifications.

Brand and model	Velodyne HDL-64E	
Sensors	***Accelerometer Specifications***	
	Number of axis:	3
	Range:	±16 g
	In-Run Bias Stability	<0.04 mg
	Linearity:	<0.5° FS
	Noise Density:	<0.14 mg/√Hz
	Bandwidth:	260 HZ
	Alignment Error:	±0.05°
	Resolution:	<0.5 mg
	Gyroscope Specifications	
	Number of axis:	3
	Range:	±2000°/s
	In-Run Bias Stability:	<10°/hr
	Linearity:	<0.1% FS
	Noise Density:	0.0035°/s √Hz
	Bandwidth:	256 Hz
	Alignment Error:	±0.05°
	Resolution:	<0.02°/s
	Magnetometer Specifications	
	Number of axis:	3
	Range:	±2.5 Gauss
	Linearity:	<0.1%
	Noise Density:	140 µGauss/√Hz
	Bandwidth:	200 HZ
	Alignment Error:	±0.05°
	Resolution:	1.5 Milligauss
	1 Barometric pressure sensor	
Communications	Serial RS-232 & TTL	
Angular resolution	<0.05 deg	
Output rate	800 Hz	

TABLE A.3
Cameras specifications.

Brand and model	Vivotek SD8321E
Zoom	18x optical
Pan Range	360 deg
Tilt range	0–90 deg
Frame rate	Up to 50 fps at 720x576
	Up to 25 fps in Wide Dynamic Range mode

TABLE A.4
Computer specifications.

Processor	Intel Core^{TM} i7-2600 CPU @ 3.40 GHz
Operating System	Windows 10, 64-bit
Memory	32GB, 1600MHz, DDR3
Hard Drive	1TB 7200 rpm Hard Drive
Optical Drive	DVD-RW Drive (Reads and Writes to DVD/CD)
	2 USB 3.0 ports
	6 USB 2.0 Ports
Ports	1 HDMI
	1 Display Port
	1 RJ-45 (10/100/1000Base)
Slots	4 DIMMs
Keyboard	Wired Keyboard
Mouse	Optical Mouse

TABLE A.5
Storage unit specifications.

Brand and model	Synology disk Statino DS213j
HDD bays	2
Capacity	3 TB

TABLE A.6
Video recorder specifications.

Brand and model	NUUO NVRmini
Number of channels	8
HDD bays	4
Capacity	6 TB
Raid Levels	0,1,5,10
Operating system	Embedded Linux

TABLE A.7
Network specifications.

Switch	
Brand and model	D-link - dgs1005g
Number of ports	5
Ethernet Technology	Gigabit Ethernet
Network Technology	10/100/1000Base-T
Router	
Brand and model	Linksys – WRT54G
Number of ports	4
Network Technology	10/100

TABLE A.8
Inverter specifications.

Brand and model	Sunforce 11260
Power	2500 Watts continuous/5000 watts surge
Input voltage	12 VDC
Output voltage	110 VAC

TABLE A.9
Mast specifications.

Type	Pneumatic
Extended height	42 ft
Number of segments	8
Collar type	Locking
Maximum load	200 lbs
Tube Diameter	9-3 in / 229-76 mm
Compressor Voltage	12 VDC
Maximum operating pressure	35 PSIG (2.4 bar)

TABLE A.10
Pan tilt base specifications.

Model	PTB-1
Rotation	0 to 355 deg
Tilt	-60 to 30 deg
Maximum load	100 lbs
Remote controller	Hard wired & wireless
Voltage	12 VDC

APPENDIX B. SSAM FILE FORMAT SPECIFICATION

Version 1.04 of SSAM is used in the TScan system.

The trajectory file records the location of each vehicle in a single simulation run for every time step of the simulation. Each trajectory file is expected to be named with a .“trj” (or .“TRJ”) extension. It utilizes a binary format in order to keep trajectory files from large network simulations from growing excessively large. The file is organized with a set of records, which are each identified by a single, initial byte value as presented in Table B.1.

Each record is of fixed number of bytes (though different record types have different sizes) and is defined in a corresponding table, which lists the fields that will appear in the record in order of appearance, with name-type-value descriptions. See Table 1.2 for an example of the first record. The first record-type byte specifies a FORMAT record, and a FORMAT record, as defined in Table B.2, contains includes an additional 5 bytes. (Note that all Byte values are encoded unsigned, whereas all Integer and Float types are encoded as signed, 4-byte values.) After the initial record-type field, the next field in the FORMAT record is a single byte value that specifies the “endianess” of the file. This byte is an ASCII formatted capital L if the file is encoded in little endian format, or the byte is an ASCII formatted capital B if the file is encoded in big endian format. This allows easier multi-platform support for the trajectory file format. The next field is the Version, which is specified as a 4-byte floating-point value. The current version of the trajectory file is 1.04.

The records of the trajectory file are organized as follows. The file starts with a single FORMAT record, specifying whether multi-byte values are encoded in big or little endian order and what version of the trajectory file is supported. Next, a single DIMENSIONS record specifies the extent of the rectangular region of the vehicle observation area in terms of the x-y coordinates. Then, a series of time-steps are encoded consecutively in chronological order. Each time step begins with a TIMESTEP record and is followed by a variable number of VEHICLE records indicating the vehicle’s locations during that time step. The file generally includes several thousand time-steps, and simply terminates when no more data are available. Figure B.1 depicts the general layout of the trajectory file in terms of record types.

The locations of all vehicles in a trajectory file are specified using the x and y coordinates. The observation area within which these vehicles travel is specified as a rectangular region using the DIMENSIONS record defined in Table B.3 according to a normal Cartesian coordinate system, where the rectangle is parallel to the x and y axes, and the x and y values increase to the right

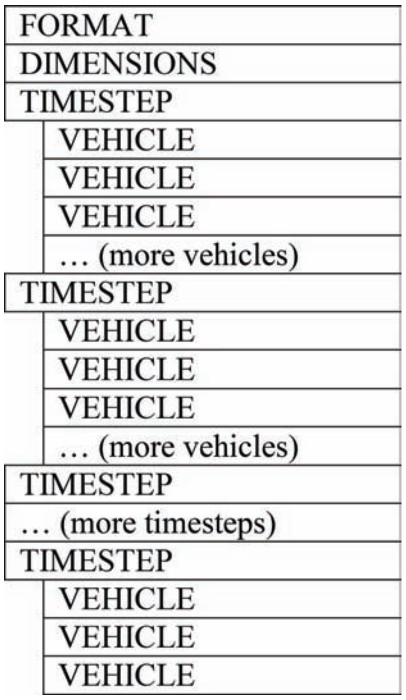


Figure B.1 Organization of records in the trajectory file.

TABLE B.1
Available record types in the trajectory file.

Record Type	Record ID	Record Description
FORMAT	0	Specifies little or big endian format and version number
DIMENSIONS	1	Specifies X-Y bounds of observation area and scale
TIMESTEP	2	Simulation time
VEHICLE	3	Specifies the location of a single vehicle for the timestep

TABLE B.2
Format record description.

Field Name	Type	Value Description
Record Type	Byte	0 = FORMAT record type
Endian	Byte	ASCII ‘L’ = little endian, used by Intel platforms
		ASCII ‘B’ = big endian, used by Motorola (Mac/Unix)
Version	Float	Allows decimal version number, which is currently 1.04

and up respectively. As a practical matter, the size of this region must be less than 10 square miles. The rectangular region is perhaps most intuitively scaled at one foot or meter per unit of x or y. The floating point precision is used to specify the scaling and the x-y coordinates. Note that while double precision would accommodate full global mapping (e.g., latitude and longitude) it also imposes the need for substantially greater computation time and memory. Thus, single precision coordinates are used as a practical matter at this time.

Each time-step of vehicle data begins with a TIME STEP record, as defined in Table B.4, which specifies the elapsed time, in seconds, since the start of the simulation (or field observation). This format allows time-steps to be specified with variable precision, but a precision of 1/10th of a second is most likely. Note that data as infrequent as once-per second could be insufficient for accurate conflict analysis.

Following each TIMESTEP a series of VEHICLE records specify the location of each vehicle during the time-step. The VEHICLE record is shown in Table B.5. All x and y values are to be encoded as scaled values in the units specified in the DIMENSIONS record. All length, width, speed, and acceleration values are to be encoded as un-scaled values in the units (i.e., feet or meters) specified in the DIMENSIONS record.

TABLE B.3
Dimensions record description.

Field Name	Type	Value Description
Record Type	Byte	1 = DIMENSIONS record type
Units	Byte	= English (i.e., feet, feet/sec, feet/sec^2) = Metric (i.e., meters, meters/sec, meters/sec^2)
Scale	Float	Distance per unit of X or Y (i.e., per "pixel") (e.g., if scale is 0.25 and the units are metric, then x = 0 is 0.25 meters left of x = 1)
MinX	Integer	Left edge of the observation area.
MinY	Integer	Bottom edge of the observation area.
MaxX	Integer	Right edge of the observation area.
MaxY	Integer	Top edge of the observation area.

TABLE B.4
Timestep record description.

Field Name	Type	Value Description
Record Type	Byte	2 = TIMESTEP record type
Timestep	Float	Seconds since the start of the simulation

TABLE B.5
Vehicle record description.

Field Name	Type	Value Description
Record Type	Byte	3 = VEHICLE record type
Vehicle ID	Integer	Unique identifier number of the vehicle
Link ID	Integer	Unique identifier number of the link (where possible)
Lane ID	Byte	Unique identifier number of the lane (where possible)
Front X	Float	X coordinate of the middle front bumper of the vehicle
Front Y	Float	Y coordinate of the middle front bumper of the vehicle
Rear X	Float	X coordinate of the middle rear bumper of the vehicle
Rear Y	Float	Y coordinate of the middle rear bumper of the vehicle
Length	Float	Vehicle length (front to back) in Units (feet or meters)
Width	Float	Vehicle width (left to right) in Units (feet or meters)
Speed	Float	Instantaneous forward speed (Units/sec)
Acceleration	Float	Instantaneous forward acceleration (Units/sec^2)

APPENDIX C. TSCAN OUTPUT FILE FORMAT

TScan divides the output into two categories: time-independent values and time-dependent values.

A value is considered as time-independent if the measurement cannot change over time. Most of the values that fall into this category are the object's characteristics plus other general values.

On the other hand, a time-dependent value must change over time. The values that describe the object's movement are those that fall into this category.

The TScan custom output contains two files, one for each category of data. The time-independent and the time-dependent data are linked by the object ID.

C.1 Time Independent File

The time-independent file is in a comma separated value format. The first row contains the variable name and the following columns contain the data as shown in Figure C.1.

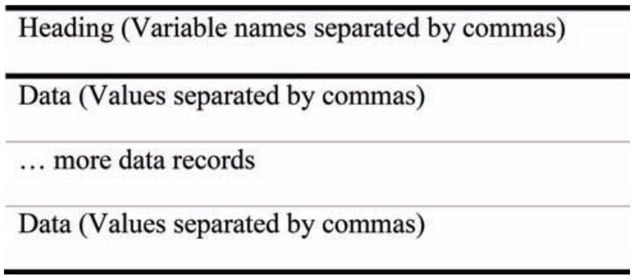
The time independent file contains the variables listed in Table C.1.

C.2 Time Dependent File

The time dependent file is in a comma separated value format. The first row contains the variable name and the following columns contain the data as shown in Figure C.1.

The time dependent file contains the variables listed in Table C.2.

The reported angle is computed as shown in Figure C.2.



Heading (Variable names separated by commas)
Data (Values separated by commas)
… more data records
Data (Values separated by commas)

Figure C.1 Organization of records in the time independent file.

TABLE C.1
Time independent variables description.

Variable Name	Description	Type
ObjectID	Unique identifier of the object	Integer
Length	Measured object length in meters	Float
Width	Measured object width in meters	Float
Height	Measured object height in meters	Float
PolygonFirst	Object centroid first polygon ID as defined as the user	Integer
PolygonLast	Object centroid last polygon ID as defined as the user	Integer
FrameFirst	First frame where the object was present	Float
FrameLast	Last frame where the object was present	Float
NbrFrames	Number of frames where the object was visible to the LiDAR(s)	Integer
ObjClassification	Classification of the object	Text
Speed75p	75th percentile of the speed in m/s	Float

TABLE C.2
Time dependent variables description.

Variable Name	Description	Type
ObjectID	Unique identifier of the object	Integer
PolyID	Polygon Id as specify by the user during data preparation	Integer
CentroidX	X coordinate of the centroid of the object in meters	Float
CentroidY	Y coordinate of the centroid of the object in meters	Float
Angle	Orientation with respect to LiDAR in degrees	Float
Speed	Instantaneous forward speed (m/s)	Float
Acceleration	Instantaneous forward acceleration (m/s^2)	Float

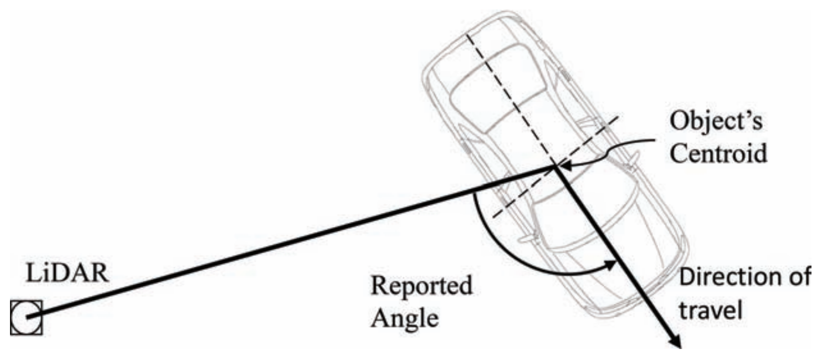


Figure C.2 Angle of object with respect to LiDAR.

APPENDIX D. TSCAN—USER MANUAL

TScan uses LiDAR technology that can detect and track various types of road users including buses, cars, pedestrians, and bicycles; and unlike video detection, LiDAR data has a one-to-one correspondence with the physical world. Hence, it is possible in principle to produce the positions and velocities of road users in real-time as needed for traffic and safety applications, with the errors of estimation dependent only on the resolution and accuracy of the LiDAR sensor.

The Initial Setup and Data Collection Module of TScan was developed by the Purdue University Center for Road Safety and includes the user interface to enter the required information and to set up the traffic scanner for data collection at a given intersection. TScan was developed as part of the project called "Stationary LiDAR for Traffic and Safety Applications – Vehicles Interpretation and Tracking (TScan)" supported through the Joint Transportation Research Program of the Indiana Department of Transportation and Purdue University.

D.1 TScan Overview

There are two major components of TScan:

(1) The System Preparation by User
(2) Data Collection Module

The System Preparation by User component consists of two steps: a) off-site location data preparation and b) on-site sensor alignment.

TScan is a computer application that allows the user to enter the required information prior to data collection and to set up the traffic scanner and collect data at a given intersection. Figure D.1 shows the TScan process overview.

D.1.1 Off-Site Process

The off-site process aims to allow the user to enter, save, and retrieve the intersection's characteristics in a graphic environment to transfer the information to TScan. This process includes selecting and uploading an orthographic-image, drawing polylines to create uniform polygons, and entering characteristics such as type, potential users, and type of maneuver of each polygon.

Table D.1 shows the supported operations for the various polygon creation-related features for generating polylines, points and polygons.

D.1.2 On-Site Process

The on-site process aims to set a common coordinate system for the TScan sensor and the orthographic-image in order to transfer the polygon characteristics to the TScan processing module. A typical workflow

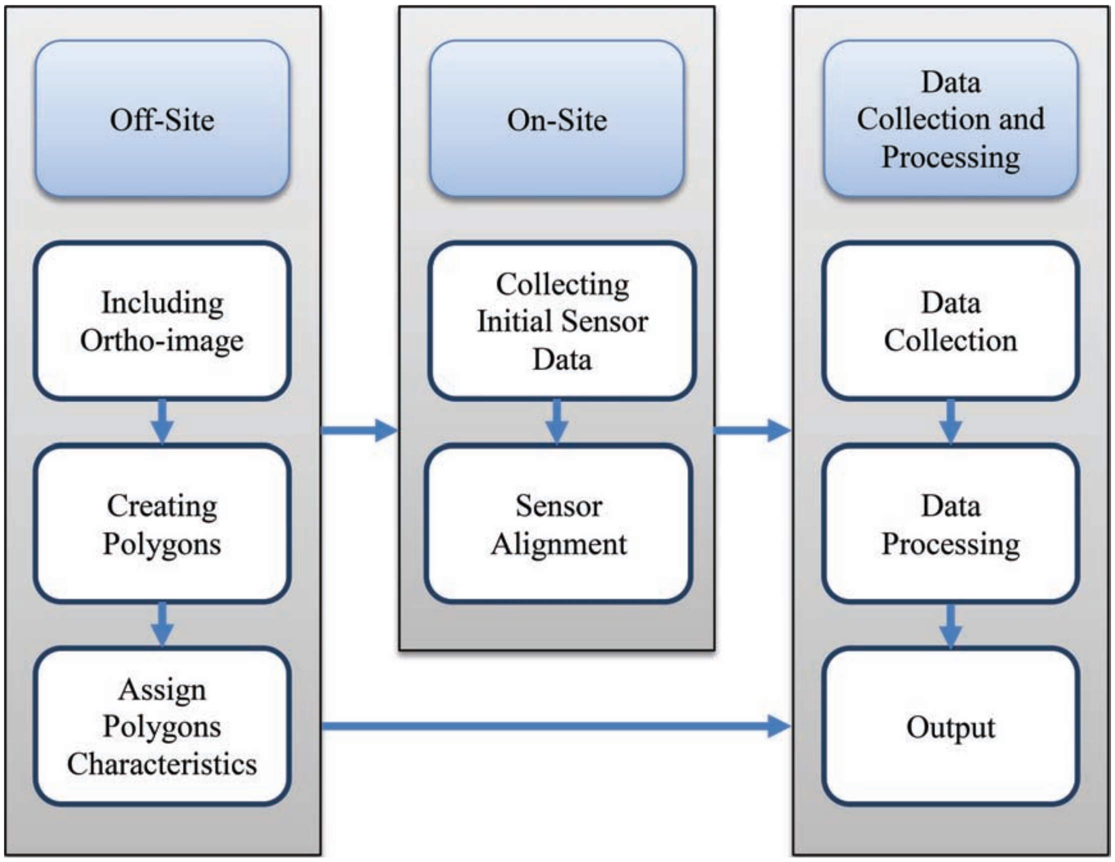


Figure D.1 TScan process overview.

TABLE D.1
List of supported operations for various polygon-related features.

Feature	Supported Operations
Polylines	• Add • Delete • Copy • Move • Trim • Extend
Points	• Insert • Add • Remove
Polygons	• Connect • Select

exercised by the user on-site can be summarized as follows:

- Information created off-site is retrieved and visualized.
- Initial sensor data is collected.
- Sensor data is visualized side by side with the orthographic-image.
- Two specific features are selected that are present in both the orthographic-image and the LiDAR sensor data.
- The application's inbuilt alignment function is used to properly align the two.
- Alignment parameters are further adjusted manually.
- The final alignment information is saved and exported to Data collection and processing module.

D.1.3 Data Collection and Processing

After the setup information is obtained, the Data Collection and Processing Module gathers information from the sensor, processes it, and produces useful information for future engineering applications such as the following:

- Traffic counts
- Traffic signals warrants
- Stop signs warrants
- Speed analysis
- Gap acceptance analysis
- Saturation flows measurement
- Pedestrians studies
- Red signal violations (signals data input)
- Change periods evaluation
- Traffic conflicts frequency studies
- Traffic conflicts diagrams (modified Collisions Diagram Builder)

The Data Collection and Processing Module DOES NOT REQUIRE any user intervention while collecting data.

D.2 Installation

TScan is compatible with Windows 7/8/10. In order to run TScan, the MS .NET 4.0 Framework or later component must be installed.

If the MS .NET 4.0 Framework is not present during the installation, TScan will attempt to install this component if the PC is connected to the Internet.

To install the TScan interface, follow the steps given below:

1. Extract the contents of the archived file to your local drive.
2. The installation process is initiated by clicking on the *setup.exe* file.
3. Step by step instructions that explains the installation process is found in *readme.exe*.

Note: The user should always read the readme.txt file included in the installation package which includes the most up-to-date installation instructions.

After checking that the program is working, the user may delete the unzipped files in the folder with the *setup.exe* file to save disk space. The user should save the zipped/compressed file in case it is needed to reinstall the program.

D.3 Launching TScan

The TScan program can be launched using any of the following methods after installation:

Method 1:

1. Double click the shortcut on the desktop.

Method 2:

1. Press Start button.
2. TScan should appear in the list of installed programs.
3. Single click on the shortcut.

Method 3:

1. Press Start button.
2. Start typing "TScan," the program shortcut should appear in the search results.
3. Single click on the shortcut.

Method 4:

1. Open My Computer.
2. Browse to the location where the software was installed. Typically C:\Program Files (x86)/TScan/.
3. Click on TScan.exe.

The main interface appears within a few seconds (Figure D.2).

D.3.1 TScan Interface

The TScan interface includes a menu bar in the first row and a command bar below (Figure D.3). The menu bar facilitates the operations on files and the view layout while the command bar, which is below the menu bar, facilitates the operations on files, polylines, points, polygons, and LiDAR alignment.

The command bar is movable and can be snapped on to any of the four window edges. The bar can be moved by simply dragging it around.

The TScan interface allows users to perform system preparation for data collection for both off-site and on-site data preparation, collection, and processing. The following sections describe each activity in detail.

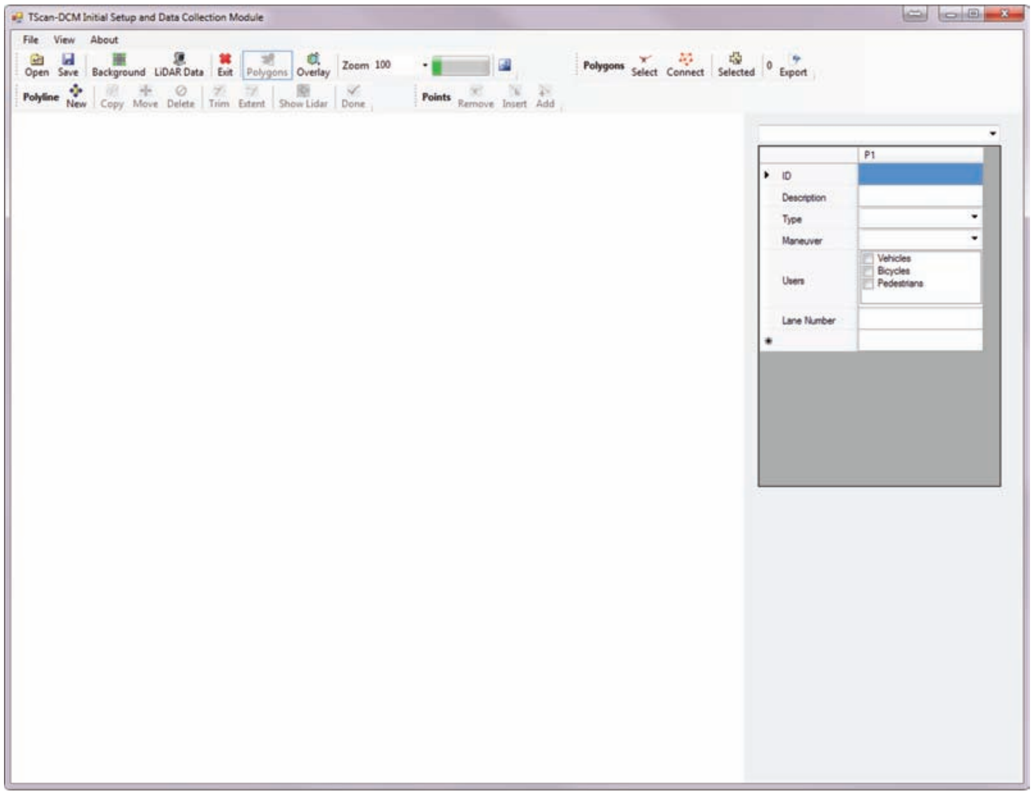


Figure D.2 TScan main interface.

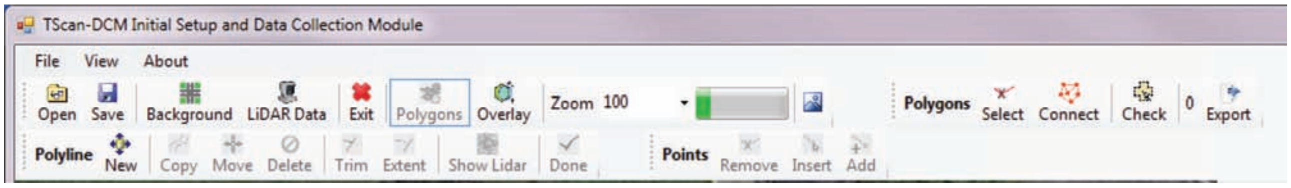


Figure D.3 Menu and command bars in the TScan interface.

D.3.2 Files

TScan requires only an orthographic-image of the intersection in which the data collection will be performed. It generates a series of files that are used for the next steps and needs to be transferred to the next activities. Table D.2 shows the list of files used by TScan.

D.4 Off-Site Process

The off-site process allows the user to enter, save, and retrieve the intersection's characteristics in a graphic environment to be used by TScan and other engineering applications.

The information required by TScan is a list of polygons that defines the road lanes, intersection areas, parking areas, sidewalks, and medians along with the corresponding potential road users and intended maneuver(s). In order to create the required polygons, the user should upload an orthographic-image that will be used to draw the polygon edges and also later used in other engineering applications.

Although it is not necessary to create new folders, it is advisable to do so in order to have all the study information regarding a particular site in a single folder.

The following section explains how to upload the orthographic-image, draw the polygon edges and generate the polygon list.

D.4.1 Background Image

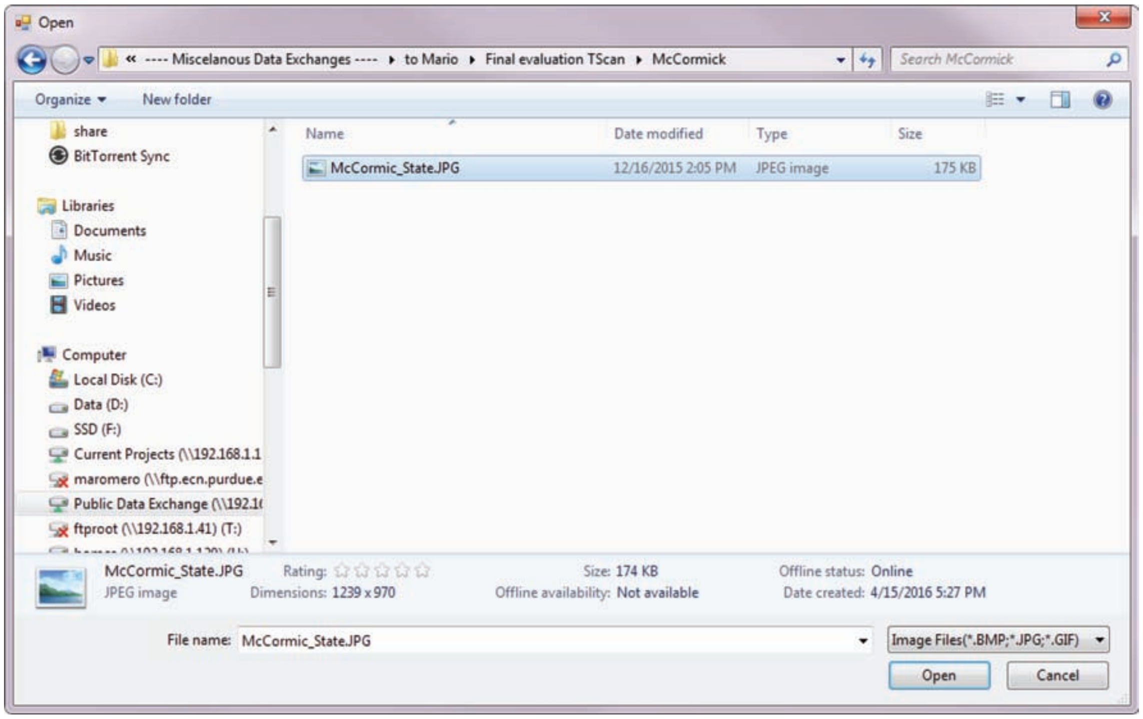
The first step is to copy an orthographic-image, obtained from sources like Google Earth or ArcGIS, into the study folder. Then, the orthographic-image is uploaded as a reference to create the layout. The orthographic-image zoom factor can be adjusted to better display the background.

Upload background image. Once the image file is in the desired folder, click on the *Background* button to open the Windows file selection window shown in Figure D.4, where the appropriate file can be selected. Once the file is selected, click on the *Open* button to upload the image.

TABLE D.2
TScan list of files.

File	Format	Description
<Orthographic-image>	Any image format such as bmp, jpg, jpeg, gif, png, tif, etc.	User must provide a top view image of the intersection saved in image format. Tis file is used as a reference to mark entry lanes, exit lanes, intersection area, parking areas, medians and sidewalks.
<Location Name>.dsc	Text format with extension dcs	TScan generated file that contains all user information regarding the data collection settings using the orthographic-image coordinates system.
<Location Name>.pd1	Text format with extension pd1	TScan generated file that contains polygon information.
<Location Name>.ppd	Text format with extension ppd	TScan generated file that contains the list of points for each polygon in the LiDAR coordinates system.
<Location Name>.trj	Binary file in SSAM* trajectory file format	TScan output file that contains the location, speed and acceleration of each vehicle every 0.1 sec. For more information visit: http://www.fhwa.dot.gov/publications/research/safety/08049/
<Location Name>-Timedependent.csv	Comma separated value	TScan output file that contains the location, speed and acceleration of each vehicle every 0.1 sec.
<Location Name>-TimeIndependent.csv	Comma separated value	TScan output file that contains the object characteristics. Each object can be linked to the time dependent data by an object ID.

*SSAM: Surrogate Safety Assessment Model.


Open
« ---- Miscelanous Data Exchanges ---- ▸ to Mario ▸ Final evaluation TScan ▸ McCormick
Search McCormick
Organize ▾
New folder
share
BitTorrent Sync
Libraries
Documents
Music
Pictures
Videos
Computer
Local Disk (C:)
Data (D:)
SSD (F:)
Current Projects (\\192.168.1.1
maromero (\\ftp.ecn.purdue.e
Public Data Exchange (\\192.1(
ftproot (\\192.168.1.41) (T:)
Open
Cancel

Name	Date modified	Type	Size
McCormic_State.JPG	12/16/2015 2:05 PM	JPEG image	175 KB

McCormic_State.JPG JPEG image Rating: Dimensions: 1239 x 970 Size: 174 KB Offline availability: Not available Offline status: Online Date created: 4/15/2016 5:27 PM

File name: McCormic_State.JPG Image Files(*.BMP;*.JPG;*.GIF)

Figure D.4 Upload orthographic-image.

Adjust zoom factor. Once the image is uploaded the scale factor can be adjusted to better display the background by selecting one of the existing zoom factors as shown in Figure D.5.

D.4.2 Intersection Layout

TScan requires the intersection layout in order to better estimate the background. Therefore, the user must define the edges of lanes, paved areas, and intersection areas. An easy to use set of drawing tools are provided to allow the user to graphically define such edges. There are two main command bar tools that are used for this purpose: the polyline command bar and the point command bar.

Polyline command bar. The basic element type to create the intersection layout is a polyline. A polyline is a continuous line composed of one or more line segments.

The polyline command bar tool includes functions for adding, deleting, coping, moving, trimming and, extending polylines as shown in Figure D.6.

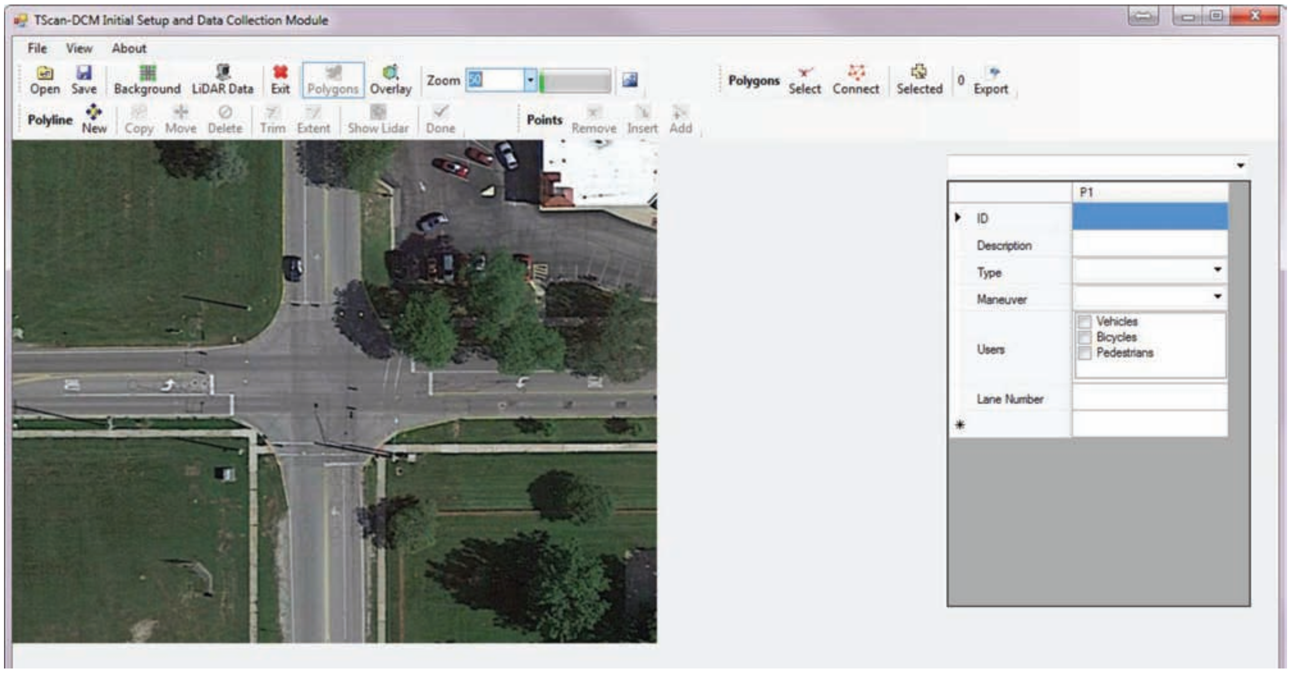


Figure D.5 Adjust background zoom factor.

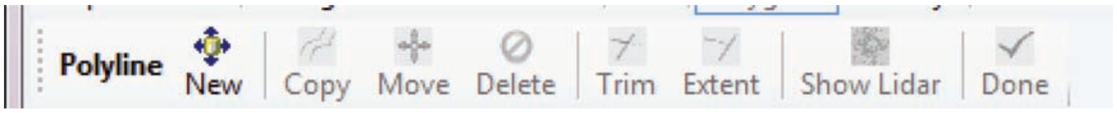


Figure D.6 Polyline command bar tools.

Add polylines. To create the Intersection layout, the first step is to add polylines. You can create a polyline by clicking on the *Add* button and then specifying the endpoints of each segment by clicking on the orthographic-image. Once all points are created, click on *Done* button or right-click on the image to finalize adding points to the polyline. The end points of the new polyline will be hidden after finish adding segments. Figure D.7 shows an example of adding polylines.

It is important to highlight that one single polyline should not have closed areas as shown in Figure D.8. It is also important that user must guarantee that two consecutive polylines intersects each other to later create polygons.

Delete polylines. To delete a polyline, it is necessary to select the polyline by clicking on it. After selecting, the end points are displayed. Then, click on the *Delete* button to remove it from the image. Figure D.9 shows an example of deleting polylines.

Copy polylines. To create several parallel polylines, it is easier to select the polylines to be copied by clicking on them, which displays the end points on the screen. Then, click on the *Copy* button and click on the image where the new polyline starts. Repeat clicking on the image to copy the polyline several times, then click *Done* or right-click on the image to end copying polylines. Figure D.10 shows an example of coping polylines.

Move polylines. The first step to move a polyline is to select it by clicking on it, which displays the end points on the screen. Then, click on *Move*. Then, maintain pressing the mouse left button and move the polyline to the desire location, and release the mouse left button when finished. Click on *Done* or right-click on the image to finalize moving the polyline. Figure D.11 shows an example of moving polylines.

Trim polylines. To trim a polyline, first select the base polyline, which is the polyline that will not change its dimension by clicking on it. After selecting, the end points are displayed on the screen. Click on the *Trim* button, and the selected base polyline will change to yellow in color. Then, click on an intersecting polyline on the side that should be removed. Repeat this operation to trim the other intersecting polylines. Finally, click on *Done* or right-click on the image to finalize trimming the polylines. Figure D.12 shows an example of trimming polylines.

Extend polylines. To extend a polyline it is necessary to select first the base polyline which is the one that will not change its dimension by clicking on it. After selecting, the end points are displayed. Then, click on the *Extend* button. The selected base polyline will change to yellow. Then click on the non-intersecting polyline on the side that should be extended. Repeat this operation to extend other non-intersecting polylines. Finally, click on *Done* button or right-click on the image to finalize extending the polylines. Figure D.13 shows an example of trimming polylines.

Point command bar tool. Another option to edit the polylines is to manipulate their endpoints. The point

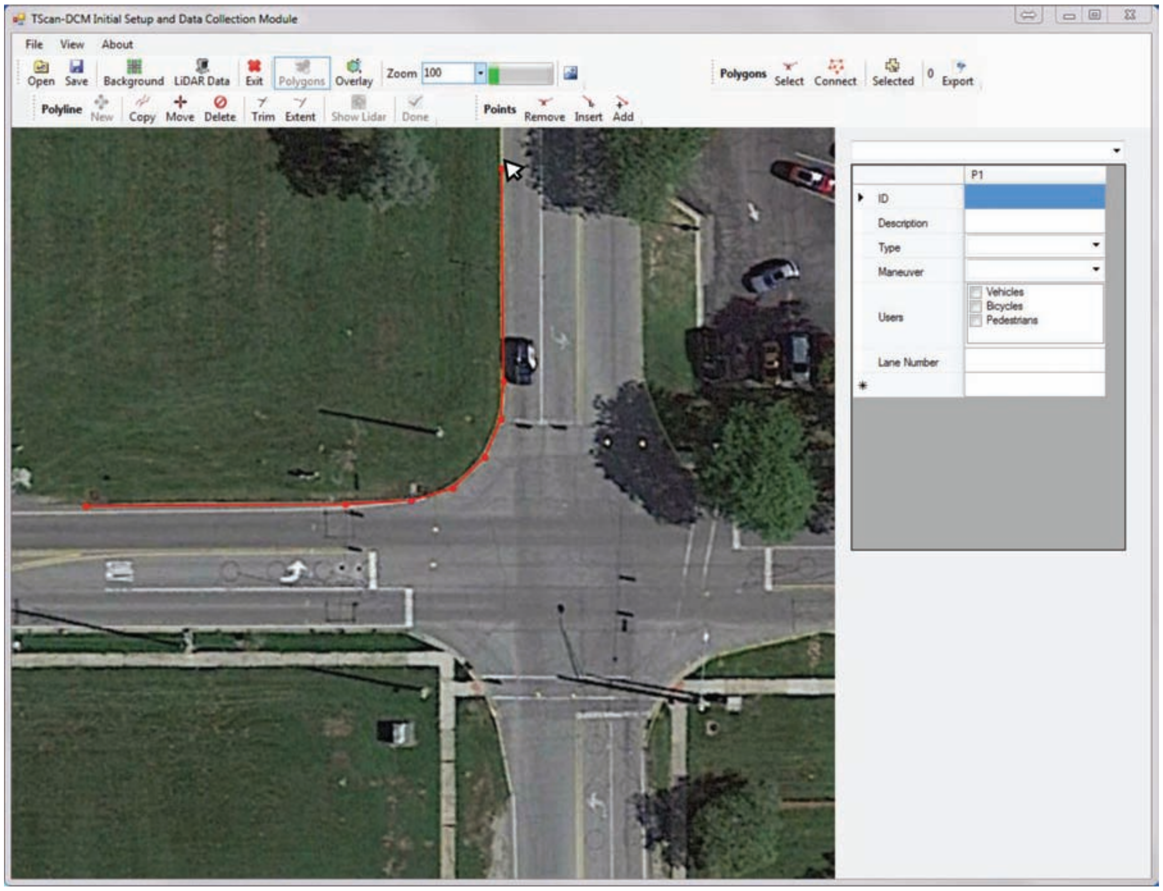


Figure D.7 Adding polylines.

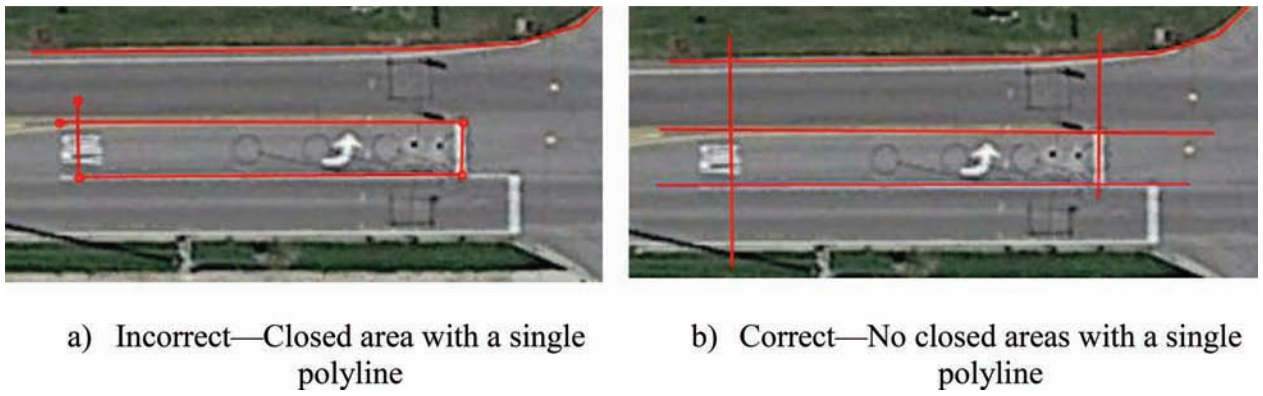

a) Incorrect—Closed area with a single polyline
b) Correct—No closed areas with a single polyline

Figure D.8 Avoid closed areas with a single polyline and ensure polylines intersect.

command bar tools include functions for editing, inserting, adding and removing the polyline's endpoints.

Edit endpoints. To edit the endpoints, first select the polyline by clicking on it. After selecting, the end points are displayed on the screen. To change the endpoint's location, simply drag it to the new position and release the mouse button. Once the endpoint's locations are correct, click on *Done* or right-click on the image to finalize editing the endpoints. Figure D.14 shows an example of moving polylines.

Insert endpoints. To insert endpoints, first select the polyline by clicking on it. After selecting, the end points are displayed on the screen. Click on *Insert* to activate the function. Then, click over the polyline in the positions where the new endpoints should be added. To finalize adding endpoints, click on *Done* or right-click on the image. Figure D.15 shows an example of inserting endpoints.

Add endpoints. The difference between inserting and adding endpoints is that inserting adds points between

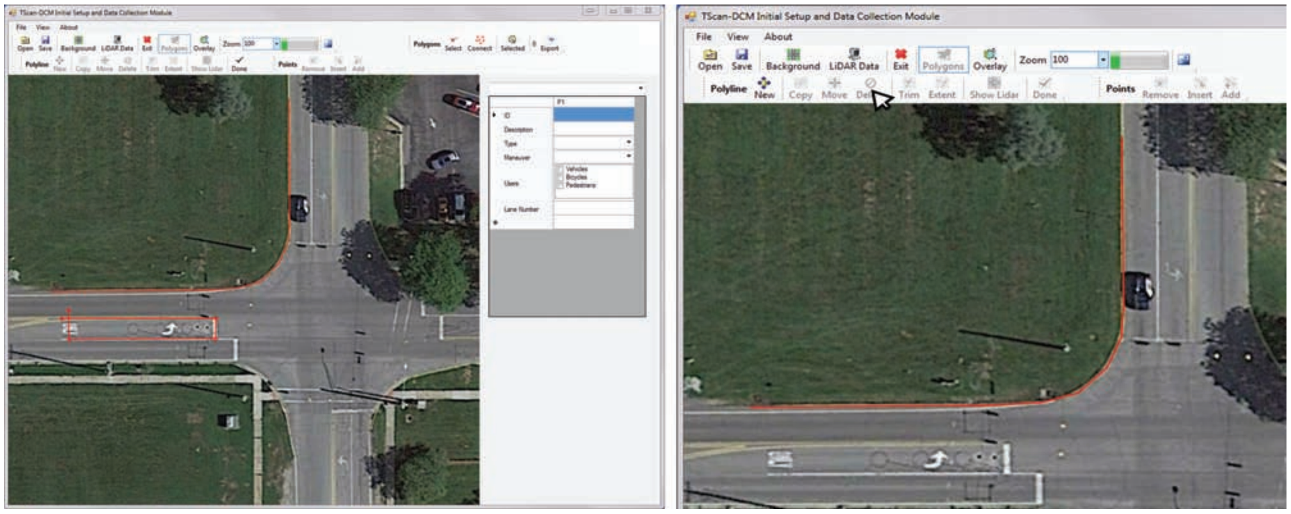


Figure D.9 Deleting polylines.

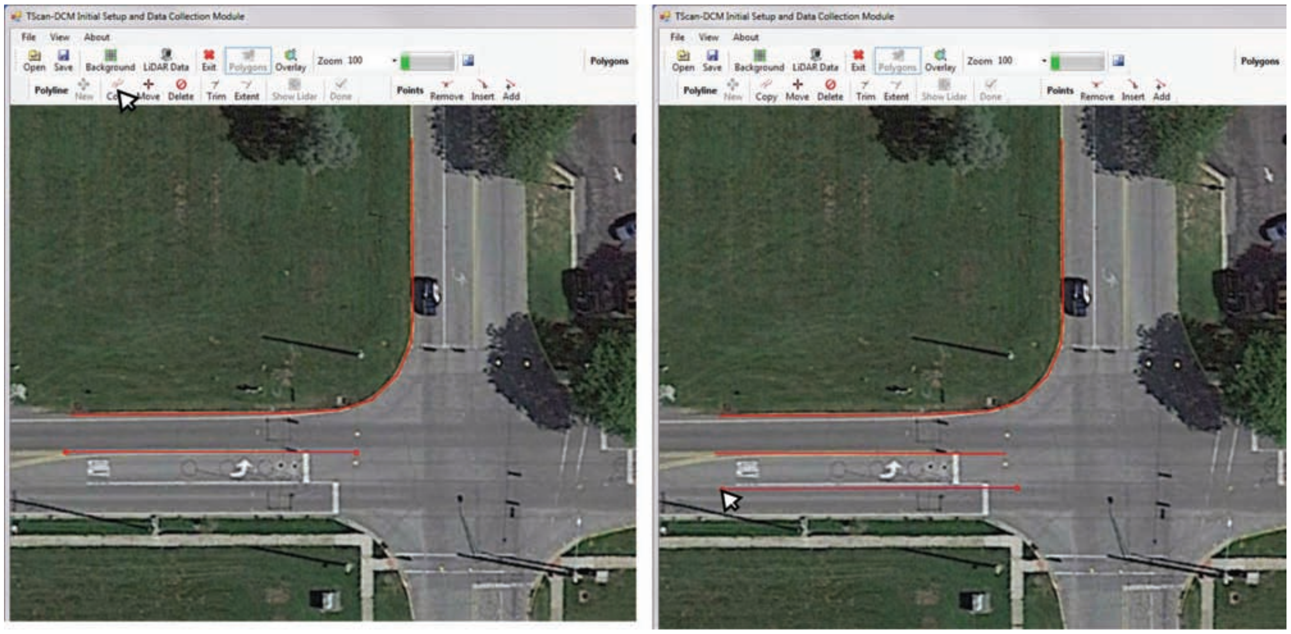


Figure D.10 Coping polylines.

two consecutive endpoints while adding endpoints creates an extra end point at the beginning or the end of the polyline. To add endpoints, select the polyline by clicking on it. After selecting, the end points are displayed on the screen. Click on *Add* to activate the function.

To create a new endpoint at the beginning of the polyline, left click on the location of the new endpoint. To create a new endpoint at the end of the polyline, right click on the location of the new endpoint. To finalize adding endpoints, click on *Done*. Figure D.16 shows an example of adding endpoints.

Remove endpoints. To remove endpoints, select the polyline by clicking on it. After selecting, the end points are displayed on the screen. Click on *Remove* to activate the function. Click on the endpoint to be removed from the selected polyline. Figure D.17 shows an example of removing endpoints.

D.4.3 Creating Polygons

Once the intersection layout is completed, the next step is to create the polygons and set their characteristics.

The polygon command bar tools allow the user to connect the polylines and to select/create polygons.

Connect polylines. Use this function to trim all segments of the polylines and to add endpoints at the intersection of any segment pairs. This command will also remove all unconnected segments. Figure D.18 shows the result of the connect function.

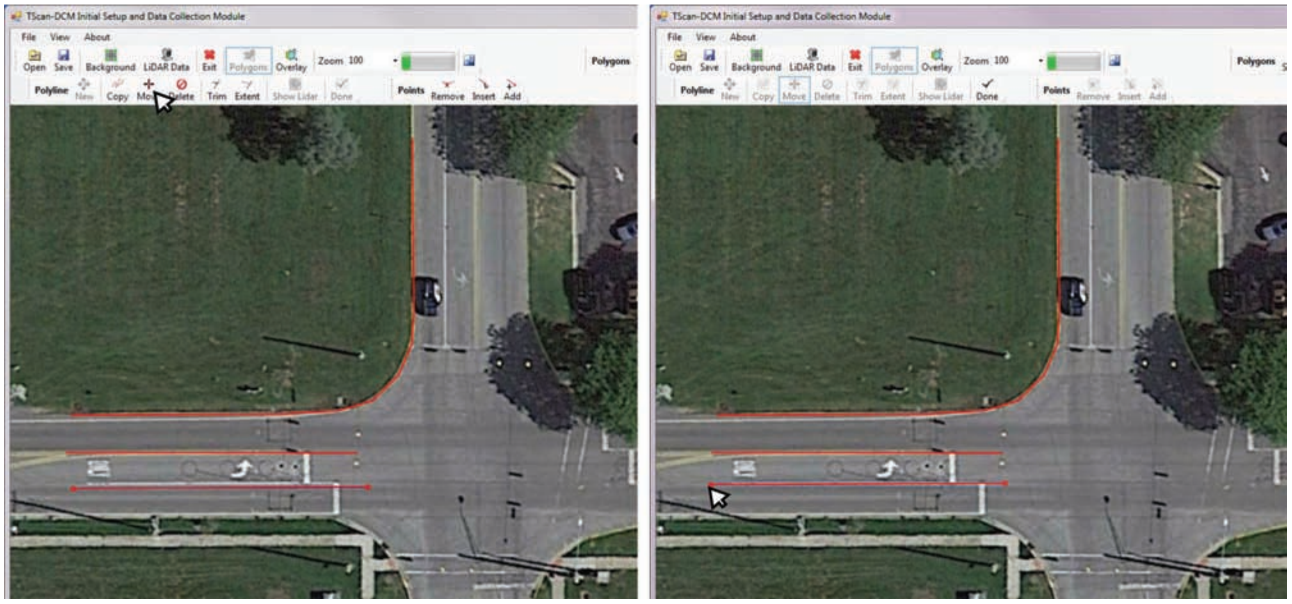

TScan-DCM Initial Setup and Data Collection Module
File View About
Open Save Background LiDAR Data Exit Polygons Overlay Zoom 100
Polygons
Polyline New Copy Move Delete Trim Extent Show Lidar Done
Points Remove Insert Add

Figure D.11 Moving polylines.

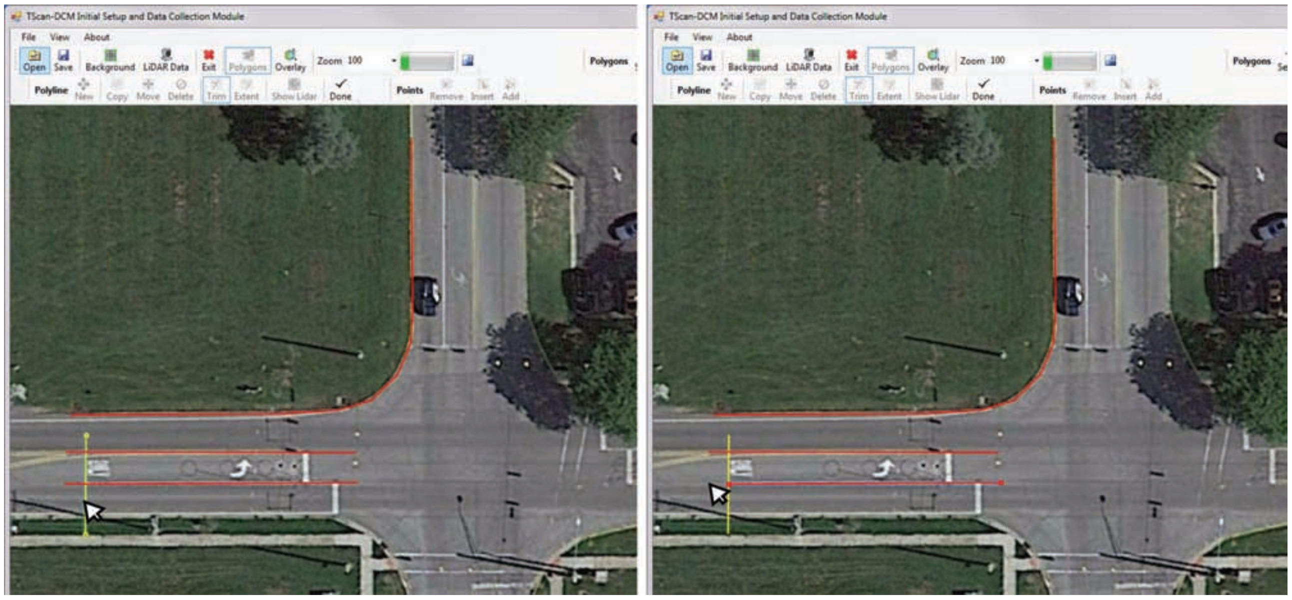

TScan-DCM Initial Setup and Data Collection Module
File View About
Open Save Background LiDAR Data Exit Polygons Overlay Zoom 100
Polygons
Polyline New Copy Move Delete Trim Extent Show Lidar Done
Points Remove Insert Add

Figure D.12 Trimming polylines.

Select polygons. Use this function to activate the function that selects/creates polygons in the intersection layout. Once the function is activated, click on the image inside the area to create a polygon or to select it. The polygon area will be highlighted. If the polygon already exists, the polygon information will be displayed, otherwise the polygon is created and a continuous id is assigned.

Once polygons are created, they also can be selected by using the polygon list box. Note that only polygons in which the user has clicked are listed.

Click on *Done* or right-click on the image to finalize selecting polygons. Figure D.19 shows the result of select polygons.

Check polygons. Use this function to highlight all created polygons. Figure D.20 shows the result of check polygons.

Set polygons characteristics. Once all polygons are created, the polygon characteristics must be set. To do so, select a polygon and change the polygon characteristics: ID, description, select the polygon type, the type of maneuver, expected user and lane number as shown in Figure D.21. Then continue with the next polygon until all polygons are selected.

Make sure all polygons have set the polygons type because those ones without polygon type will not be used to track objects.

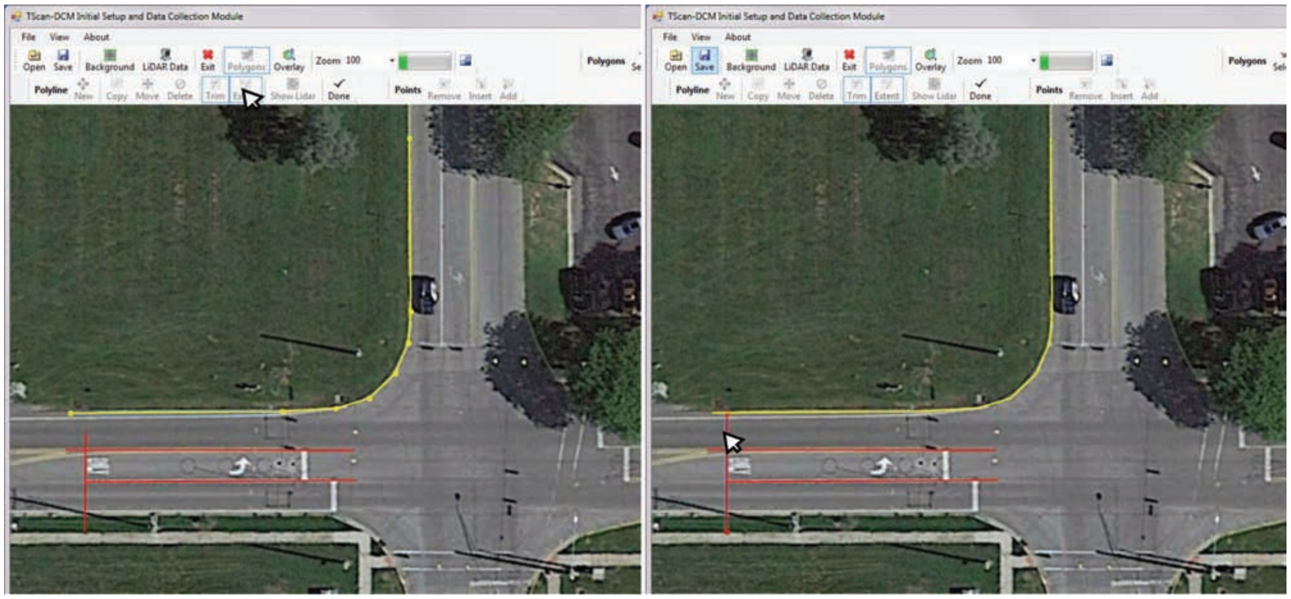

TScan-DCM Initial Setup and Data Collection Module
File View About
Open Save Background LiDAR Data Exit Polygons Overlay Zoom 100
Polyline New Copy Move Delete Trim Extent Show Lidar Done
Points Remove Insert Add
Polygons

Figure D.13 Extending polylines.

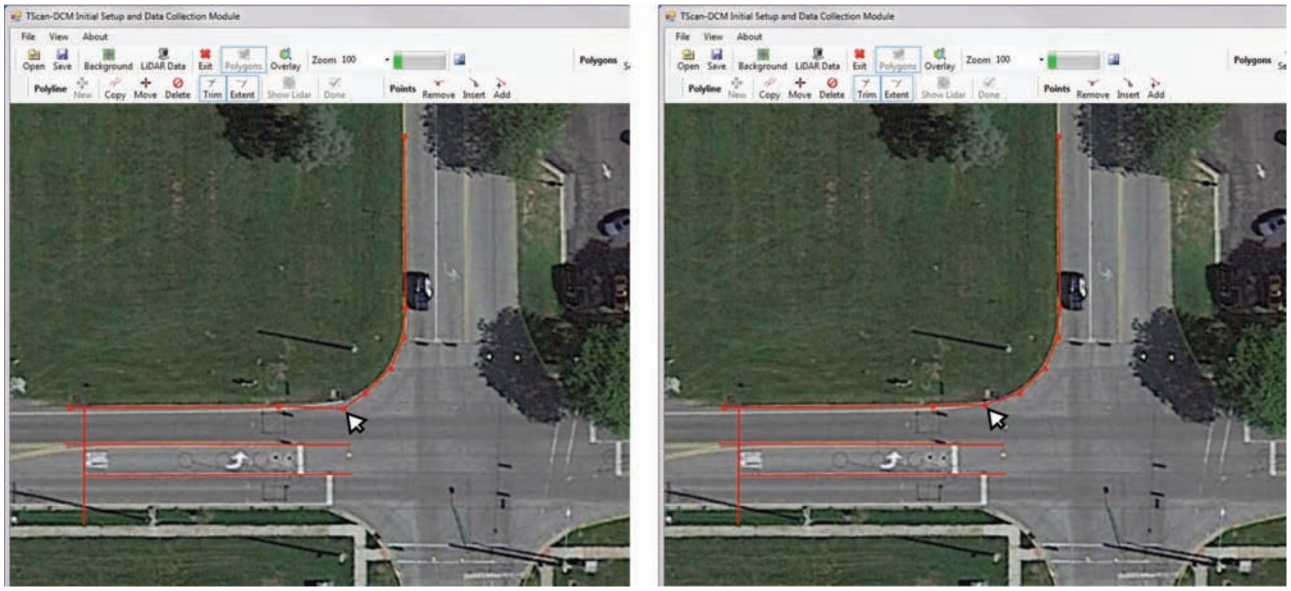

TScan-DCM Initial Setup and Data Collection Module
File View About
Open Save Background LiDAR Data Exit Polygons Overlay Zoom 100
Polyline New Copy Move Delete Trim Extent Show Lidar Done
Points Remove Insert Add
Polygons

Figure D.14 Editing endpoints.

D.4.4 Saving and Retrieving Layouts

Saving layouts. After clicking on the *Save* button, the user is asked to provide a file name and a folder to which the layout is to be saved (Figure D.22). The layout creates a .dcs file that contains all user entry data and the orthographic-image path.

Retrieving layouts. After clicking on *Open*, the user is asked to select the file to be opened as shown in Figure D.23. If the orthographic-image path is not found, it only retrieves the polylines and the polygon information so the user should open the background image as well.

D.5 On-Site Process

The on-site process aims to set a common reference system for the TScan sensor and the orthographic-image in order to transfer the polygon characteristics to the TScan processing module.

The information created off-site is retrieved and displayed, then an initial sensor data is collected and displayed side by side with the orthographic-image as shown in Figure D.24. Then, the interface allow user to select two common points in both the orthographic-image and the TScan points in order to properly align them and to perform further adjustments. Once the user is satisfied with the alignment the information is saved and exported

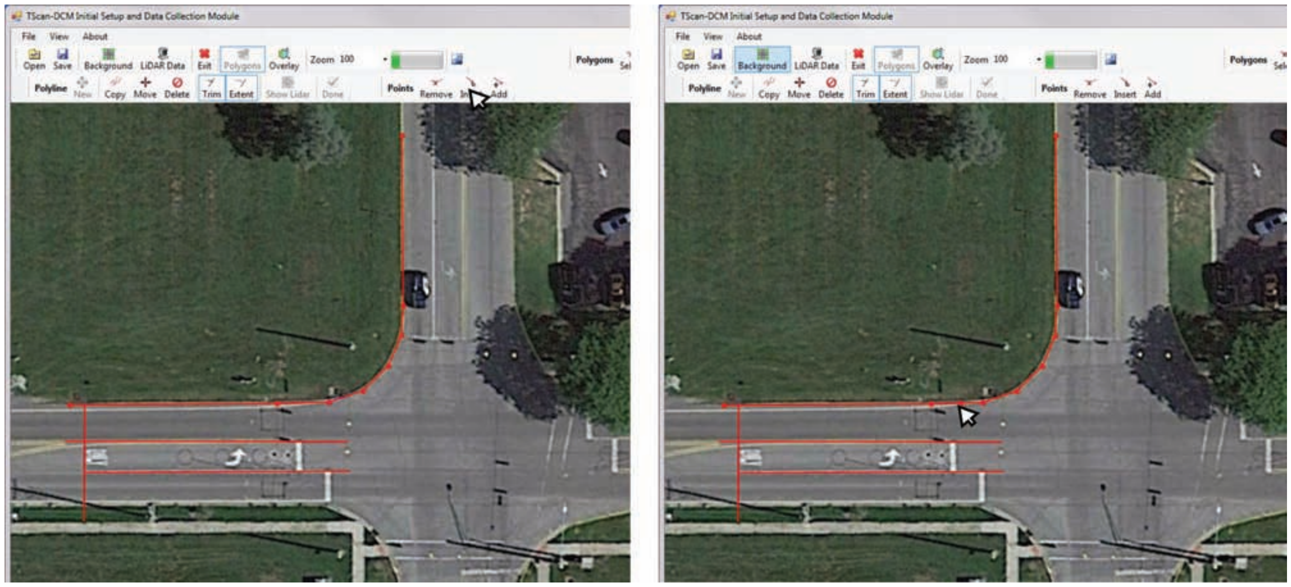


Figure D.15 Inserting endpoints.

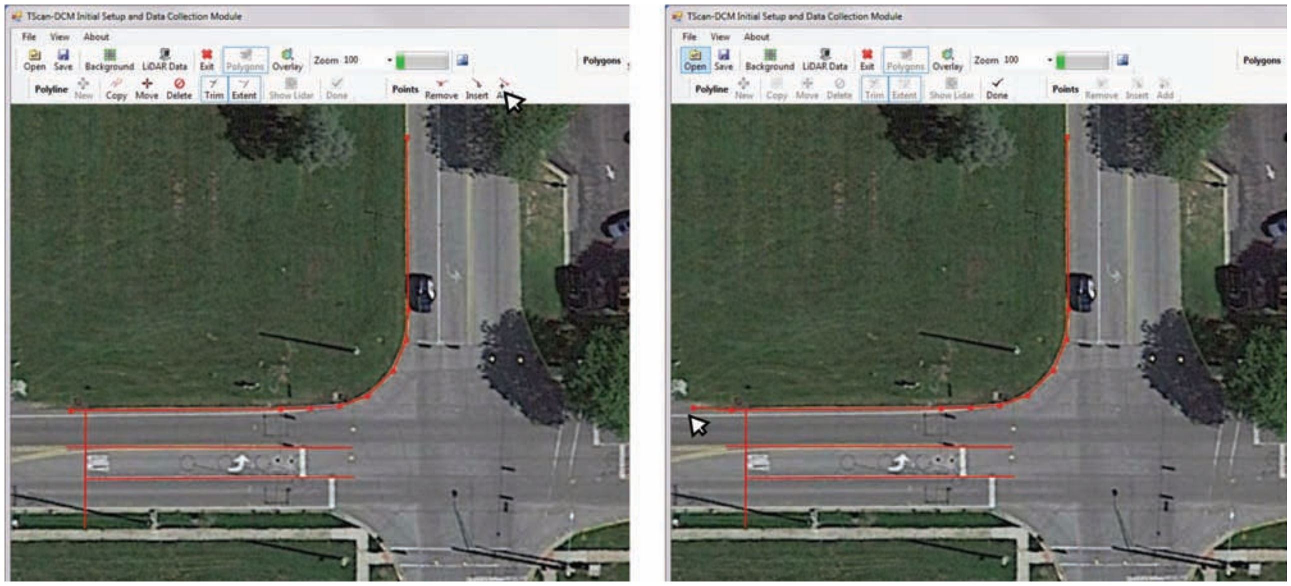


Figure D.16 Adding endpoints.

to the Data collection and Processing Module. The orientation of the LiDAR should not be changed after this point. This includes rising and lower the mast, changing the Pan-Tilt base setting or leveling the unit.

D.5.1 Initial Sensor Data

By clicking on the *LiDAR Data* button, few frames from the TScan sensor are collected. Then click on the *Overlay* button to display both the orthographic-image and the TScan information side by side as shown in Figure D.24.

D.5.2 LiDAR Alignment

The orthographic-image and the TScan sensor must have a common reference system therefore aligning them is necessary. The initial alignment can be performed semi-automatically based on user defined reference points. Then fine manual adjustments can (also) be made.

Semi-automatic alignment. The initial alignment is based on two common points in both the orthographic-image and the TScan sensor data.

Alignment references. Add a polyline with only two points in both the orthographic-image and the TScan data and click on *Done* or right-click on the image. The first point in the orthographic-image must correspond to the first point in the TScan; data and similarly, the second one also should correspond as shown in Figure D.25.

Align. Once the reference points are created, the semi-automatic alignment is completed after clicking on

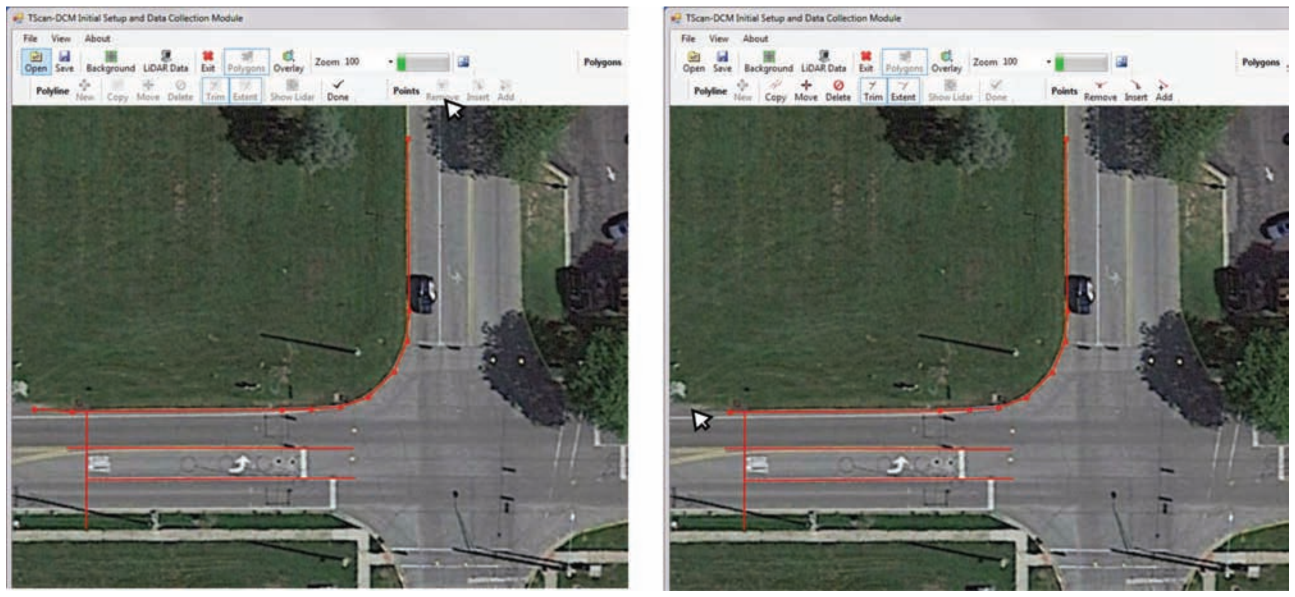


Figure D.17 Removing endpoints.

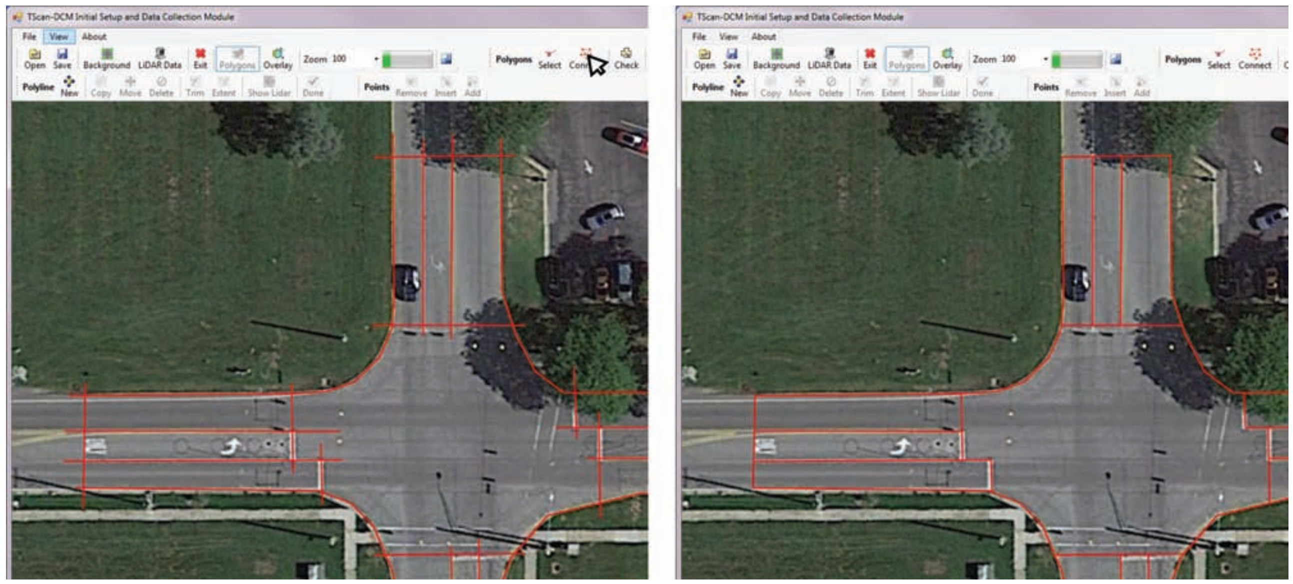


Figure D.18 Connect function.

Align to overlay the orthographic-image and the TScan data as shown in Figure D.26. It also updates the alignment parameters for rotation, scale, and vertical and horizontal shift.

Manual alignment (adjustments). The alignment can be adjusted manually by modifying the four alignment parameters: rotation, scale, vertical shift, and horizontal shift. The alignment command bar provides tools to adjust the parameters, but they also can be modified by typing them in their corresponding boxes.

Rotation adjustment. Click on *Rotate* to adjust the rotation angle by 0.1 degrees or right click to adjust the rotation angle by 0.01 degrees with the location of the click determining the rotation direction. The overlay will be updated after clicking as shown in Figure D.27. Press and hold the mouse button to repeat the operation 5 times per second and release it to stop.

Scale adjustment. Click on *Scale* to adjust the scale factor between the orthographic-image and the TScan data by 0.01 or right click on *Scale* to adjust the scale factor between the orthographic-image and the TScan data by 0.001 with the location of the click determining if the scale will be increased or decreased. The new overlay will be updated after clicking as shown in Figure D.28. Press and hold the left mouse button to repeat the operation 5 times per second and release it to stop.

Horizontal and vertical shift adjustment. Click on *Shift* to adjust the horizontal or vertical shift by 0.1%

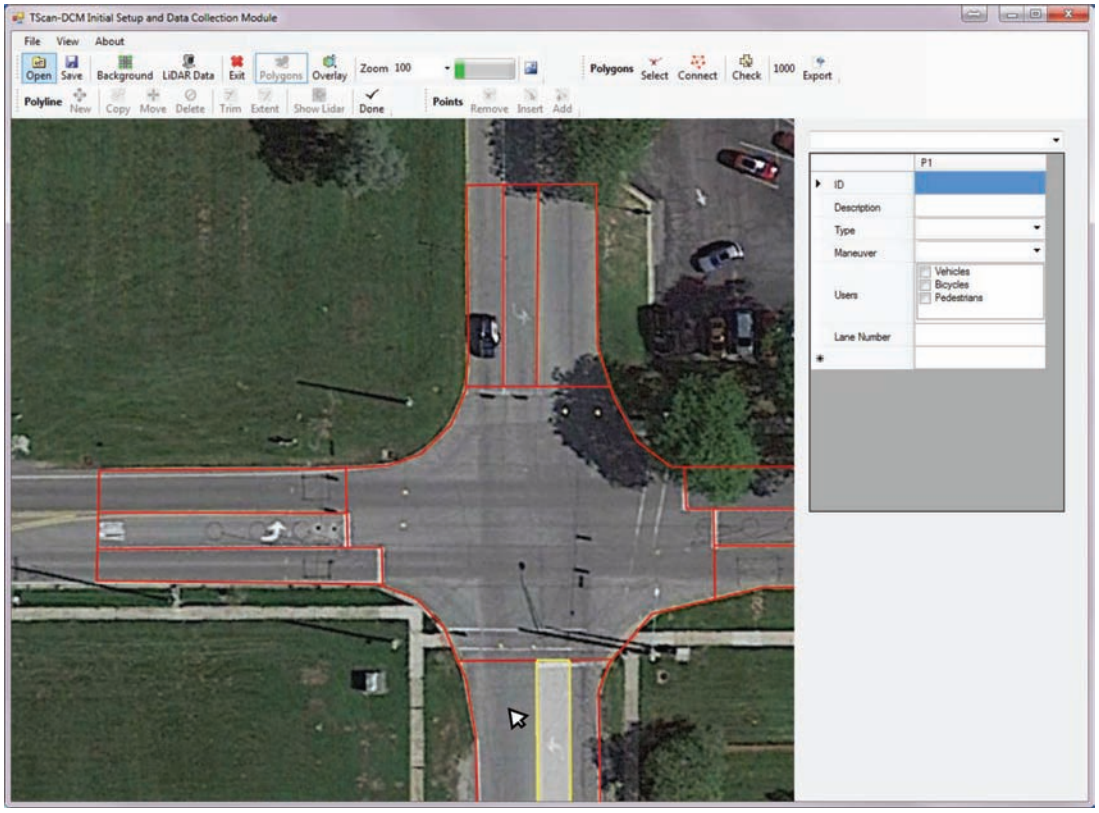


Figure D.19 Select/create polygons.

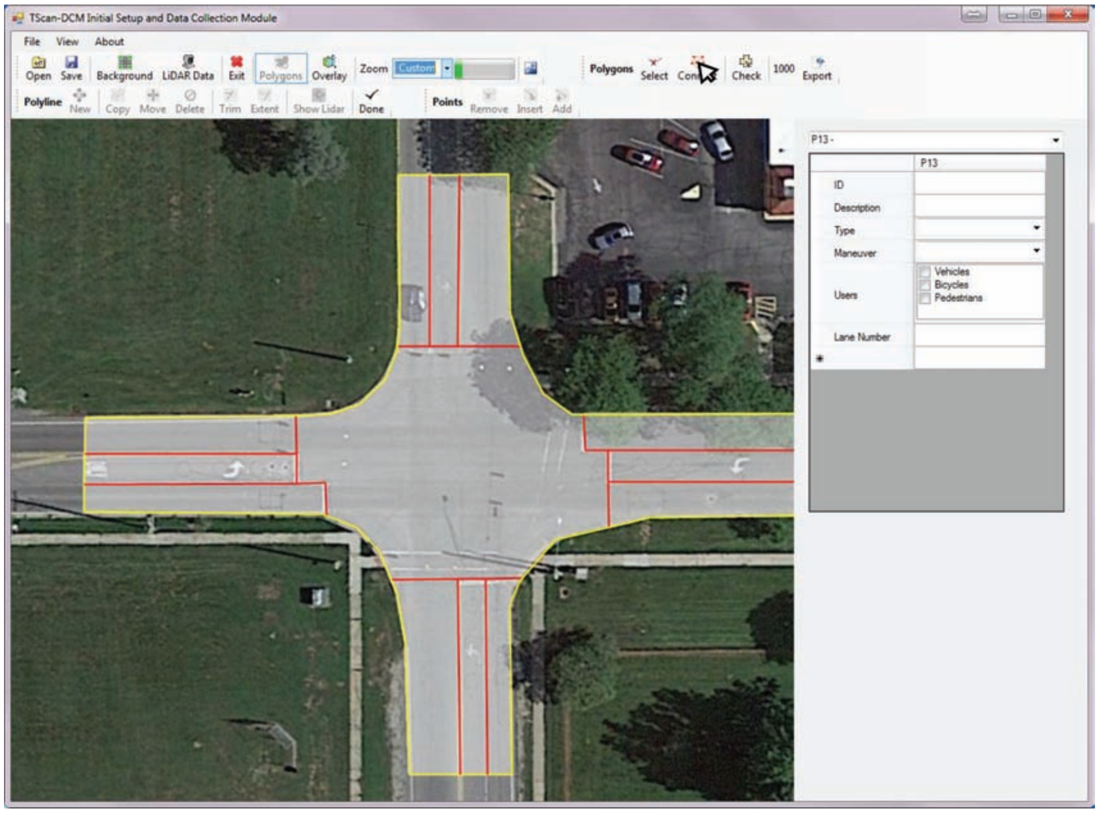
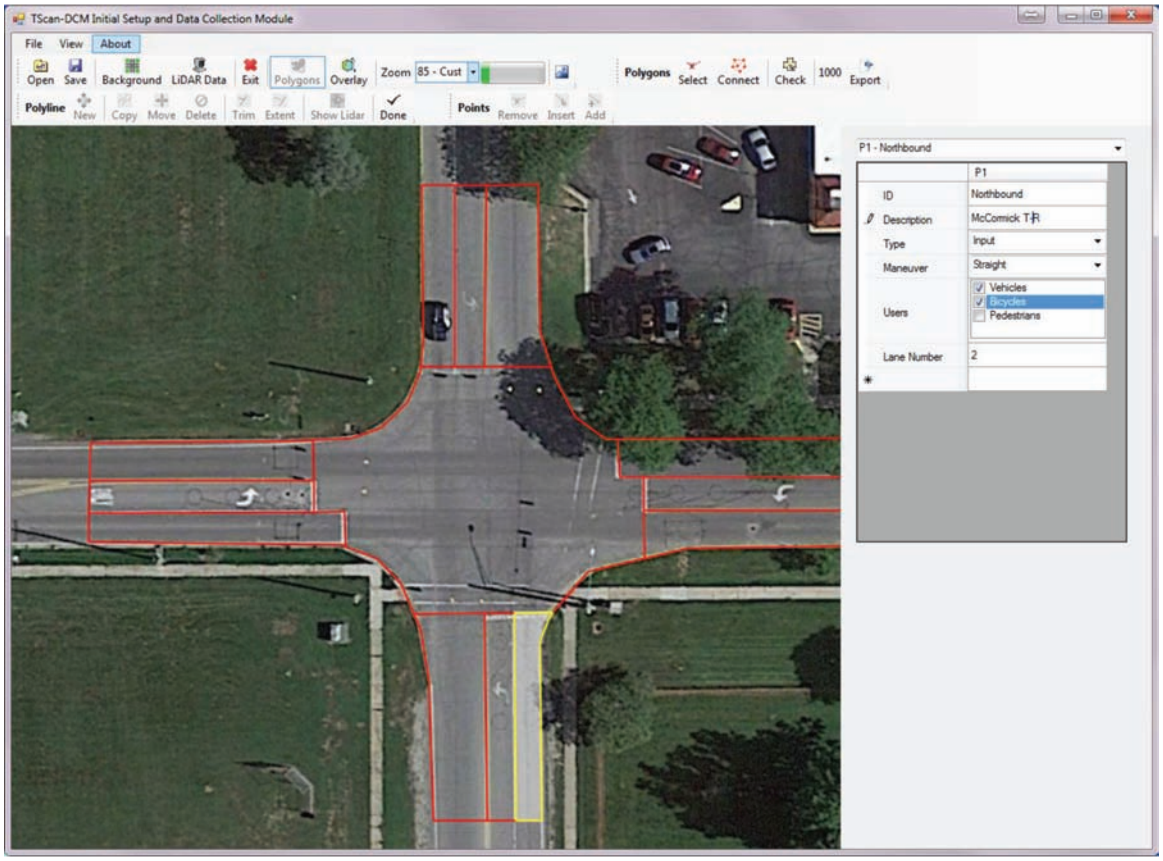


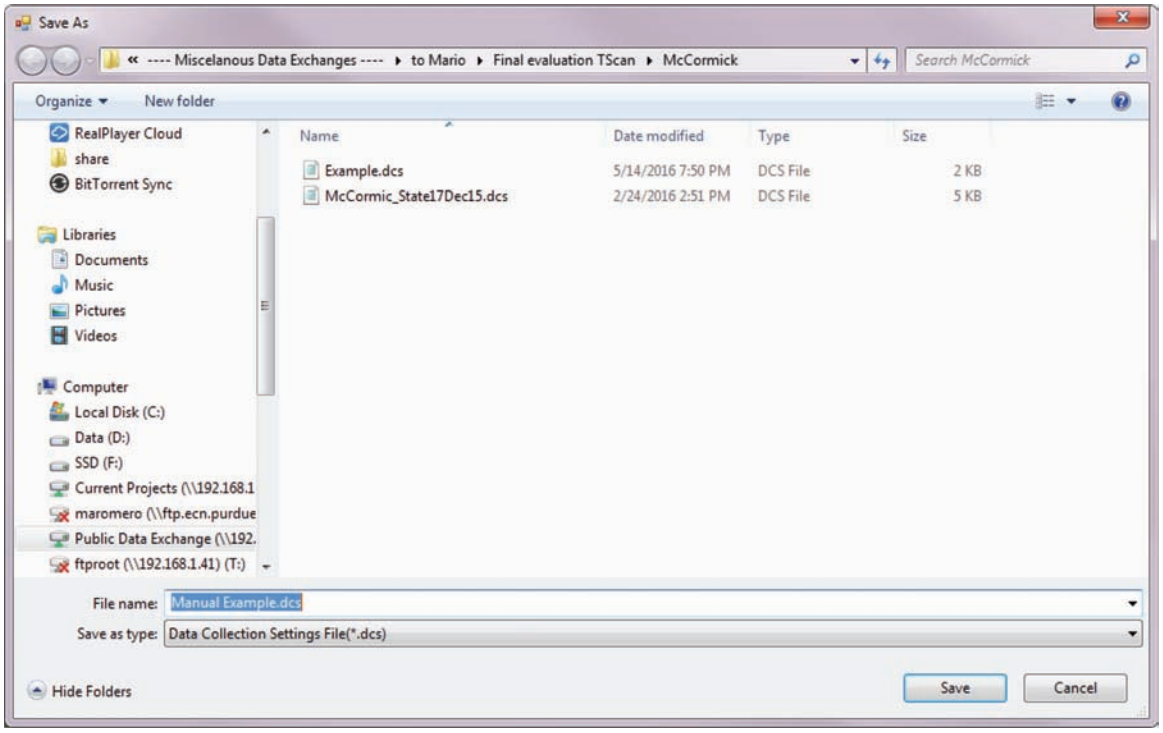
Figure D.20 Check polygons.



P1 - Northbound

	P1
ID	Northbound
Description	McCormick T-R
Type	Input
Maneuver	Straight
Users	✓ Vehicles ✓ Bicycles Pedestrians
Lane Number	2

Figure D.21 Setting polygons characteristics.

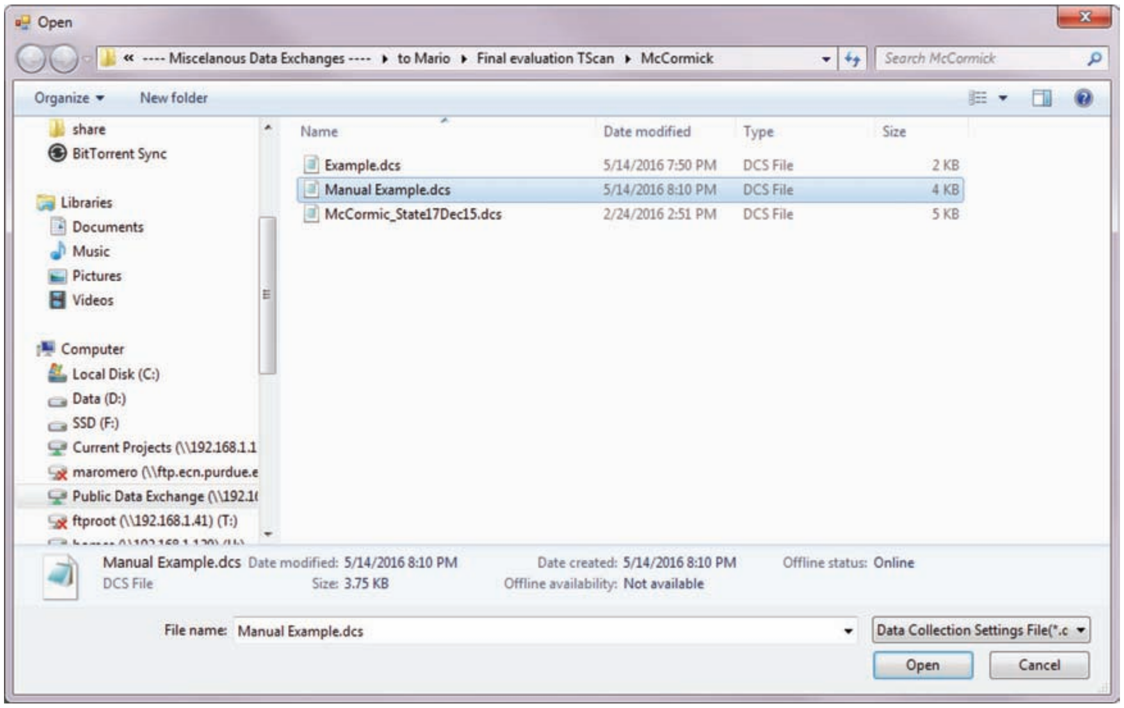

Save As
« ---- Miscelanous Data Exchanges ---- ▸ to Mario ▸ Final evaluation TScan ▸ McCormick
Search McCormick
Organize
New folder
RealPlayer Cloud
share
BitTorrent Sync
Libraries
Documents
Music
Pictures
Videos
Computer
Local Disk (C:)
Data (D:)
SSD (F:)
Current Projects (\\192.168.1
maromero (\\ftp.ecn.purdue
Public Data Exchange (\\192.
ftproot (\\192.168.1.41) (T:)
Hide Folders
Save
Cancel

Name	Date modified	Type	Size
Example.dcs	5/14/2016 7:50 PM	DCS File	2 KB
McCormic_State17Dec15.dcs	2/24/2016 2:51 PM	DCS File	5 KB

File name: Manual Example.dcs

Save as type: Data Collection Settings File(*.dcs)

Figure D.22 Save layout.


Open
« ---- Miscelanous Data Exchanges ---- ▸ to Mario ▸ Final evaluation TScan ▸ McCormick
Search McCormick
Organize
New folder
share
BitTorrent Sync
Libraries
Documents
Music
Pictures
Videos
Computer
Local Disk (C:)
Data (D:)
SSD (F:)
Current Projects (\\192.168.1.1
maromero (\\ftp.ecn.purdue.e
Public Data Exchange (\\192.1(
ftproot (\\192.168.1.41) (T:)
Open
Cancel

Name	Date modified	Type	Size
Example.dcs	5/14/2016 7:50 PM	DCS File	2 KB
Manual Example.dcs	5/14/2016 8:10 PM	DCS File	4 KB
McCormic_State17Dec15.dcs	2/24/2016 2:51 PM	DCS File	5 KB

Manual Example.dcs Date modified: 5/14/2016 8:10 PM Date created: 5/14/2016 8:10 PM Offline status: Online
DCS File Size: 3.75 KB Offline availability: Not available

File name: Manual Example.dcs
Data Collection Settings File(*.c

Figure D.23 Open layout.

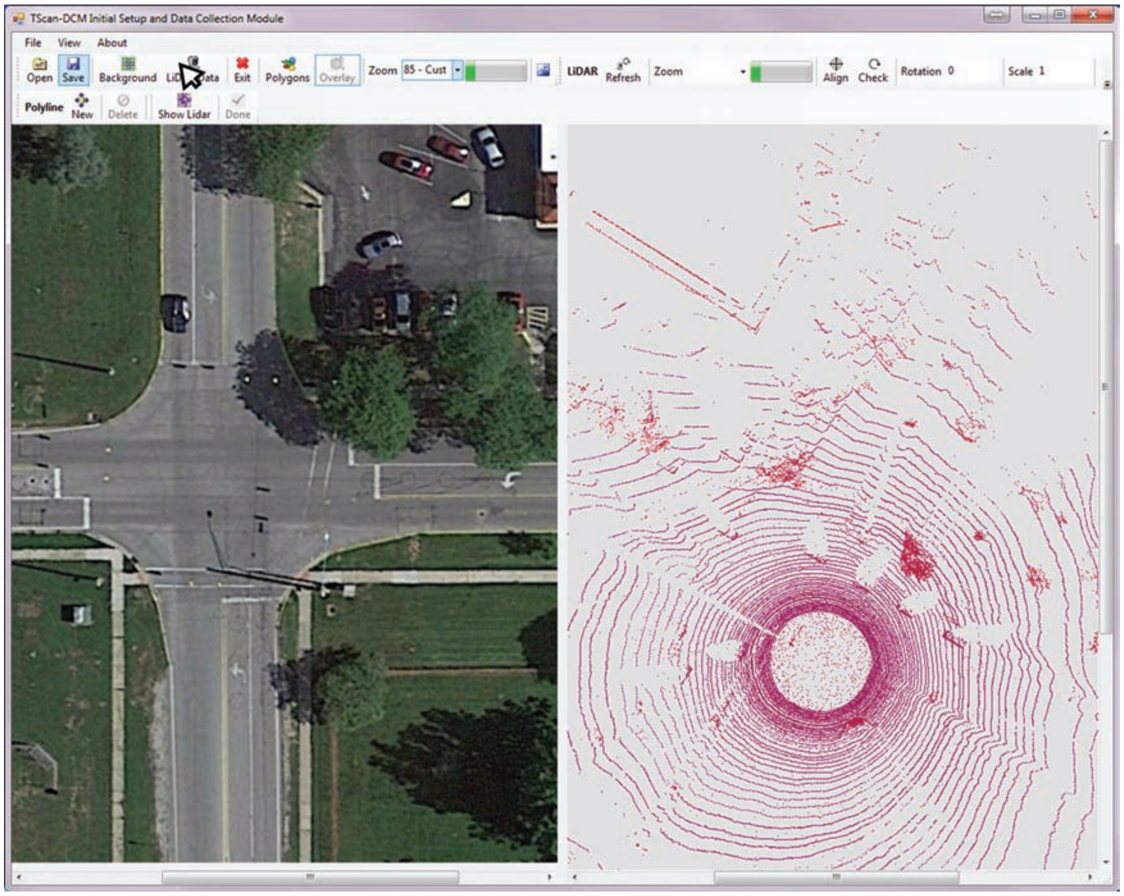


Figure D.24 Initial sensor data.

or right click on *Shift* to adjust the horizontal or the vertical shift by 0.01% with the location of the click on the button determining the shift direction. The overlay will be updated after clicking as shown in Figure D.29. Press and hold the left mouse button to repeat the operation five times per second and release it to stop.

Display commands

Background transparency adjustment. To facilitate the visualization a button to change the background transparency is available. Click on the *Transparency* button by 10% or right click on the *Transparency* button by 5% to adjust the orthographic-image, the location of the click on the button will determinate if the transparency will be increased or decreased. The new overlay will be updated after clicking as shown in Figure D.30. Press and hold the left mouse button to repeat the operation 5 times per second and release it to stop.

Refresh. The *Refresh* button will display the TScan data points without overlaying the orthographic-image as shown in Figure D.31.

Check. Clicking on *Check* displays the TScan data points overlaying the orthographic-image as shown in Figure D.32 when a manual change in the parameters values is made.

D.5.3 Exporting Aligned Polygons

Once the alignment is completed, return to the polygons view by clicking on *Polygons*. Then click on the *Export* button to save the polygons in the TScan reference system. This process may take some time. After it ends, the user is asked to provide a file name and a folder to which the results are saved as shown in Figure D.33.

D.6 Data Collection and Processing

Once the data is transferred from the System Preparation Modules, the Data Collection and Processing Module gathers the information from the sensor, processes it, and produces the results required for future engineering applications.

Click on *Collect Data* to provide a file name and a folder to which the data is to be saved. After providing the file name, the data collection begins (Figure D.34).

The Data Collection and Processing Module does not require user intervention while collecting data. Click on *Stop* to finalize the data collection. The program will continue running for several minutes until the final batch data are processed. A message box will be displayed to inform the user that i is safe to exit the program as shown in Figure D.35.

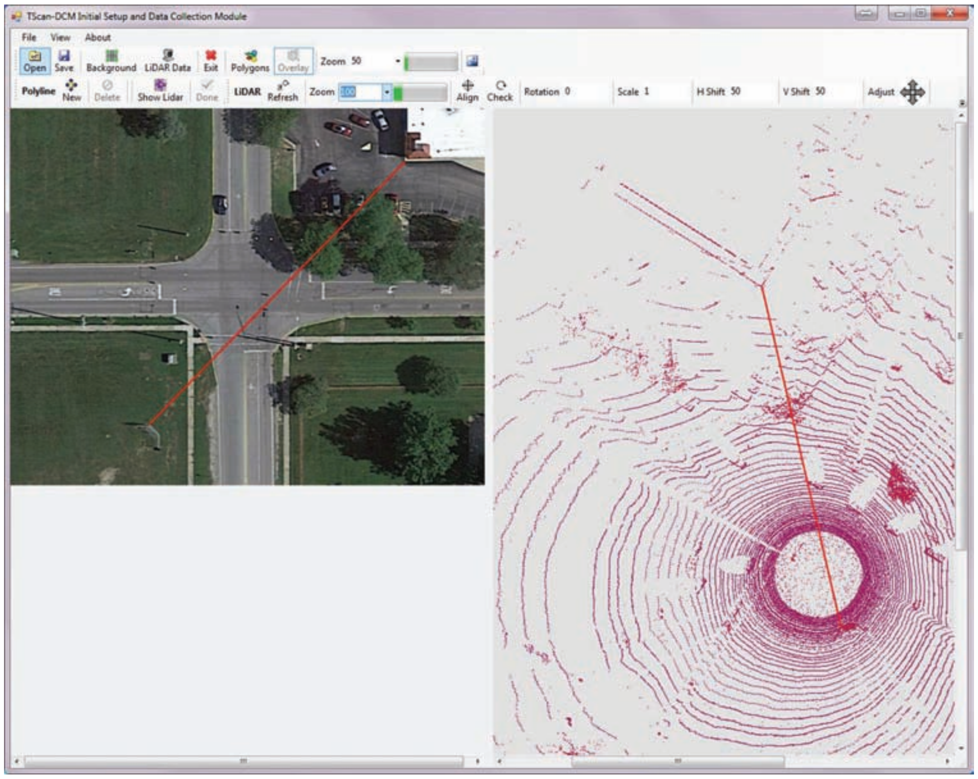
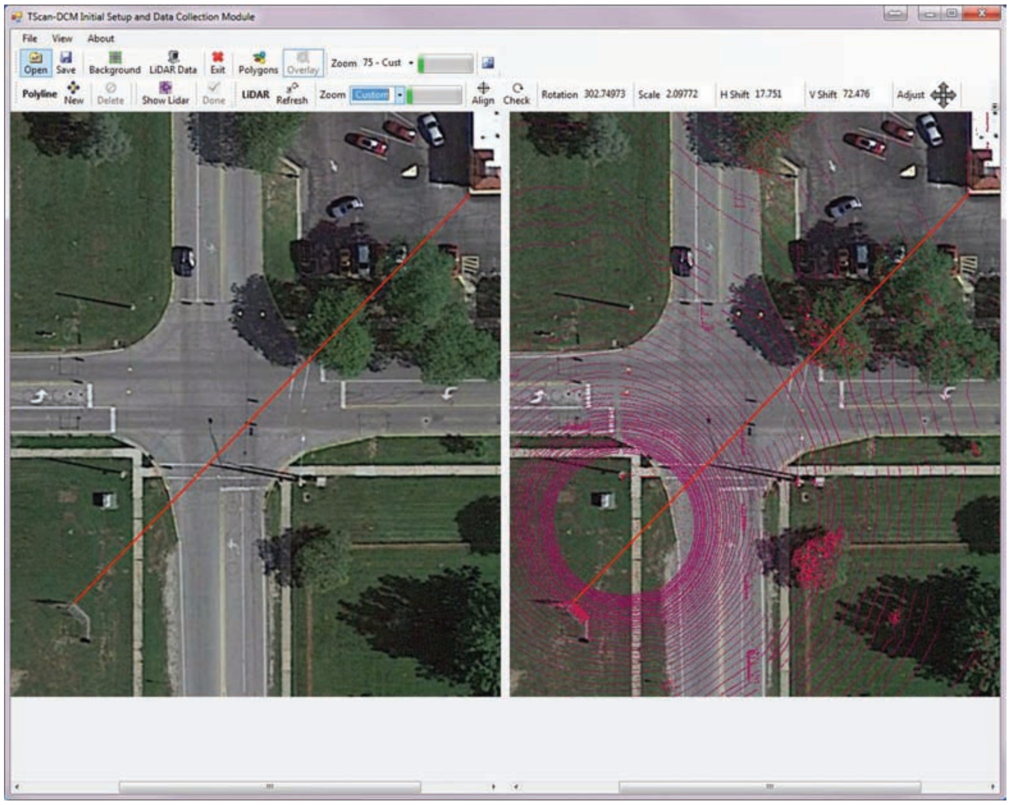


Figure D.25 Point correspondence to align TScan and orthographic-image.

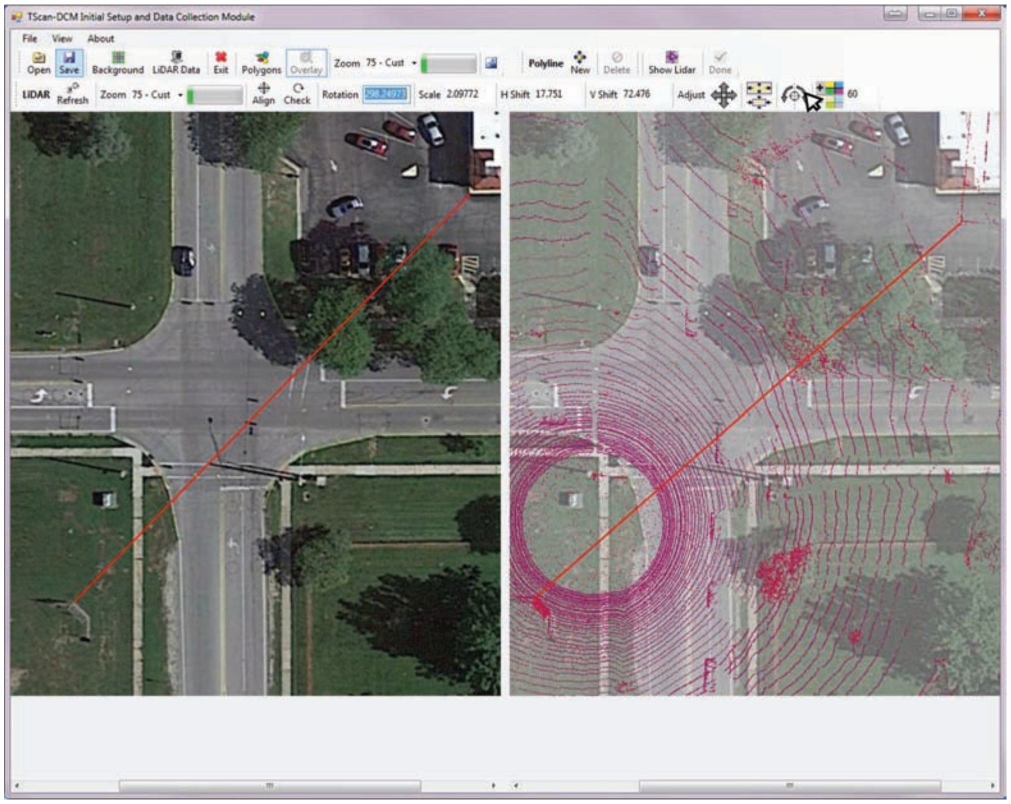


Zoom 75 - Cust

Zoom Custom

Rotation 302.74973 Scale 2.09772 H Shift 17.751 V Shift 72.476

Figure D.26 TScan and orthographic-image alignment.

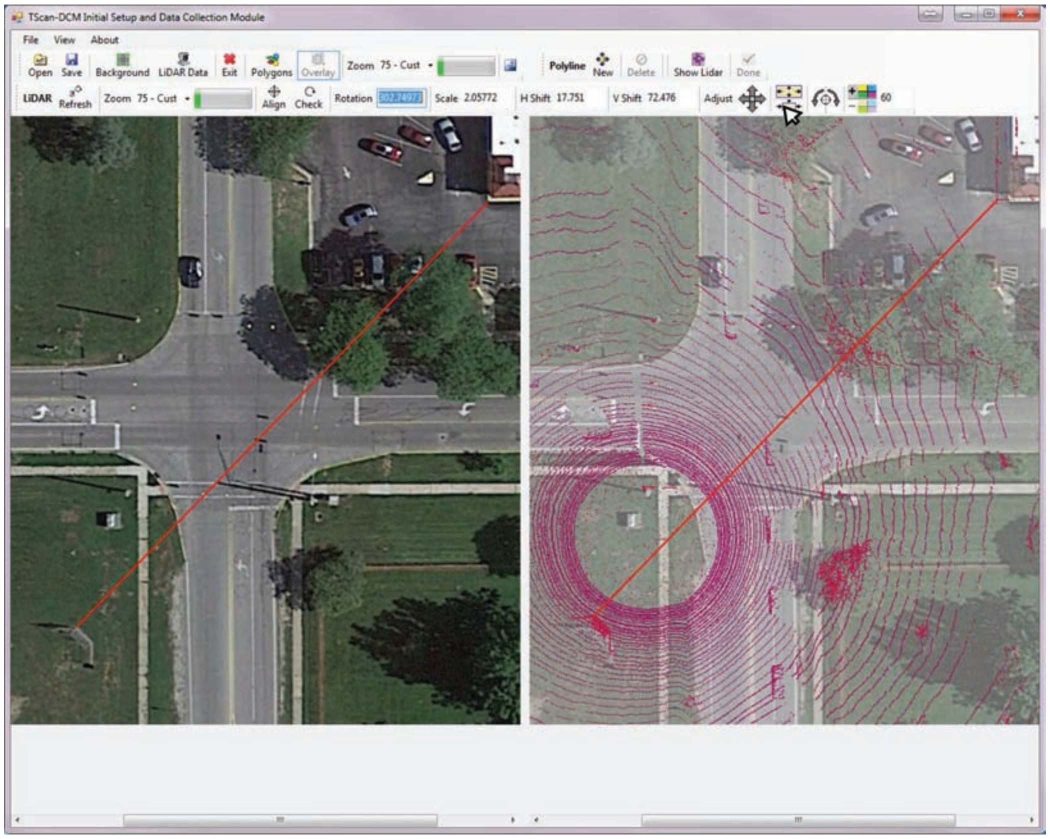


Zoom 75 - Cust

Zoom 75 - Cust

Rotation 298.24973 Scale 2.09772 H Shift 17.751 V Shift 72.476

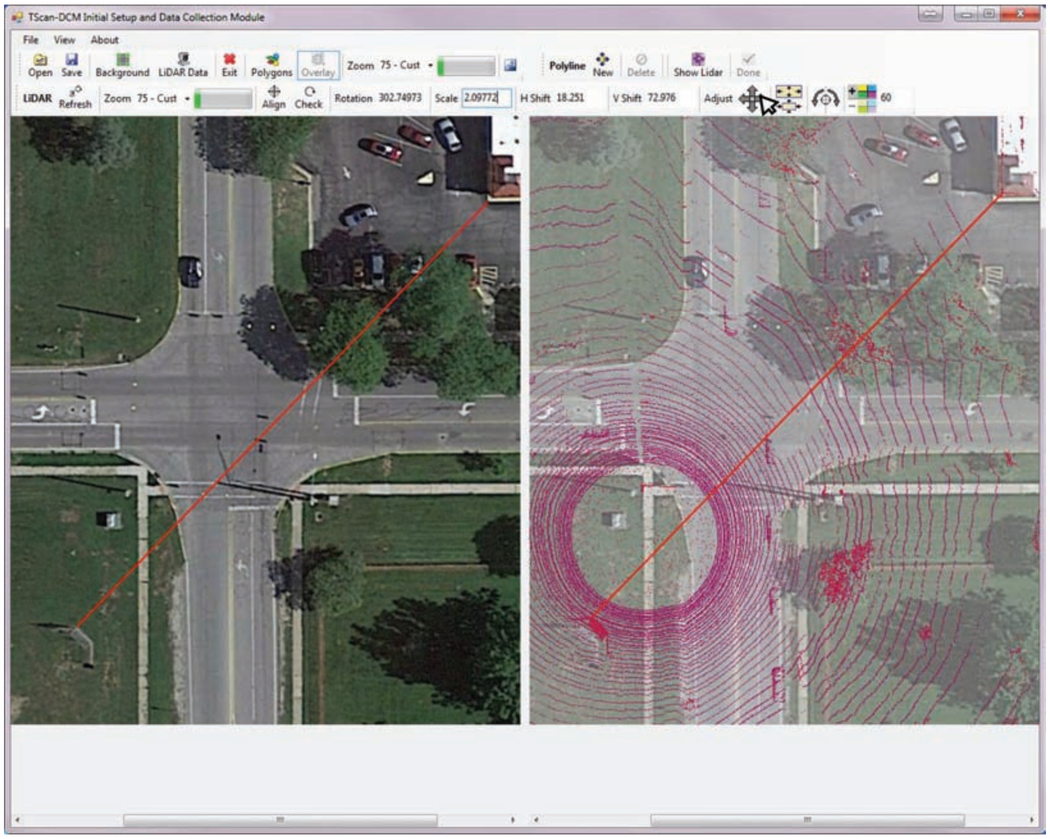
Figure D.27 Rotation adjustment.



Zoom 75 - Cust

Zoom 75 - Cust Rotation 302.74973 Scale 2.05772 H Shift 17.751 V Shift 72.476

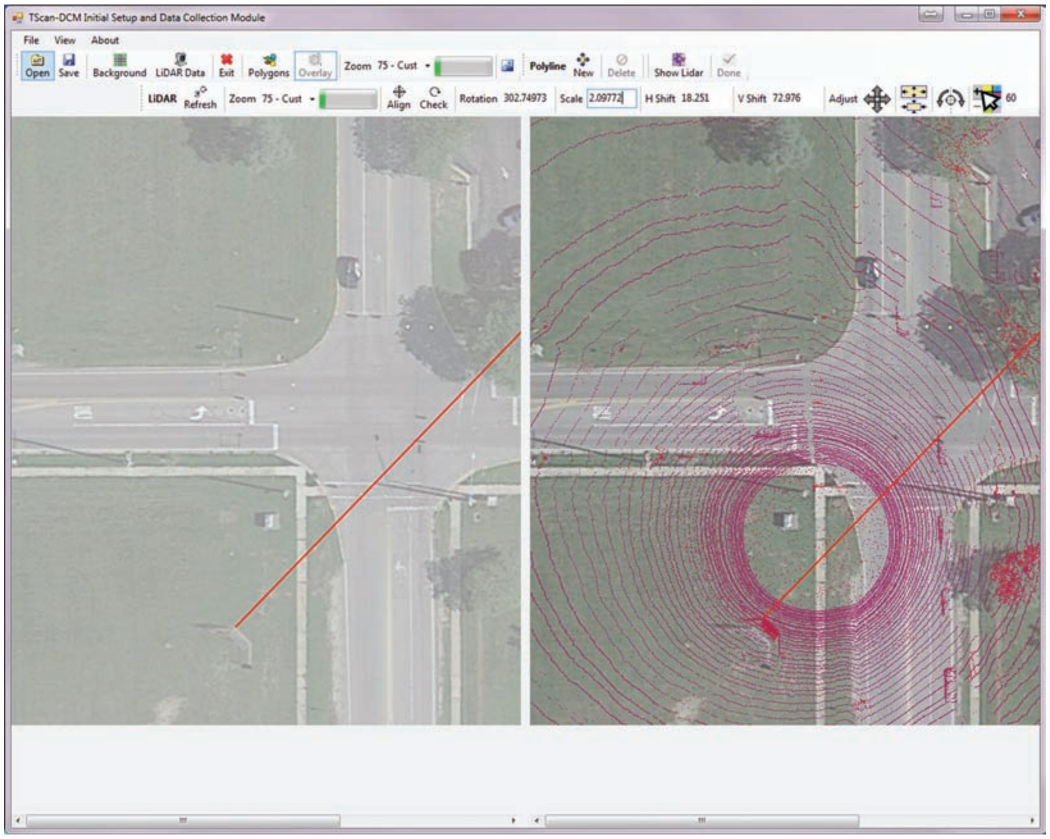
Figure D.28 Scale adjustment.



Zoom 75 - Cust

Zoom 75 - Cust Rotation 302.74973 Scale 2.09772 H Shift 18.251 V Shift 72.976

Figure D.29 Horizontal and vertical shift adjustment.



Zoom 75 - Cust

Zoom 75 - Cust

Rotation 302.74973

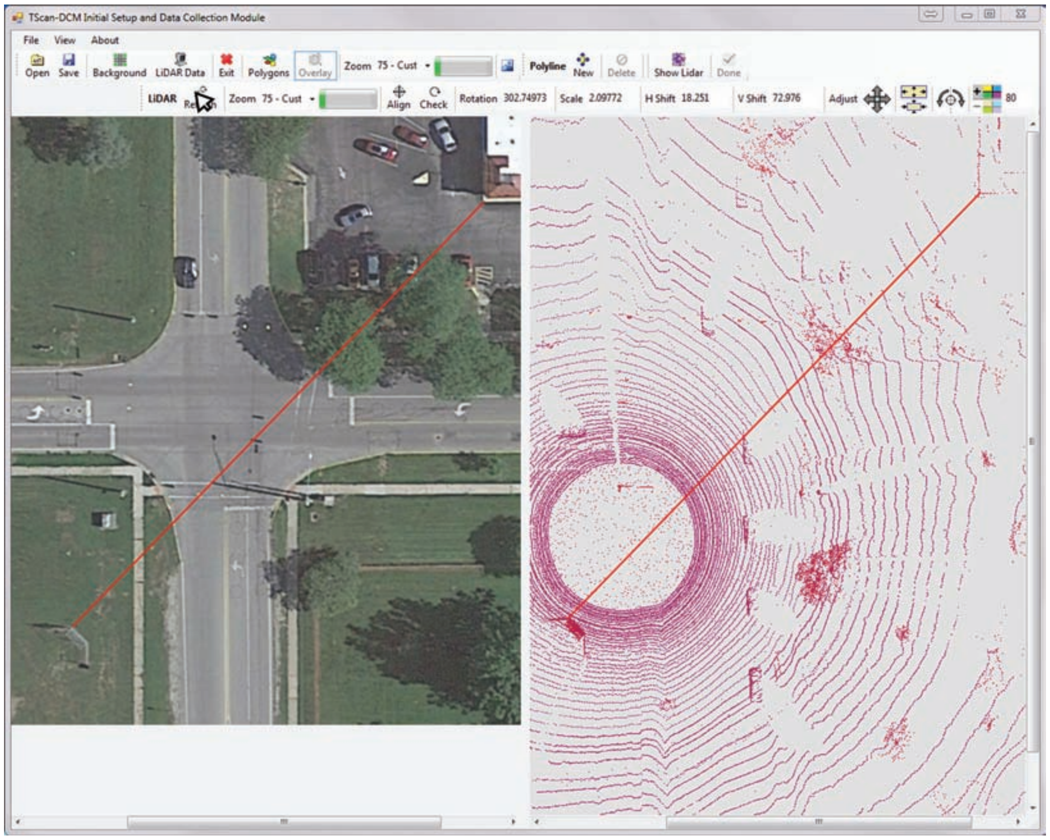
Scale 2.09772

H Shift 18.251

V Shift 72.976

60

Figure D.30 Background transparency adjustment.



Zoom 75 - Cust

Zoom 75 - Cust

Rotation 302.74973

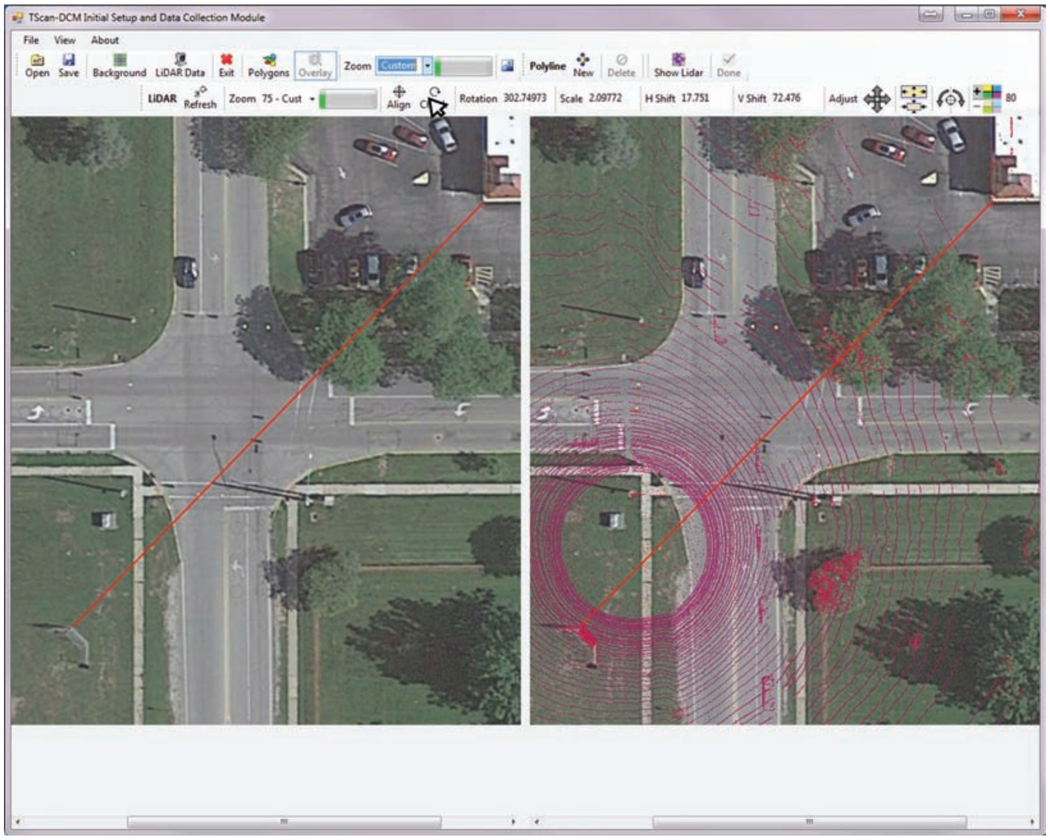
Scale 2.09772

H Shift 18.251

V Shift 72.976

80

Figure D.31 Refresh results.



Zoom Custom

Zoom 75 - Cust

Rotation 302.74973

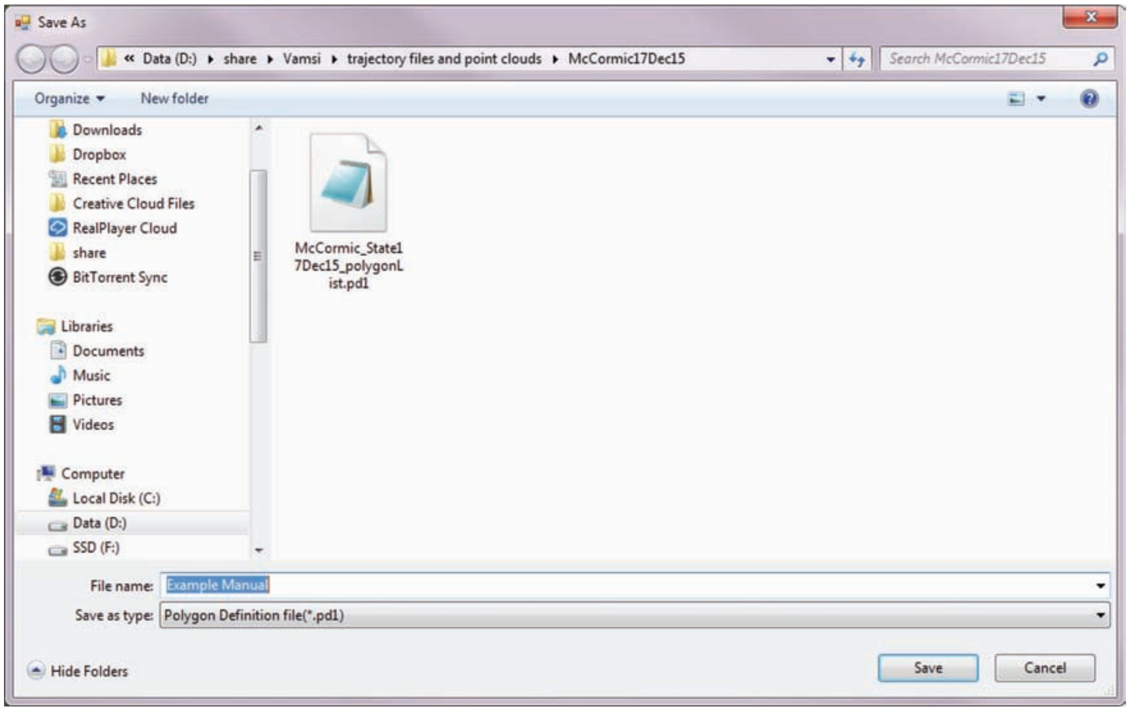
Scale 2.09772

H Shift 17.751

V Shift 72.476

80

Figure D.32 Check results.



« Data (D:) ▸ share ▸ Vamsi ▸ trajectory files and point clouds ▸ McCormic17Dec15

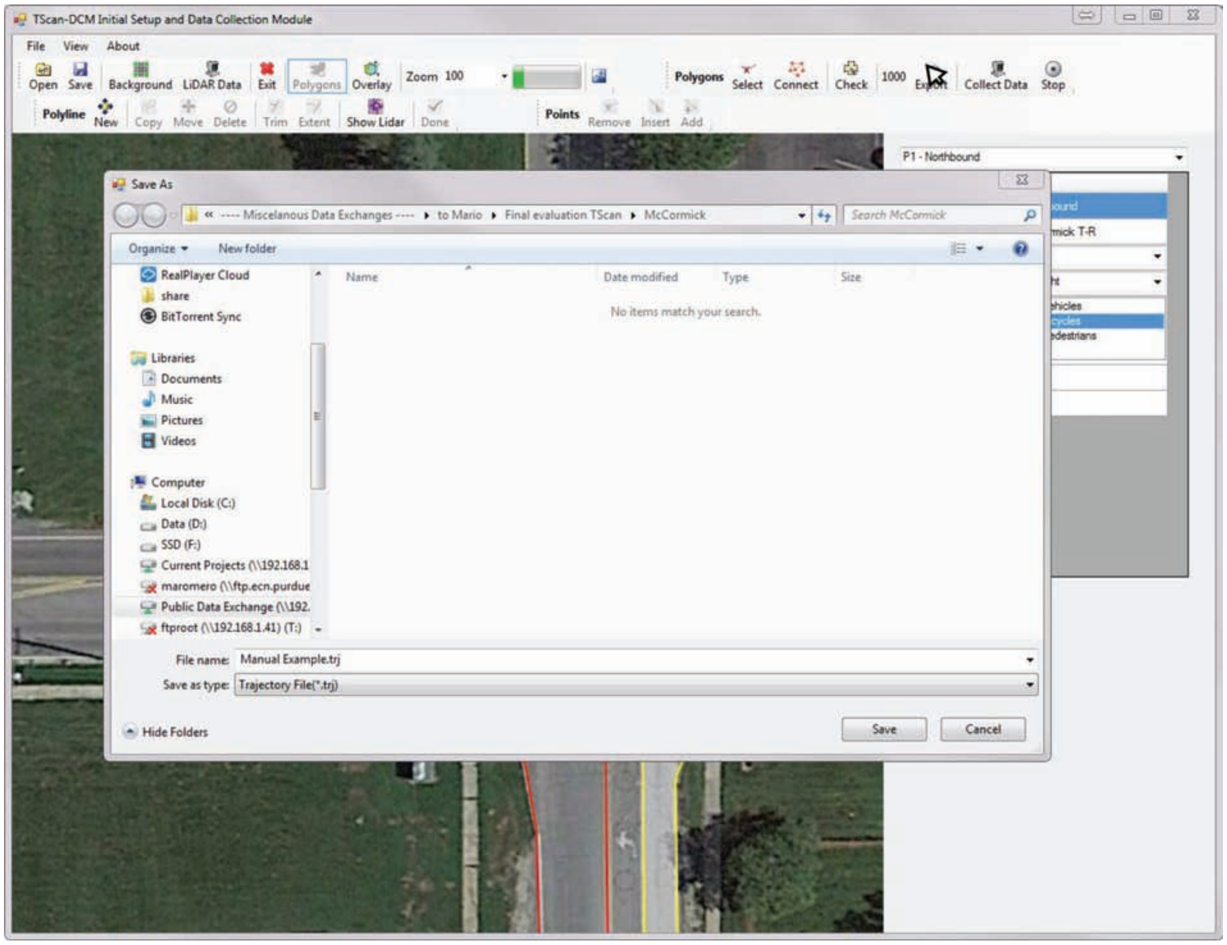
Search McCormic17Dec15

McCormic_State17Dec15_polygonList.pd1

File name: Example Manual

Save as type: Polygon Definition file(*.pd1)

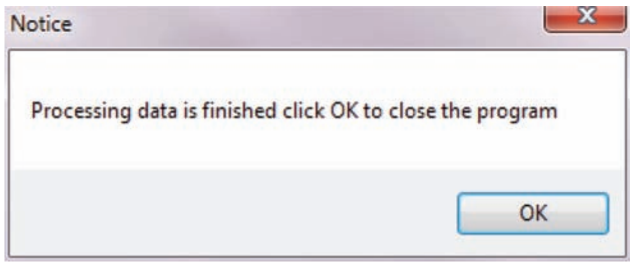
Figure D.33 Exporting polygons.


TScan-DCM Initial Setup and Data Collection Module
File View About
Open Save Background LiDAR Data Exit Polygons Overlay Zoom 100 Polygons Select Connect Check 1000 Export Collect Data Stop
Polyline New Copy Move Delete Trim Extent Show Lidar Done Points Remove Insert Add
P1 - Northbound
Save As
« ---- Miscelanous Data Exchanges ---- ▸ to Mario ▸ Final evaluation TScan ▸ McCormick
Search McCormick
Organize New folder
RealPlayer Cloud
share
BitTorrent Sync
Libraries
Documents
Music
Pictures
Videos
Computer
Local Disk (C:)
Data (D:)
SSD (F:)
Current Projects (\\192.168.1
maromero (\\ftp.ecn.purdue
Public Data Exchange (\\192.
ftproot (\\192.168.1.41) (T:)
Name Date modified Type Size
No items match your search.
Hide Folders
Save Cancel

File name: Manual Example.trj

Save as type: Trajectory File(*.trj)

Figure D.34 Save TScan data.



Processing data is finished click OK to close the program

Figure D.35 End calculations.

APPENDIX E. TSCAN TRAFFIC COUNTER—USER MANUAL

TScan uses LiDAR technology that can detect and track various types of road users including buses, cars, pedestrians, and bicycles; and unlike video detection, LiDAR data has a one-to-one correspondence with the physical world. Hence, it is possible in principle to produce the positions and velocities of road users in real-time as needed for traffic and safety applications, with the errors of estimation dependent only on the resolution and accuracy of the LiDAR sensor.

The engineering application for counting vehicles and display trajectory files (The TScan Traffic Counter) User Manual is a tool developed by the Purdue University Center for Road Safety that includes the user interface to upload information from the Initial Setup and Data Collection Module (TScan) and to enter the required information for counting vehicles and for displaying trajectory files generated at a given intersection. TScan Traffic Counter was developed as part of the project called "Stationary LiDAR for Traffic and Safety Applications – Vehicles Interpretation and Tracking (TScan)" supported through the Joint Transportation Research Program of the Indiana Department of Transportation and Purdue University.

This user manual describes TScan Traffic Counter and illustrates its features.

E.1 TScan Traffic Counter Overview

TScan Traffic Counter is one of the engineering applications developed to demonstrate TScan's capabilities.

Figure E.1 shows the TScan framework in which the TScan Traffic Counter is included.

The TScan Traffic Counter tool is a computer application that allows the user to set the directional counting based on the TScan output files and obtain directional counting using user- defined counting periods. It includes creating the origin-destination matrix based on the created reference polygons of the intersection.

A set of tools to retrieve TScan information, assign polygons to the O_D matrix, set up the counting interval, and count vehicles is available.

E.2 Installation

TScan Traffic Counter is compatible with Windows 7/8/10. In order to run TScan Traffic Counter, the MS .NET 4.0 Framework or later component must be installed.

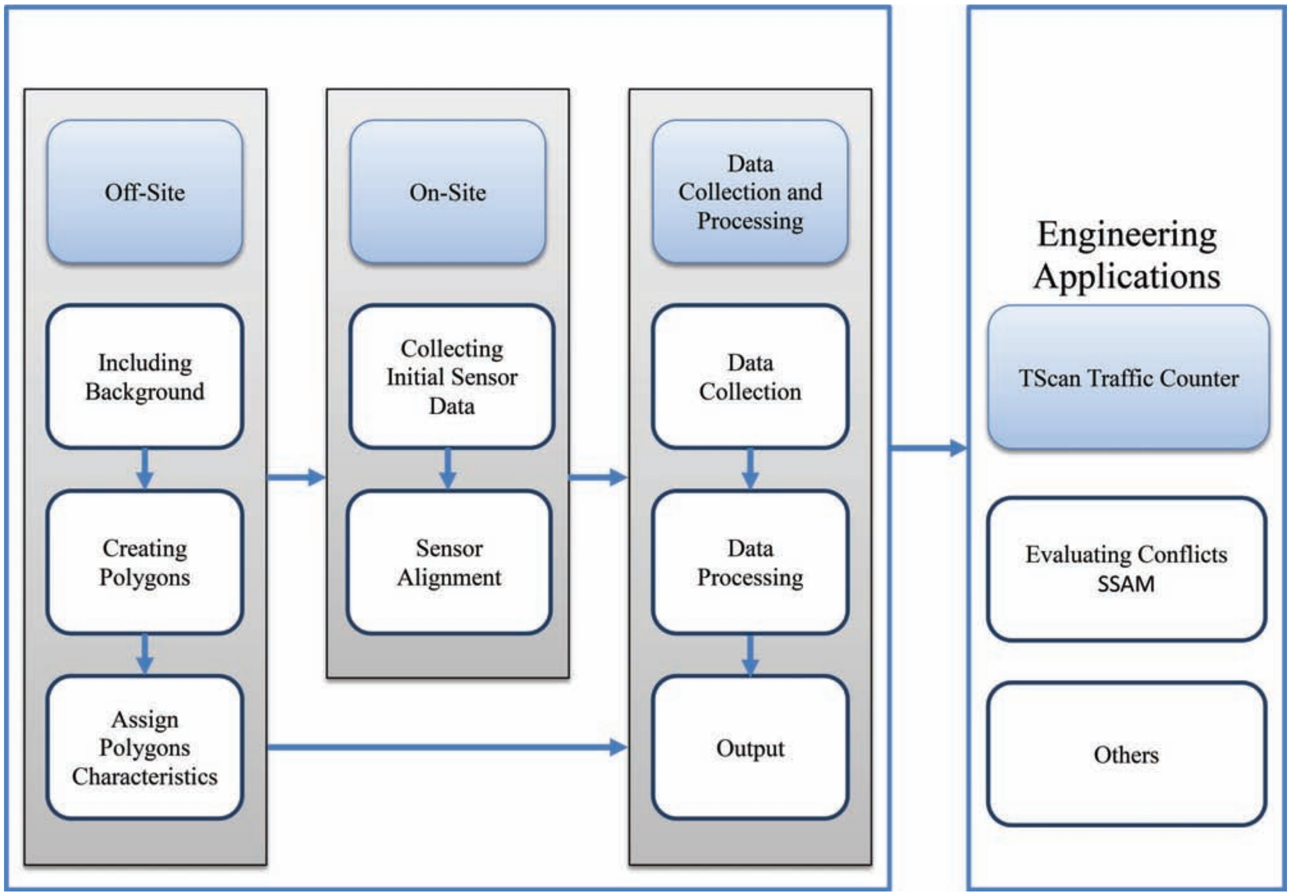


Figure E.1 TScan overview.

If the MS .NET 4.0 Framework is not present during the installation, TScan Traffic Counter will attempt to install this component if the PC is connected to the Internet.

To install the TScan Traffic Counter interface, follow the steps given below:

1. Extract the contents of the archived file to your local drive.
2. The installation process is initiated by clicking on the *setup.exe* file.
3. Step by step instructions that explains the installation process is found in *readme.exe*.

Note: The user should always read the readme.txt file included in the installation package which includes the most up-to-date installation instructions.

After checking that the program is working, the user may delete the unzipped files in the folder with the *setup.exe* file to save disk space. The user should save the zipped/compressed file in case it is needed to reinstall the program.

E.3 Launching TScan Traffic Counter

The TScan Traffic Counter program can be launched using any of the following methods after installation:

Method 1:

1. Double click the shortcut on the desktop.

Method 2:

1. Press *Start* button.
2. TScan Traffic Counter should appear in the list of installed programs.
3. Single click on the shortcut.

Method 3:

1. Press *Start* button.
2. Start typing "*TScan Traffic Counter*," the program shortcut should appear in the search results.
3. Single click on the shortcut.

Method 4:

1. Open My Computer.
2. Browse to the location where the software was installed. Typically C:\Program Files (x86)/TScan Traffic Counter/.
3. Click on TScan Traffic Counter.exe.

The main interface appears within a few seconds (see Figure E.2).

E.3.1 TScan Traffic Counter Interface

The TScan Traffic Counter interface includes a command bar in the first row and a tab control below (Figure E.3). The command bar facilitates operations on files and the background. The Settings tab allows the user to input the origin-destination matrix for directional counting and to set the counting interval, whereas the Output tab allows the user to display and save the counting results. The following sections describe each activity in detail.

The command bar is movable and can be snapped on to any of the four window edges. The bar can be moved by simply dragging it around.

E.3.2 Files

TScan Traffic Counter requires only an orthographic-image of the intersection in which the data collection will be performed. It reads the files generated with the TScan program. Table E.1 shows the list of files used by TScan Traffic Counter.

E.4 TScan Traffic Counter

The TScan Traffic Counter extracts the directional counts of moving objects from the trajectory file obtained from TScan. It requires the intersections characteristics, the trajectory file, a user- defined origin-destination matrix, and the counting interval.

This chapter will explain how to upload both files, enter the required data, run the counting process and export results.

E.4.1 Upload Data

There are two files required for counting turning movements: The data collection setting file and the trajectory file. Both files are generated using TScan.

Upload data collection settings file. To open the data collection settings file <*Location Name*>*.dsc*, obtained from TScan, click on *Background* to open the Windows file selection window shown in Figure E.4, where the appropriate file can be selected. Once the file is selected, click on *Open* to upload the image and the corresponding data. The orthographic-image is uploaded automatically since its location is saved on the *dcs* file.

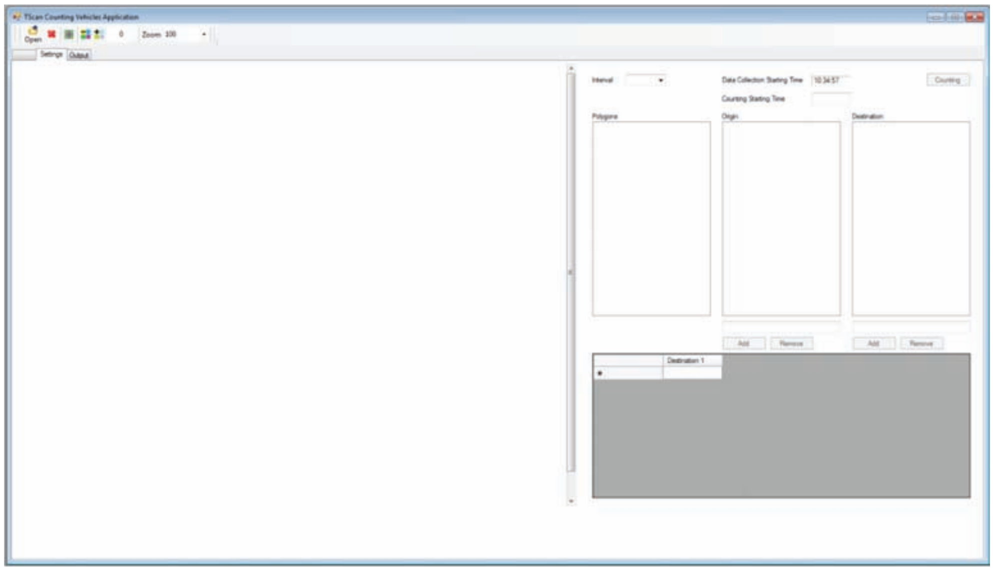
Upload trajectory file. To open the trajectory file <Location Name>.trj, obtained from TScan, click on Open. The Windows file selection window will open as shown in Figure E.5. Select the appropriate trajectory file. Once the file is selected, click on Open to upload the vehicle information.

E.4.2 Counting Setup

In order to setup the counting process it is recommended to follow the procedural steps below:

- Review the uploaded data.
- Select the counting interval.
- Set the counting starting time.
- Create the origin destination matrix.

Each of these steps are explained in further detail below.



Data Collection Starting Time 10:34:57

Counting Starting Time

Figure E.2 TScan traffic counter main interface.

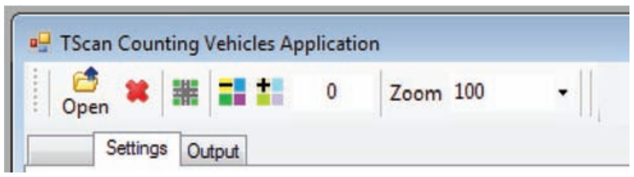


Figure E.3 Command bar in the TScan Traffic Counter interface.

Review uploaded data. Once the data collection settings file is open, the data collections starting time is displayed. The list of polygons IDs of the intersection area are set as well as the lists of polygons arranged by polygon type. The background image is displayed. The polygons IDs and the polygons edges are also displayed on the image.

Review the uploaded values and images to check the intersection setup.

Set the counting starting time. The counting starting time is set by default as the data collection starting time. To change the counting starting time, enter the appropriated time on the counting starting time text box as shown in Figure E.6.

Vehicles entering the intersection polygon before the counting starting time are not considered in the counts.

Select counting aggregation time interval. The selected time interval governs the counting aggregation level. Select the counting interval from the combo-box as shown in Figure E.7. The interval options are 1, 3, 5, 10, 15, 20, 30, and 60 minutes.

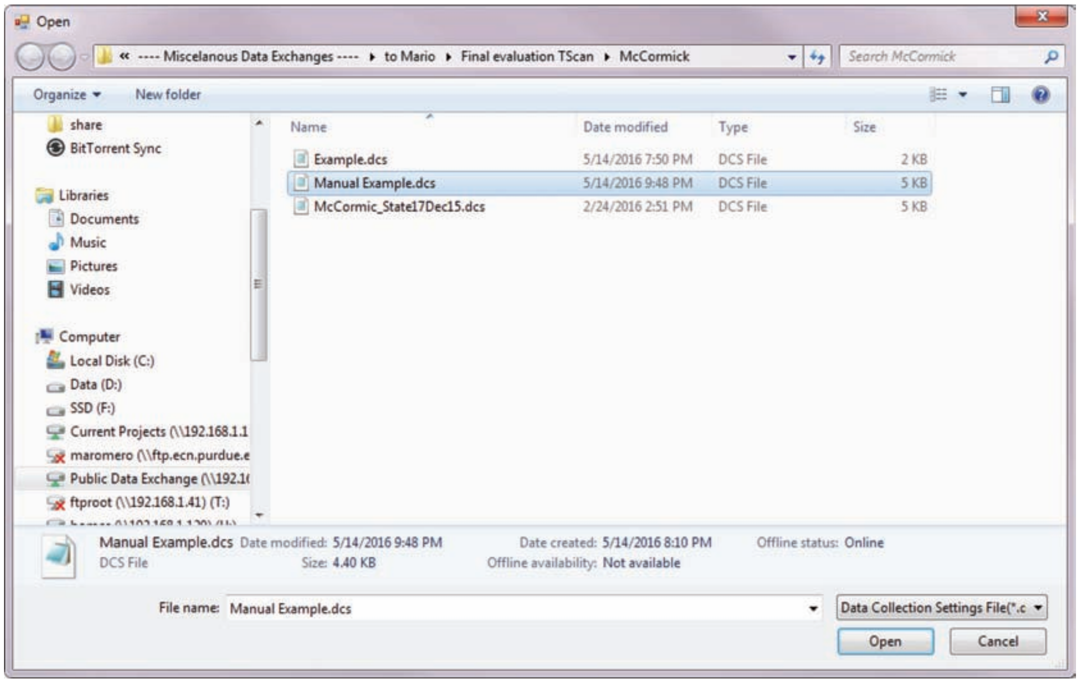
Note: Only full intervals are included in the results i.e. if the data set span time period is not a multiple of the selected interval, then the last portion is ignored.

Creating origin-destination matrix. In order to count turning movements it is necessary to set the origin

TABLE E.1
TScan Traffic Counter list of files.

File	Format	Description
<Location Name>.dsc	Text format with extension dcs	TScan generated file that contains all user information regarding the data collection settings using the orthographic-image coordinates system.
<Orthographic-image>	Any image format such as bmp, jpg, jpeg, gif, png, tif, etc.	User provided top view image of the intersection saved in image format. It must be the same file used in TScan.
<Location Name>.trj	Binary file in SSAM* trajectory file format	TScan output file that contains the location, speed and acceleration of each vehicle every 0.1 sec. For more information visit: http://www.fhwa.dot.gov/publications/research/safety/08049/
<User defined Name>.csv	Comma separated value	TScan Traffic Counter output file that contains the counting results.

*SSAM: Surrogate Safety Assessment Model.



Open

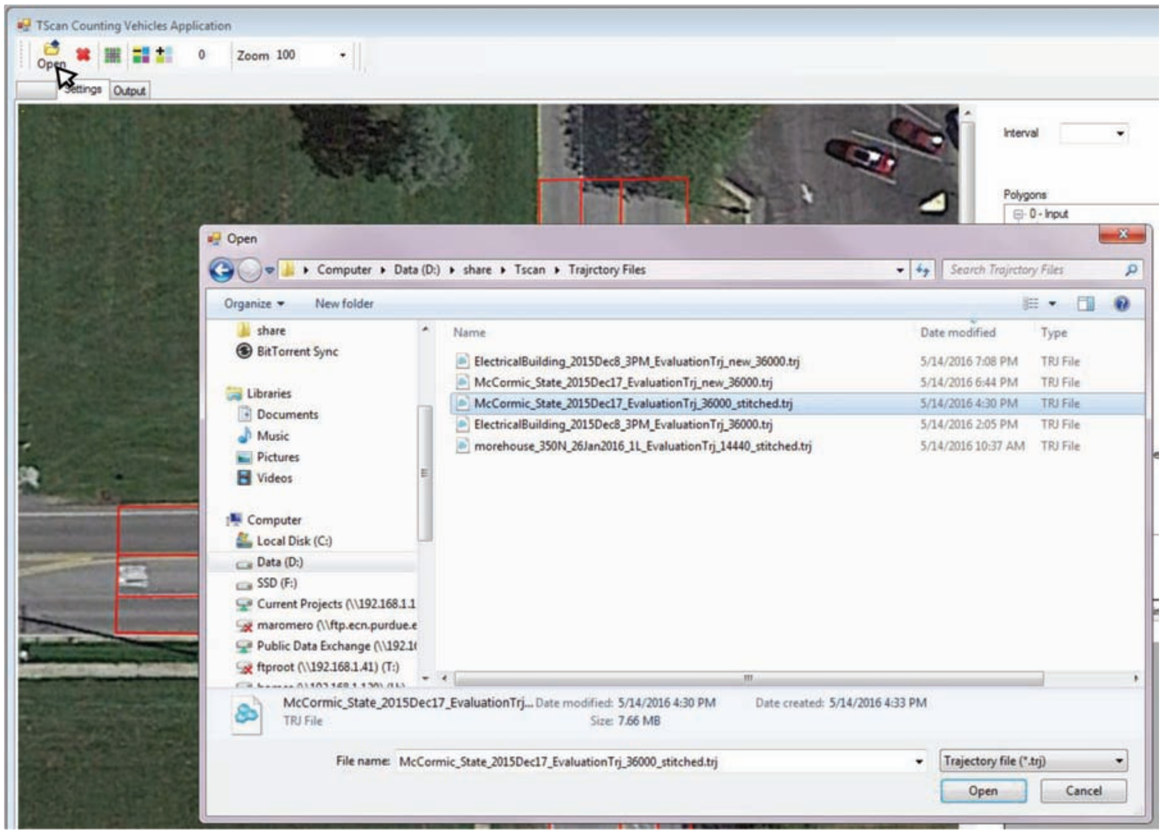
« ---- Miscelanous Data Exchanges ---- ▸ to Mario ▸ Final evaluation TScan ▸ McCormick

Name	Date modified	Type	Size
Example.dcs	5/14/2016 7:50 PM	DCS File	2 KB
Manual Example.dcs	5/14/2016 9:48 PM	DCS File	5 KB
McCormic_State17Dec15.dcs	2/24/2016 2:51 PM	DCS File	5 KB

Manual Example.dcs Date modified: 5/14/2016 9:48 PM Date created: 5/14/2016 8:10 PM Offline status: Online
DCS File Size: 4.40 KB Offline availability: Not available

File name: Manual Example.dcs
Data Collection Settings File(*.c

Figure E.4 Upload data collection settings file.



Open

▸ Computer ▸ Data (D:) ▸ share ▸ Tscan ▸ Trajrctory Files

Name	Date modified	Type
ElectricalBuilding_2015Dec8_3PM_EvaluationTrj_new_36000.trj	5/14/2016 7:08 PM	TRJ File
McCormic_State_2015Dec17_EvaluationTrj_new_36000.trj	5/14/2016 6:44 PM	TRJ File
McCormic_State_2015Dec17_EvaluationTrj_36000_stitched.trj	5/14/2016 4:30 PM	TRJ File
ElectricalBuilding_2015Dec8_3PM_EvaluationTrj_36000.trj	5/14/2016 2:05 PM	TRJ File
morehouse_350N_26Jan2016_1L_EvaluationTrj_14440_stitched.trj	5/14/2016 10:37 AM	TRJ File

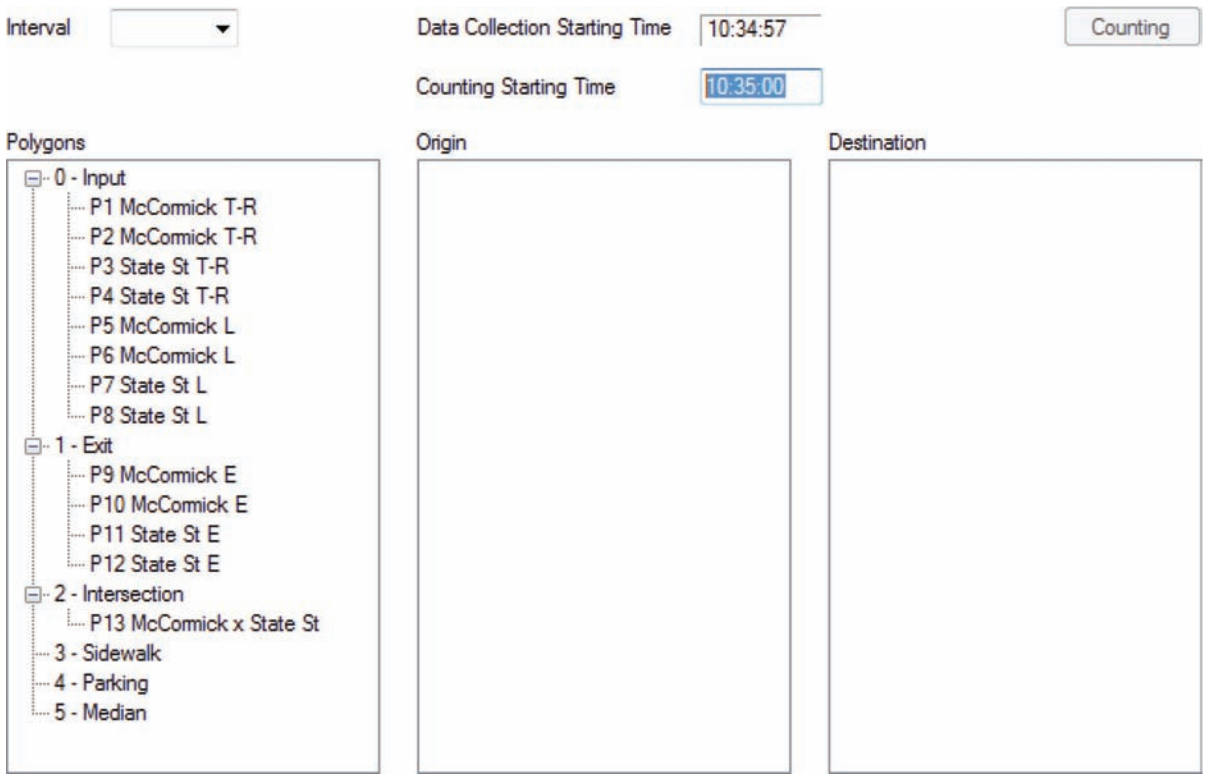
McCormic_State_2015Dec17_EvaluationTrj... Date modified: 5/14/2016 4:30 PM Date created: 5/14/2016 4:33 PM
TRJ File Size: 7.66 MB

File name: McCormic_State_2015Dec17_EvaluationTrj_36000_stitched.trj
Trajectory file (*.trj)

Figure E.5 Upload trajectory file.

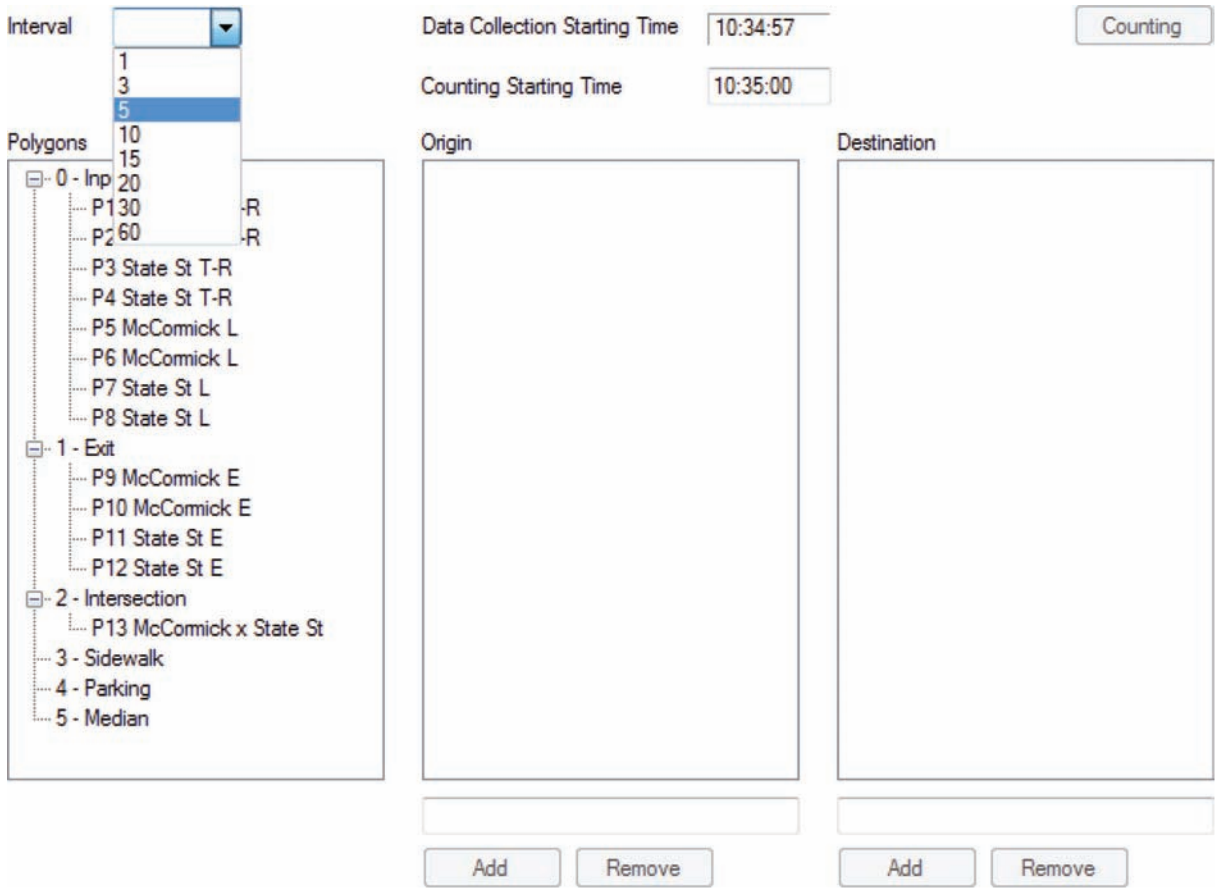
approaches and the destination areas by assigning polygons to each origin and destination. One single origin or destination can include several polygons. The require steps are explained below.

Creating origin approaches. To create an origin approach. Select the current origin textbox. Enter the origin name and click on *Add*. This action will create an origin placeholder and selects it as shown in Figure E.8.


Data Collection Starting Time 10:34:57

Counting Starting Time 10:35:00

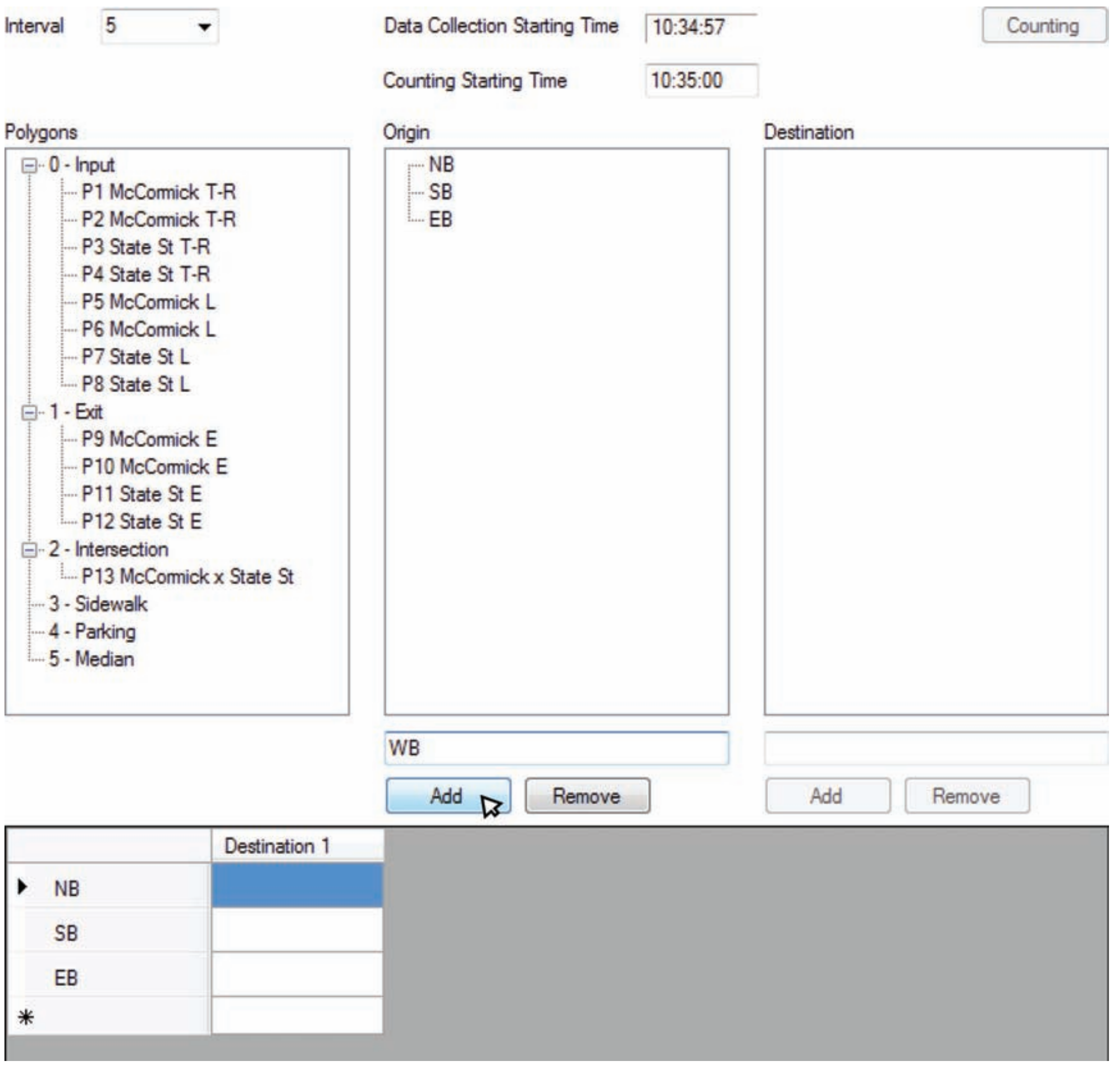
Figure E.6 Setting the counting starting time.


Data Collection Starting Time 10:34:57

Counting Starting Time 10:35:00

Interval: 1, 3, 5, 10, 15, 20, 30, 60 (5 selected)

Figure E.7 Counting interval selection.


Interval 5

Data Collection Starting Time 10:34:57

Counting Starting Time 10:35:00

Polygons
- 0 - Input
  - P1 McCormick T-R
  - P2 McCormick T-R
  - P3 State St T-R
  - P4 State St T-R
  - P5 McCormick L
  - P6 McCormick L
  - P7 State St L
  - P8 State St L
- 1 - Exit
  - P9 McCormick E
  - P10 McCormick E
  - P11 State St E
  - P12 State St E
- 2 - Intersection
  - P13 McCormick x State St
- 3 - Sidewalk
- 4 - Parking
- 5 - Median

Origin
- NB
- SB
- EB

WB

	Destination 1
NB	
SB	
EB	
*	

Figure E.8 Creating origin approaches.

Then add the corresponding polygons to the selected origin.

Creating a destination area. To create a destination area, select the current destination textbox. Enter the destination name and click on Add. This action will create a destination area placeholder and becomes the selected destination as shown in Figure E.9. Then, add the corresponding polygons to the selected destination.

Adding polygons to an origin approach or to a destination area. To add a polygon to a selected origin or destination, select the origin/destination on the corresponding list by clicking on the name. The origin/destination name will be set as the selected one. Then double-click on the polygon list to add the polygon as shown in Figure E.10.

Removing polygons from an origin approach or a destination area. To remove a polygon from an origin approach or a destination area, select the polygon on the area to be removed and click on *Remove* button as shown in Figure E.11.

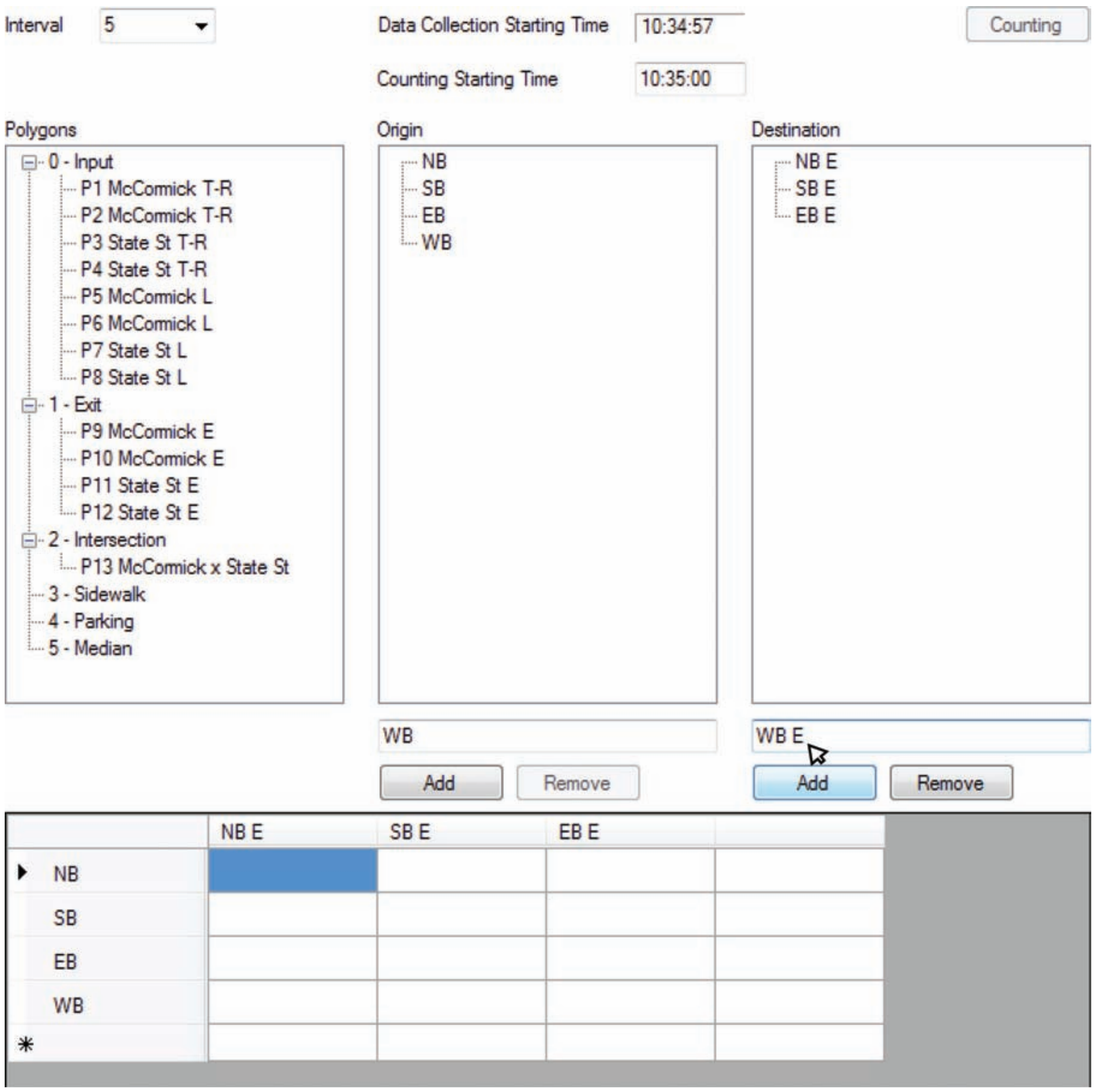
Removing an origin approach or a destination area. To remove an origin approach or a destination area, select the origin approach or the destination area to be removed and click on *Remove* button as shown in Figure E.12.

Setting movement names. Once the origin approaches and the destination areas are set it is necessary to name the movements corresponding to the origin destination pairs as shown in Figure E.13. Only the named movements are counted in the counting process.

E.4.3 Counting Vehicles

Once the counting interval is selected, the counting starting time is set and the origin-destination matrix is created the counting process can be initiated.

Counting process. Click on the *Counting* button to initiate the counting process. Once completed the *Statistics* tab is automatically selected to display results as shown in in Figure E.14.


Interval 5

Data Collection Starting Time 10:34:57

Counting Starting Time 10:35:00

Polygons
- 0 - Input
  - P1 McCormick T-R
  - P2 McCormick T-R
  - P3 State St T-R
  - P4 State St T-R
  - P5 McCormick L
  - P6 McCormick L
  - P7 State St L
  - P8 State St L
- 1 - Exit
  - P9 McCormick E
  - P10 McCormick E
  - P11 State St E
  - P12 State St E
- 2 - Intersection
  - P13 McCormick x State St
- 3 - Sidewalk
- 4 - Parking
- 5 - Median

Origin
- NB
- SB
- EB
- WB

WB

Destination
- NB E
- SB E
- EB E

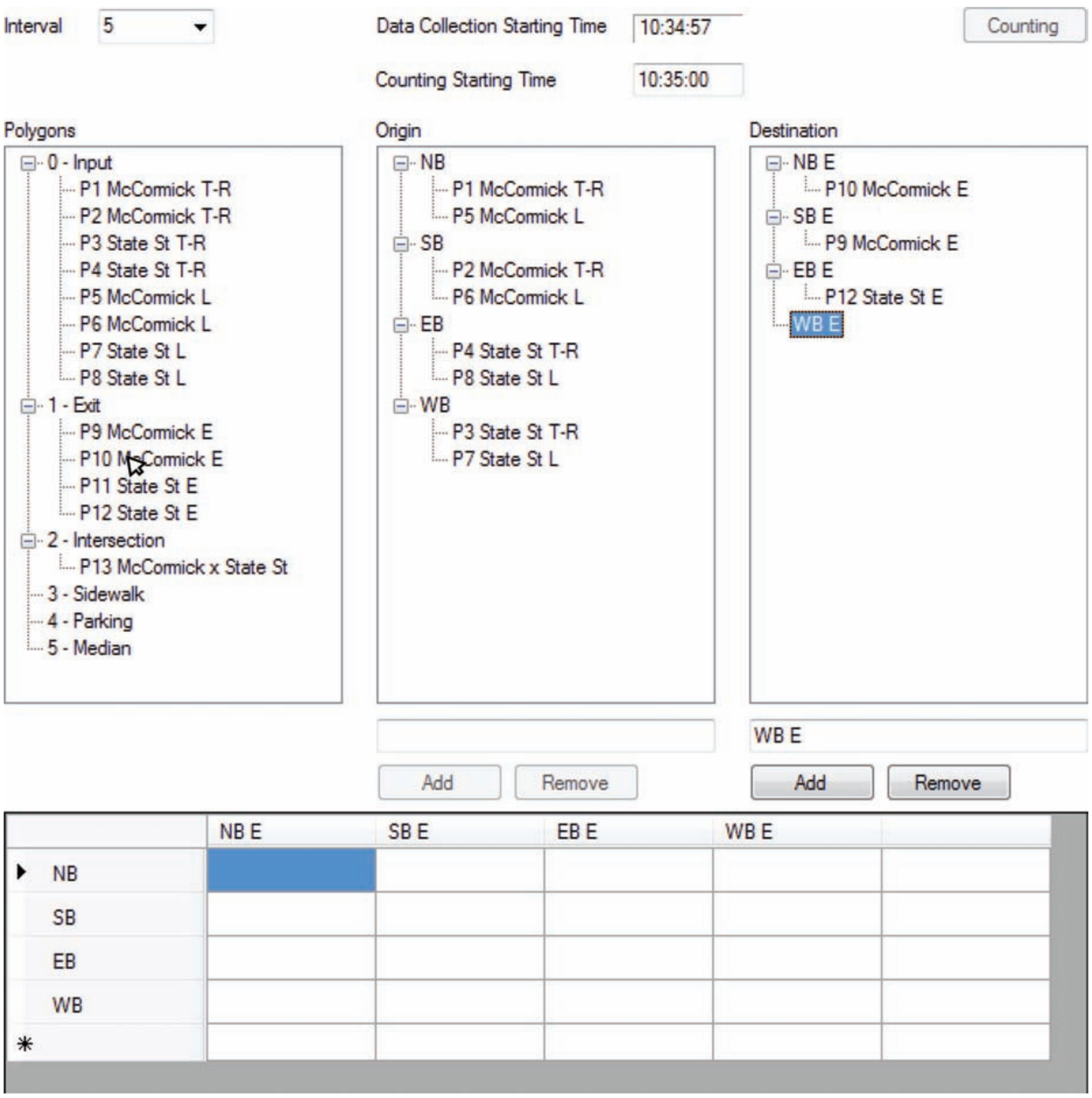
WB E

	NB E	SB E	EB E	
NB				
SB				
EB				
WB				
*				

Figure E.9 Creating destination areas.

The results are organized in a table. Each row corresponds to a time interval and each column correspond to a named movement from de origin-destination matrix.

Export pesults. To save the shown results table into a comma separated values file format, click on *Export*. The user is asked to provide a file name and a folder to which the results file is to be saved as shown in Figure E.15.



Interval 5

Data Collection Starting Time 10:34:57

Counting Starting Time 10:35:00

Polygons
- 0 - Input
  - P1 McCormick T-R
  - P2 McCormick T-R
  - P3 State St T-R
  - P4 State St T-R
  - P5 McCormick L
  - P6 McCormick L
  - P7 State St L
  - P8 State St L
- 1 - Exit
  - P9 McCormick E
  - P10 McCormick E
  - P11 State St E
  - P12 State St E
- 2 - Intersection
  - P13 McCormick x State St
- 3 - Sidewalk
- 4 - Parking
- 5 - Median

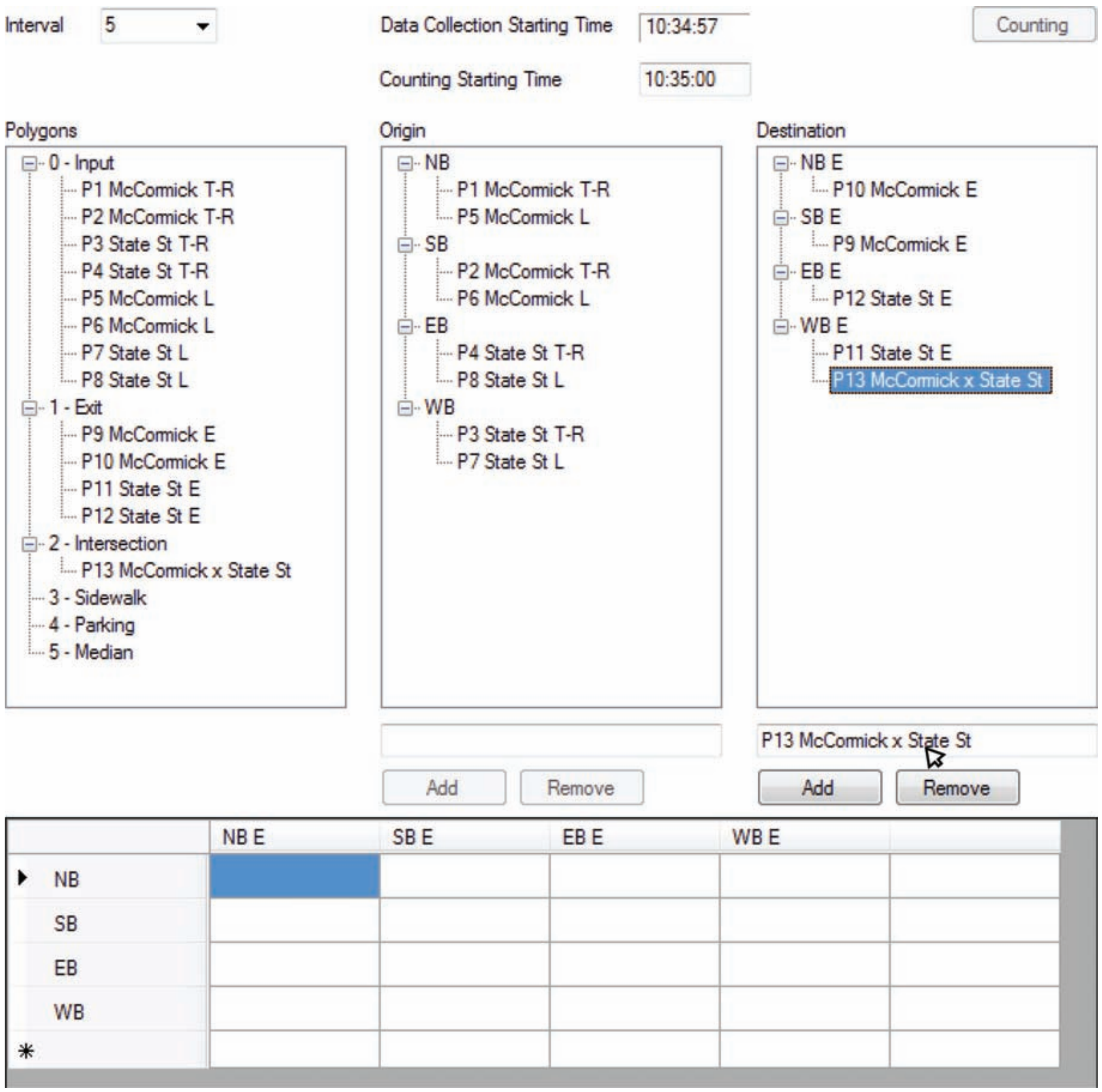
Origin
- NB
  - P1 McCormick T-R
  - P5 McCormick L
- SB
  - P2 McCormick T-R
  - P6 McCormick L
- EB
  - P4 State St T-R
  - P8 State St L
- WB
  - P3 State St T-R
  - P7 State St L

Destination
- NB E
  - P10 McCormick E
- SB E
  - P9 McCormick E
- EB E
  - P12 State St E
- WB E

WB E

	NB E	SB E	EB E	WB E	
NB					
SB					
EB					
WB					
*					

Figure E.10 Adding polygons to an origin approach or to a destination area.



Interval 5

Data Collection Starting Time 10:34:57

Counting Starting Time 10:35:00

Counting

Polygons
- 0 - Input
  - P1 McCormick T-R
  - P2 McCormick T-R
  - P3 State St T-R
  - P4 State St T-R
  - P5 McCormick L
  - P6 McCormick L
  - P7 State St L
  - P8 State St L
- 1 - Exit
  - P9 McCormick E
  - P10 McCormick E
  - P11 State St E
  - P12 State St E
- 2 - Intersection
  - P13 McCormick x State St
- 3 - Sidewalk
- 4 - Parking
- 5 - Median

Origin
- NB
  - P1 McCormick T-R
  - P5 McCormick L
- SB
  - P2 McCormick T-R
  - P6 McCormick L
- EB
  - P4 State St T-R
  - P8 State St L
- WB
  - P3 State St T-R
  - P7 State St L

Destination
- NB E
  - P10 McCormick E
- SB E
  - P9 McCormick E
- EB E
  - P12 State St E
- WB E
  - P11 State St E
  - P13 McCormick x State St

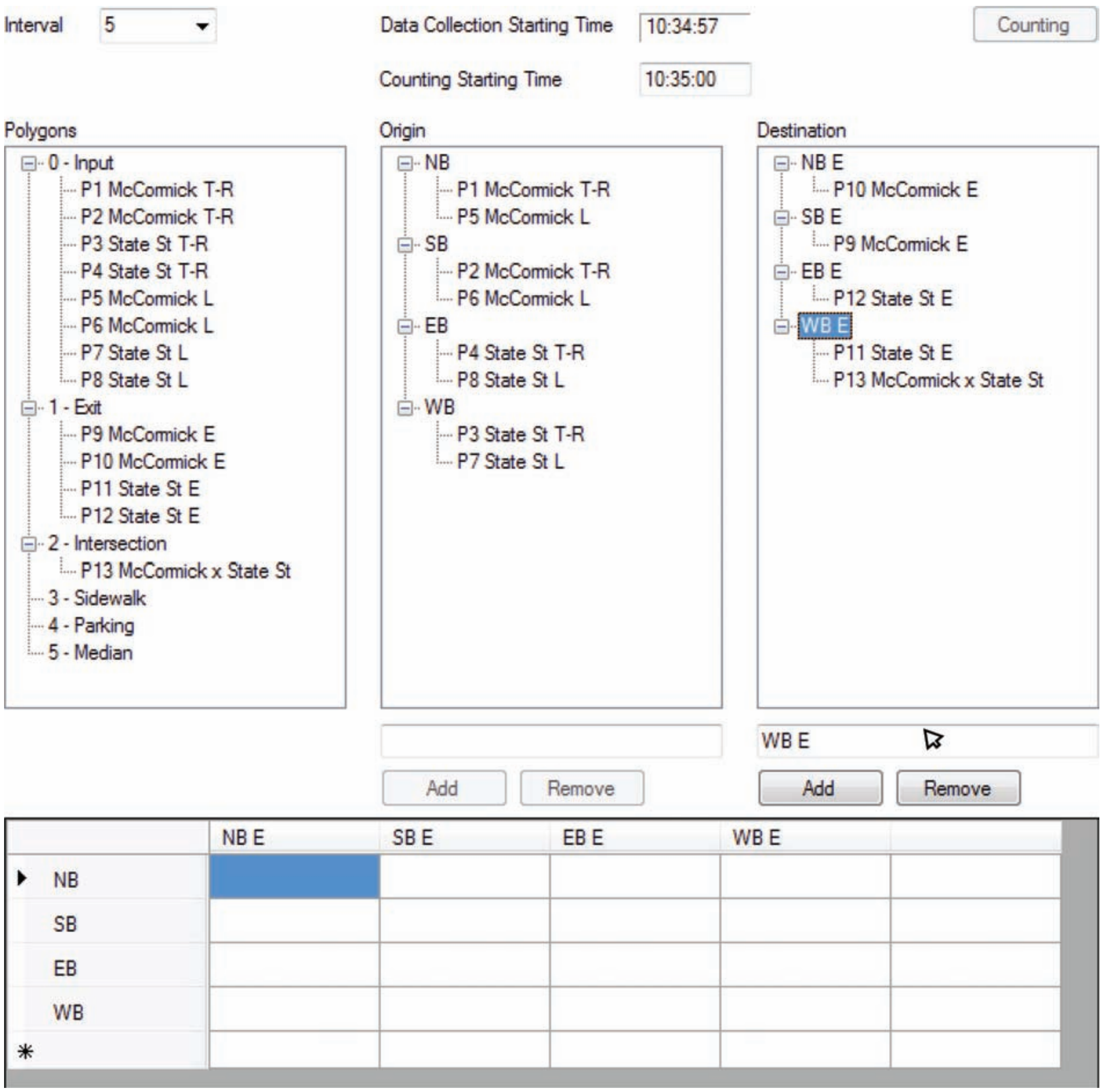
P13 McCormick x State St

Add Remove

Add Remove

	NB E	SB E	EB E	WB E	
NB					
SB					
EB					
WB					
*					

Figure E.11 Removing polygons from an origin approach or a destination area.


Interval 5

Data Collection Starting Time 10:34:57

Counting Starting Time 10:35:00

Counting

Polygons
- 0 - Input
  - P1 McCormick T-R
  - P2 McCormick T-R
  - P3 State St T-R
  - P4 State St T-R
  - P5 McCormick L
  - P6 McCormick L
  - P7 State St L
  - P8 State St L
- 1 - Exit
  - P9 McCormick E
  - P10 McCormick E
  - P11 State St E
  - P12 State St E
- 2 - Intersection
  - P13 McCormick x State St
- 3 - Sidewalk
- 4 - Parking
- 5 - Median

Origin
- NB
  - P1 McCormick T-R
  - P5 McCormick L
- SB
  - P2 McCormick T-R
  - P6 McCormick L
- EB
  - P4 State St T-R
  - P8 State St L
- WB
  - P3 State St T-R
  - P7 State St L

Destination
- NB E
  - P10 McCormick E
- SB E
  - P9 McCormick E
- EB E
  - P12 State St E
- WB E
  - P11 State St E
  - P13 McCormick x State St

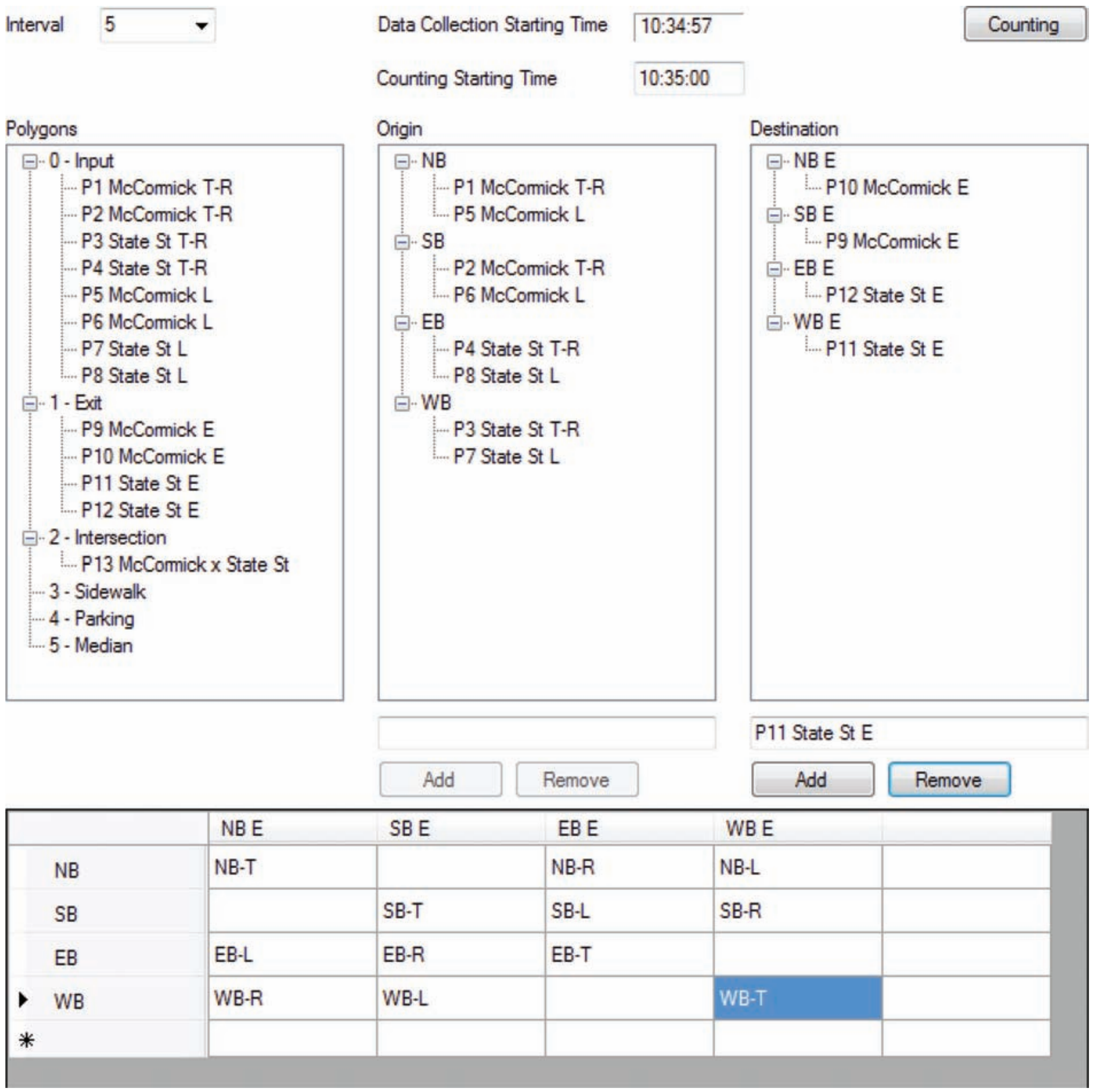
WB E

Add Remove

Add Remove

	NB E	SB E	EB E	WB E	
NB					
SB					
EB					
WB					
*					

Figure E.12 Removing an origin approach or a destination area.


Interval 5

Data Collection Starting Time 10:34:57

Counting Starting Time 10:35:00

Polygons
- 0 - Input
  - P1 McCormick T-R
  - P2 McCormick T-R
  - P3 State St T-R
  - P4 State St T-R
  - P5 McCormick L
  - P6 McCormick L
  - P7 State St L
  - P8 State St L
- 1 - Exit
  - P9 McCormick E
  - P10 McCormick E
  - P11 State St E
  - P12 State St E
- 2 - Intersection
  - P13 McCormick x State St
- 3 - Sidewalk
- 4 - Parking
- 5 - Median

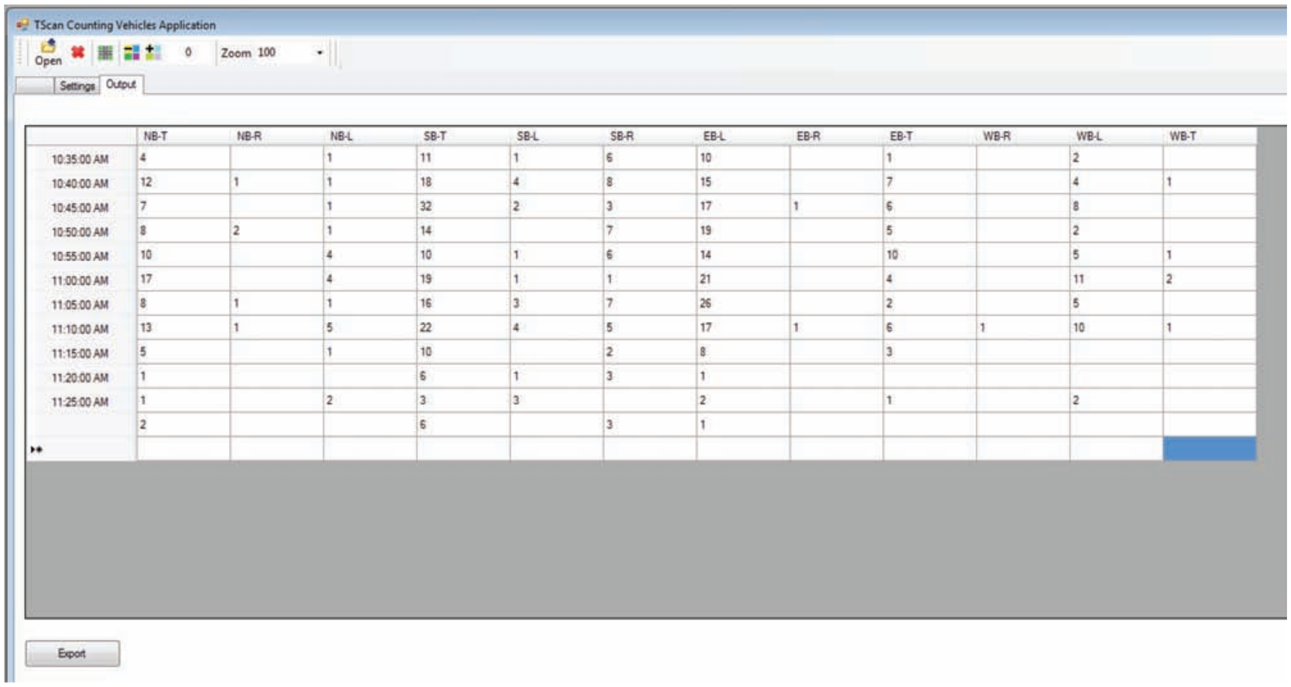
Origin
- NB
  - P1 McCormick T-R
  - P5 McCormick L
- SB
  - P2 McCormick T-R
  - P6 McCormick L
- EB
  - P4 State St T-R
  - P8 State St L
- WB
  - P3 State St T-R
  - P7 State St L

Destination
- NB E
  - P10 McCormick E
- SB E
  - P9 McCormick E
- EB E
  - P12 State St E
- WB E
  - P11 State St E

P11 State St E

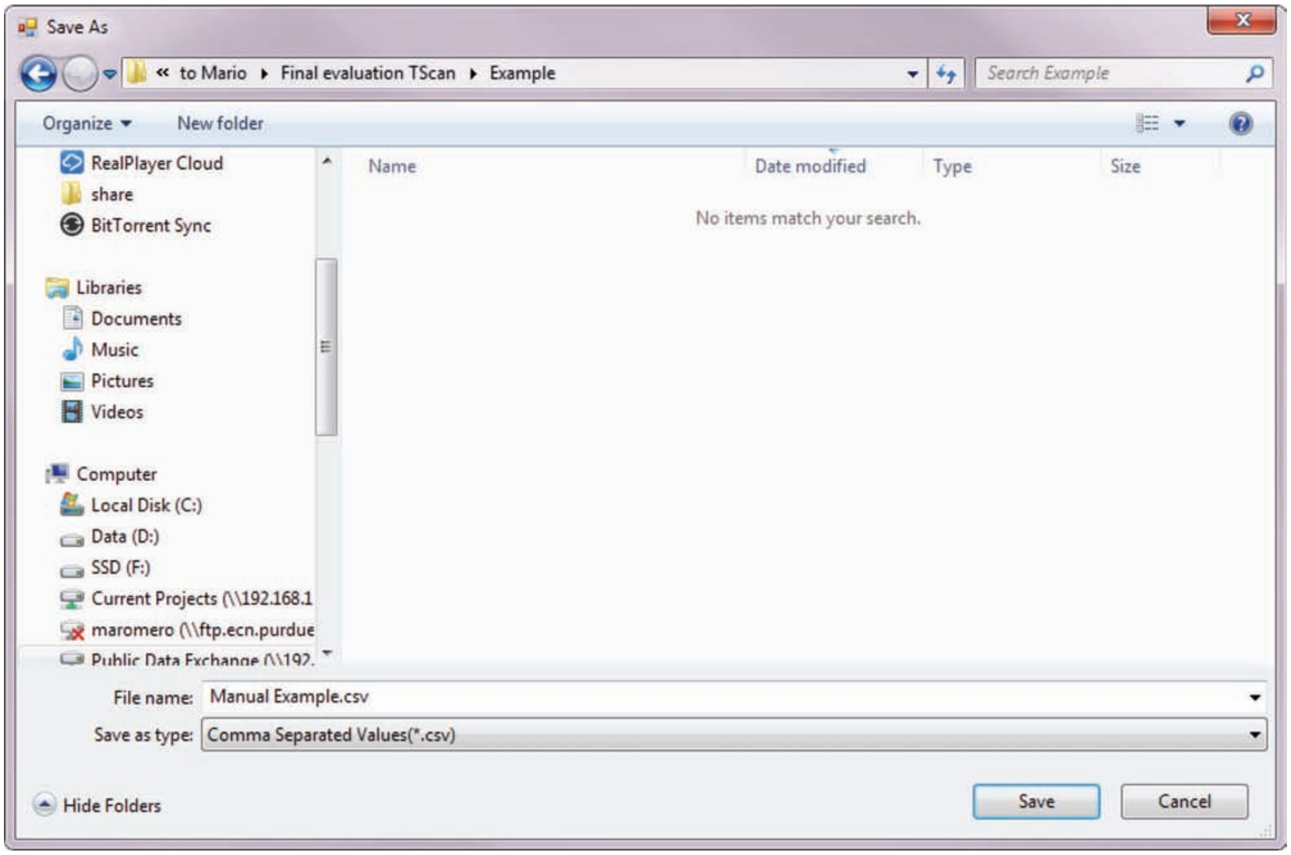
	NB E	SB E	EB E	WB E	
NB	NB-T		NB-R	NB-L	
SB		SB-T	SB-L	SB-R	
EB	EB-L	EB-R	EB-T		
WB	WB-R	WB-L		WB-T	
*					

Figure E.13 Setting movement names.



	NB-T	NB-R	NB-L	SB-T	SB-L	SB-R	EB-L	EB-R	EB-T	WB-R	WB-L	WB-T
10:35:00 AM	4		1	11	1	6	10		1		2	
10:40:00 AM	12	1	1	18	4	8	15		7		4	1
10:45:00 AM	7		1	32	2	3	17	1	6		8	
10:50:00 AM	8	2	1	14		7	19		5		2	
10:55:00 AM	10		4	10	1	6	14		10		5	1
11:00:00 AM	17		4	19	1	1	21		4		11	2
11:05:00 AM	8	1	1	16	3	7	26		2		5	
11:10:00 AM	13	1	5	22	4	5	17	1	6	1	10	1
11:15:00 AM	5		1	10		2	8		3			
11:20:00 AM	1			6	1	3	1					
11:25:00 AM	1		2	3	3		2		1		2	
	2			6		3	1					

Figure E.14 Counting results.



File name: Manual Example.csv

Save as type: Comma Separated Values(*.csv)

Figure E.15 Export results.

APPENDIX F. TSCAN PROTOTYPE SPECIFICATIONS

The TScan prototype hardware requirements are included in the following section.

F.1 Sensors and Cameras

F.1.1 LiDAR

The **Velodyne HDL-64E S3** increases the capabilities of the currently used LiDAR in the research unit (see Figure F.1). It has 5 -20 Hz user-selectable frame rate instead from 5 – 15 Hz. It increases the number of points per second output rate from 1.33 million to over 2.2 million, and the angular resolution (azimuth) changes from 0.09 to 0.08 degree.

The field of view remains as full 360 HFOV by 26.8 VFOV as well as the 100 MBPS Ethernet interfacing.

F.1.2 IMUs

TScan will require an IMU that accurately measure the orientation of the LiDAR sensor. The IMU specifications are given in the Table F.1.

F.1.3 GPS

Taking into account that the LiDAR sensor can synchronize its data with precision GPS-supplied time pulses over a dedicated RS-232 serial port, the GPS device (Figure F.2) must have the following characteristics:

- Issue a once-a-second synchronization pulse over a dedicated wire.
- Configure an available RS-232 serial port to issue a once-a-second $GPRMC NMEA record.
- Issue the sync pulse and NMEA record sequentially.

Figure F.1 Velodyne HDL-64E S3 LiDAR.

- The sync pulse length is not critical (typical lengths are between 20ms and 200ms). Start the $GPRMC record between 50ms and 500ms after the end of the sync pulse.
- Configure the $GPRMC record either in the hhmmss or hhmmss.s format.

The GARMIN GPS 18x LVC can be used for the time synchronization since it can output data in NMEA 0183 format (industry standard). It provides a pulse-per-second logic-level output with a rising edge aligned to within 1 microsecond of UTC time.

F.1.4 Camera

In order to cover the entire intersection, a fisheye IP camera with a field of view of 360 degrees should be used (see Figure F.3). No pan and tilt is required.

TABLE F.1
IMU specifications.

Sensors	***Accelerometer Specifications***	
	Number of axis:	3
	Range:	±16 g
	In-Run Bias Stability	<0.04 mg
	Linearity:	<0.5° FS
	Noise Density:	<0.14 mg/√Hz
	Bandwidth:	260 HZ
	Alignment Error:	±0.05°
	Resolution:	<0.5 mg
	Gyroscope Specifications	
	Number of axis:	3
	Range:	±2000°/s
	In-Run Bias Stability:	<10°/hr
	Linearity:	<0.1% FS
	Noise Density:	0.0035°/s √Hz
	Bandwidth:	256 Hz
	Alignment Error:	±0.05°
	Resolution:	<0.02°/s
	Magnetometer Specifications	
	Number of axis:	3
	Range:	±2.5 Gauss
	Linearity:	<0.1%
	Noise Density:	140 μGauss/√Hz
	Bandwidth:	200 HZ
	Alignment Error:	±0.05°
	Resolution:	1.5 Milligauss
Communications	Serial RS-232 & TTL	
Angular resolution	<0.05 deg	
Output rate	Minimum 100 Hz	

Figure F.2 Garmin GPS receiver.

A 5 Megapixel sensor is recommended with a 30fps at 1080p Full HD.

F.2 Computer and Communications

F.2.1 Computer

The computer specifications are given in Table F.2.

F.2.2 External Storage

TScan should have two external storage units in order to transfer the required files and the output data from the TScan unit to the office.

The storage units should be at least 1 TB and with a USB 3-C connector to increase the transfer speed (e.g., Figure F.4).

F.2.3 Communication Devices

A standard Gigabit Ethernet switch with at least 8 ports should be used as well as a standard USB to RS232 converter.

F.3 Power Supply

The TScan unit must have multiple power supply alternatives, including direct connection to an external outlet, power from a generator, and an inverter as a temporary backup system. For equipment that requires DC current, it will receive power from the batteries. Finally, a AC/DC converter will recharge the batteries.

Table F.3 shows the power consumption estimate used to estimate the power requirements.

F.3.1 External Outlet

Taking into account that the estimated power consumption of the system is less than 1200 watts, a regular 110 VAC input is required. When the external power supply is connected, the generator should be disconnected so a selector switch should be included.

F.3.2 Power Generator

In order to have the ability to use TScan in places with no access to power, it is necessary to have a power generator. Since a generator requires fuel, a diesel power generator is recommended since it is more stable than gasoline and easier to store.

Figure F.3 Vivotek FE8181 5MP 360° fisheye camera.

Figure F.4 Samsung T3 Portable 1 TB USB 3.0 External SSD.

TABLE F.2
Computer specifications.

Processor	4th Generation Intel Core^{TM} i7-5820K Processor or better
Operating System	Windows 7 or higher, 64-bit
Memory	32GB, 1666MHz or higher, DDR4
Hard Drive	3TB 7200 rpm Hard Drive SSD 512 Samsung EVO or better
Graphics Card	NVIDIA GeForce GT 950 2GB GDDR5
Optical Drive	DVD-RW Drive (Reads and Writes to DVD/CD)
Ports	4 USB 3.0 ports 4 USB 2.0 Ports 1 HDMI 1 Display Port 1 RJ-45 (10/100/1000Base)
Slots	4 DIMMs
Wireless	802.11bgn + Bluetooth 4.0, 2.4GHz, 1x1
Keyboard	Wired Keyboard
Mouse	Optical Mouse

The power generator should be able to provide at least 1000 watts for at least 4 days without refilling the tank (e.g., Figure F.5). It must be mounted on a trailer to facilitate transportation. If a portable light tower is used, the power generator, trailer, and mast are included.

TABLE F.3
Power consumption estimation.

Equipment	Power Consumption	Voltage
LiDAR	80 watts	12-24 VDC
IMUs	0.7 watts	5 VDC
Computer (including monitor)	300 watts	110 VAC
External storage	30 watts	110 VAC
Camera	25 watts	24 VAC
Switch	20 watts	110 VAC
Router	30 watts	110 VAC
Batteries (charge)	200 watts	12 VDC
Total	685.7 watts	

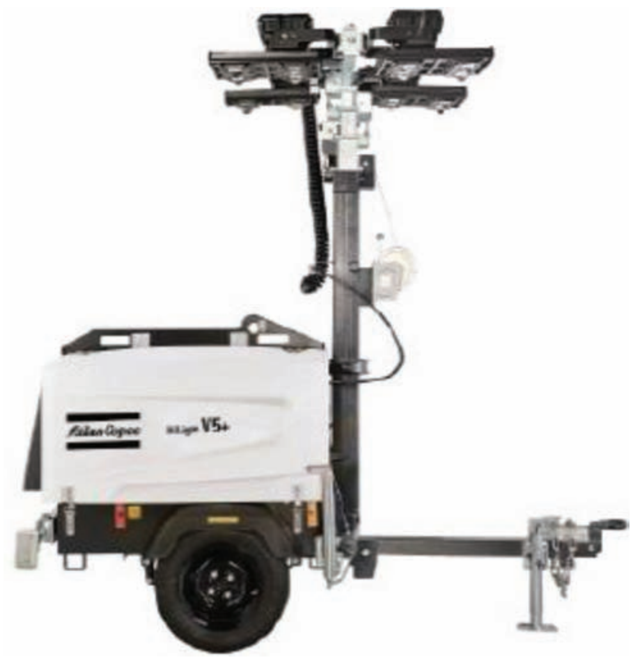


Figure F.5 Atals Copco HiLight V4.

F.3.3 Uninterruptible Power Supply

A true uninterruptible power supply (UPS) that gives at least 1.2 KVA should be used to generate the 110V from the batteries (e.g., Figure F.6). The batteries should last for 30 minutes at full consumption.

F.4 Infrastructure

F.4.1 Trailer

If a portable light tower is used the trailer with the power generator is already included. An extra housing for the electronic equipment is needed.

The trailer must have stabilizers for normal operation.

The trailer must have spring axle and tow hitch to conform to DOT requirements.

F.4.2 Mast

The mast must be able to rise up to 9 meters.

A vertical position for transportation is preferred.

The maximum load should be higher than 80 lbs.

F.4.3 Pan-Tilt Base

A pan tilt base to control the LiDAR orientation must be included if the mast doesn't have this option. The Pan-Tilt base specifications are shown in Table F.4.

TABLE F.4
Pan-Tilt base specifications.

Rotation	0 to 355 deg
Tilt	-30 to 30 deg or more
Maximum load	Minimum 60 lbs
Remote controller	Hard wired
Voltage	12-24 VDC / 110 VAC

Figure F.6 Tripp Lite SM2200RMXL2UP 2.2 KVA UPS.

APPENDIX G. DATA PORTAL—USER-FOCUSED RECOMMENDATIONS

The INDOT Division of Management Information Systems (MIS) has undertaken a project to develop a Crash Data Portal for the purpose of improved data access for various users. Currently, the Indiana State Police operate an online database called the Automated Reporting Information Exchange System (ARIES). However, ARIES's capabilities are limited in that it allows authorized users access to individual crash data rather than the desired, more comprehensive road-oriented database. By modernizing the delivery of crash and other data, a more diverse group of users can have their data needs met in a cost-effective manner. The initial discussions have concentrated on data acquired in real-time for short-term operational analysis and quick response; recent developments in the availability of dynamic data in real-time raise the possibility of integrating various types of dynamic data with the more traditional stable type, such as the data stored in crash records databases like ARIES. An effort to store data from different sources in a single depository or integrated multiple depositaries, and make them available to various users, on the other hand, requires carefully designed user specifications that include the types of users and their needs and the available sources of data in the format required. Consistency is desired, as it is crucial in developing a comprehensive, comparable database (Andreassen et al., 2013). In response, we propose to (1) develop users' specifications for the INDOT data portal with focus on safety-related data, and (2) develop a renewal and management process for the traditional stable safety-focused database.

G.1 Survey of Potential User's Data Needs

In order to ascertain the data needs of potential users, CRS submitted an electronic survey to 33 individuals, associated with INDOT, ICJI, and MPOs. The survey instructions suggested that the link be passed to other individuals who might find the idea of a data portal useful and see themselves as potential users.

After 3 weeks, twenty-eight responses were received. Twenty-five of the respondents identified themselves as being associated with INDOT. Two of them were from the Indiana Criminal Justice Institute and one self-identified as other/retired.

Consequently, the results and recommendations in this document for the most part reflect data usage expected to suit INDOT's needs. In particular transportation analysts, planners, data managers, researchers and designers, which constitute the most frequent occupations mentioned in the survey (see Figure G.1).

The datasets most often used and needed by the survey participants are displayed in Figure G.2. INDOT is already the source of 8 of the 10 most requested databases, but just as crash data is provided to INDOT by an external source (ARIES/Apriss), it may be useful for the portal to incorporate other databases available from other state agencies, like JTAC's E-Citations.

Data most often used or needed that are of restricted access (Figure G.3) include: Crash Data, Road Network Representation, Road Inventory Data, Signs and Signals Inventory, Pavement Data, Bridge Data, E-Citations, Detector Data, Weight in Motion Data, Road barriers Inventory and Travel Time data.

With exception of E-Citation and Travel Time data, INDOT already controls of most of the other restricted access datasets. Adding these two extra sources to the site would bring a great benefit to users, by centralizing most of the frequently used restricted access data in one place. Especially since the Electronic Citation and Warning System (eCWS) is maintained by the Indiana judicial branch, division of State Court Administration, and their contact person has expressed willingness to share data during TRCC meetings at the Criminal Justice Institute.

The data most often used or needed that are somewhat accessible to the public include: Annual Average Daily Traffic, Census Data, and Weather Data. The portal could include links to these public accessible sources.

From Table G.1, one can see that INDOT is the source of most datasets frequently required by the respondents of the survey, with the exceptions being Weather data, E-Citations and Travel Time data. With the possibility of bringing E-Citations to the portal, and with the establishment of links to public sites containing weather information, it may be also desirable to add travel time information to the portal, as about half of the respondents expressed that such data is used (5 users) or needed (9 users) on a regular basis.

The potential users surveyed were asked which formats they are most familiar with and prefer to use on their daily routines. Most users favored formats that can be easily read with Microsoft Office. These include excel native formats, csv and html. PDF is also a widely used standard used to provide tabulated or aggregated results, but is may be difficult to import PDF data content for analysis. ArcGIS compatible layer formats seem to be used by about a third of the respondents (Figure G.4.)

G.2 Review of Current Traffic Safety Data Portals

In an attempt to ascertain the state of the practice, CRS performed a review of multiple traffic safety data repositories and portals. These web sites provided traffic safety information in different formats and with different levels of aggregation.

This review was focused on evaluating the availability of the features listed on Table G.2. Functional related characteristics include public vs private access, aggregation levels, ways of exhibiting information and export capabilities. The analyzed data portals safety data at the national, state, county, and local level are shown in Table G.2.

At the national level, safety data portals from Great Britain, European Union, and USA were analyzed. Great Britain provides public safety data from 1979

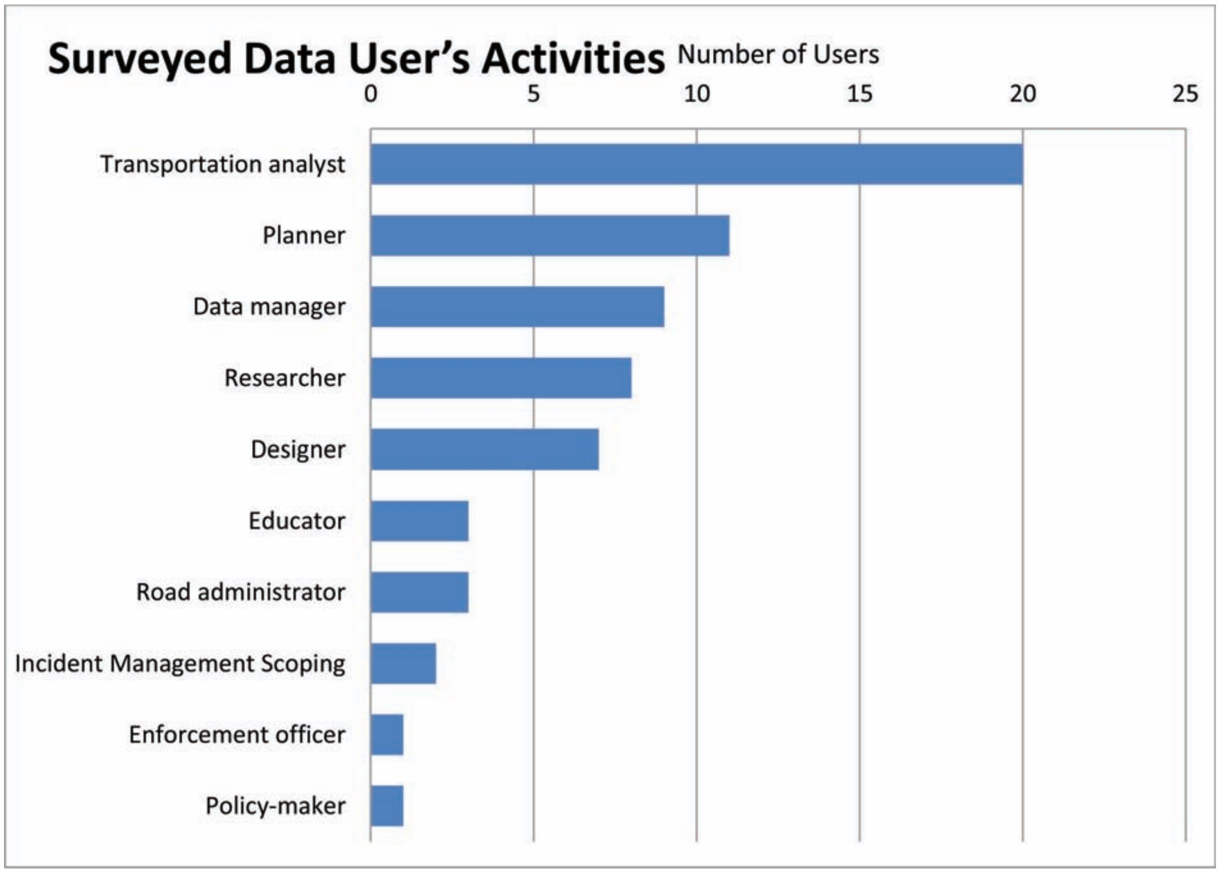


Figure G.1 Tripp data user's occupation.

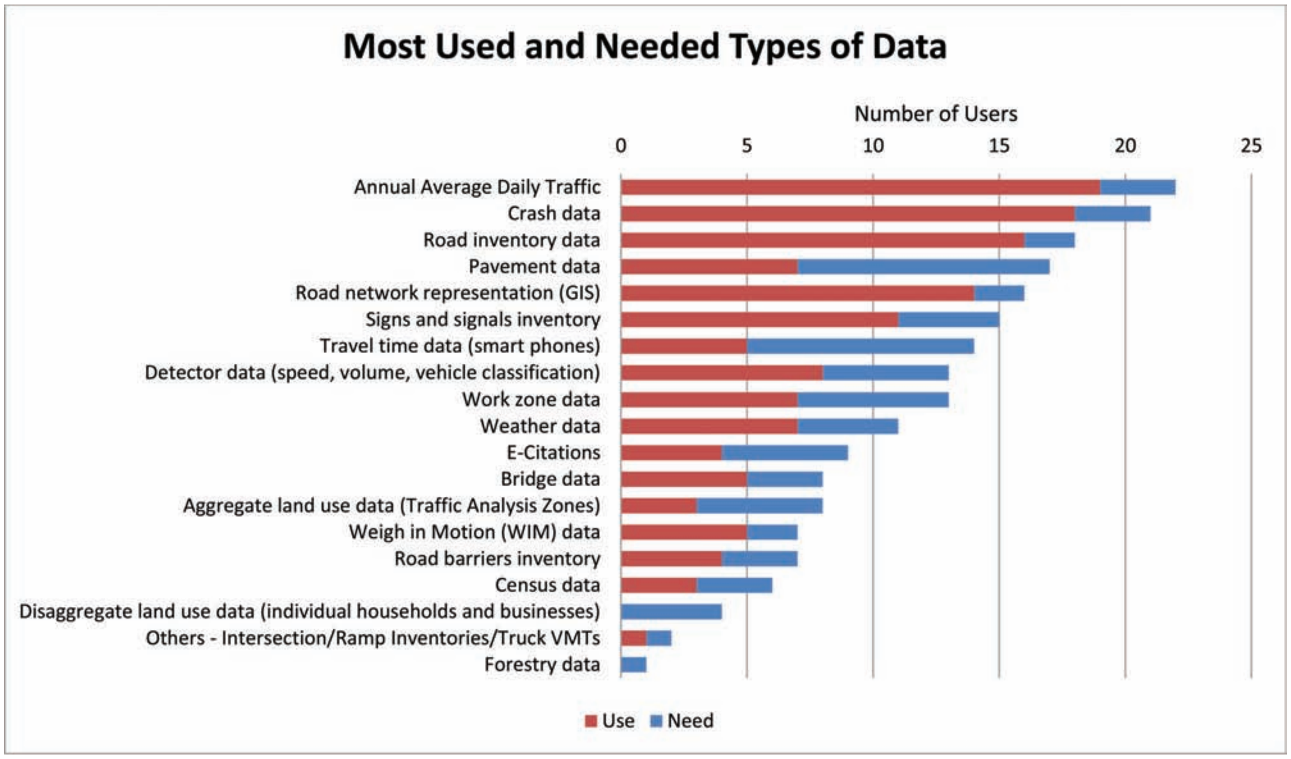


Figure G.2 Most requested types of data.

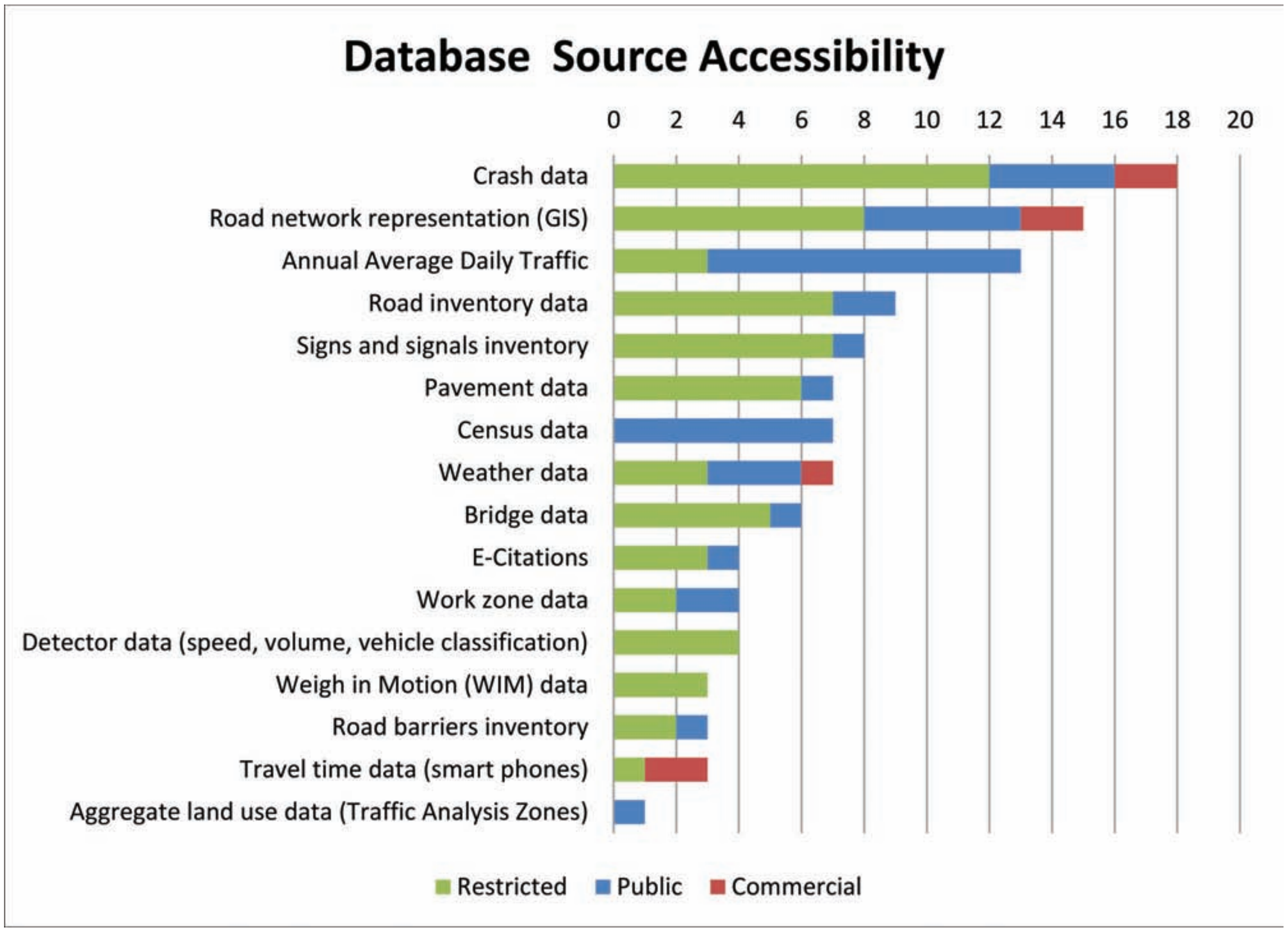


Figure G.3 Data accessibility.

until 2014. The data is aggregated at the crash level providing the circumstances of personal injury, the types of vehicles involved, and possible causes of the event. The data can be downloaded in .csv format. A new data portal was developed for providing a map interface. Both data portals are updated simultaneously by the project UK open government. On the other hand, safety data at the national level in the USA is provided by the NHTSA. The data in this portal is aggregated by crash severity and regional scope. It can be downloaded in excel or .txt format for additional processing. Finally, the data portal in the European Union is maintained by the statistical office of the European Union Eurostat. This safety data portal offers a user friendly interface. It provides tables, graphs, and maps for display safety statistics. The data can be exported in different formats such as excel, txt, pdf and others.

States dealt with data privacy differently, and a number of them provided both public and restricted areas in their portals. Public areas usually display pre queried results, reports and statistics; while private areas allow selected users to access to raw data. California has two different data portal from statistics reported by the California Highway Patrol and the additional one by the California Department of Public Health. Out of the 14 analyzed data portals from states, 8 sites aggregated results by severity, 11 sites aggregated results by geographical areas (county, district, city, etc.), and 3 provided results aggregated at the road level. From the perspective of types of output available, 13 sites provide statistical results in tables, 6 sites provided graphical results and 10 sites also provided results in the form of maps (Figure G.5) Lastly, the most common formats available in the visited states to download the data are pdf's (11 sites), excel (5 sites) and csv (2 sites).

At the local level, the cities of San Francisco, CA and Austin, TX were included in the study. In this case Austin offers open data for promoting transparency and opportunities for community app development. This city provides data at the crash level of aggregation including location, date, time, and type of the crash. The data is reported in tables and maps provided in a PDF format. The tables can be easily exported to .csv format or excel. San Francisco restricts the access of raw safety data. The data is displayed in tables on the online application. However, the data cannot be downloaded.

TABLE G.1
Data sources.

Data Sources	Crash Data	Annual Average Daily Traffic	Road Inventory Data	Road Network Representation	Signs and Signals Inventory	Census Data	Detector Data (speed, volume, vehicle type)	Bridge Data	Weather Data	E-Citations	Travel Time Fata (smart phones)
ARIES/APRISS	12								1	2	
Census Bureau						4					
CRS / Purdue			1	1	1						
District Inventory			1		2						
ESRI ArcGIS				1							
INDOT	3	11	7	4	5		4	5	1		
INRIX / Vendor											2
JTAC										1	
Multiple Sources						1					
News / Media									1		
NHTSA (FARS)	1										
NWS / InTelicast									1		
OPO Database										1	
State of Indiana Map		1		2			1				
Traffic Count Database System (TCDS)		1									

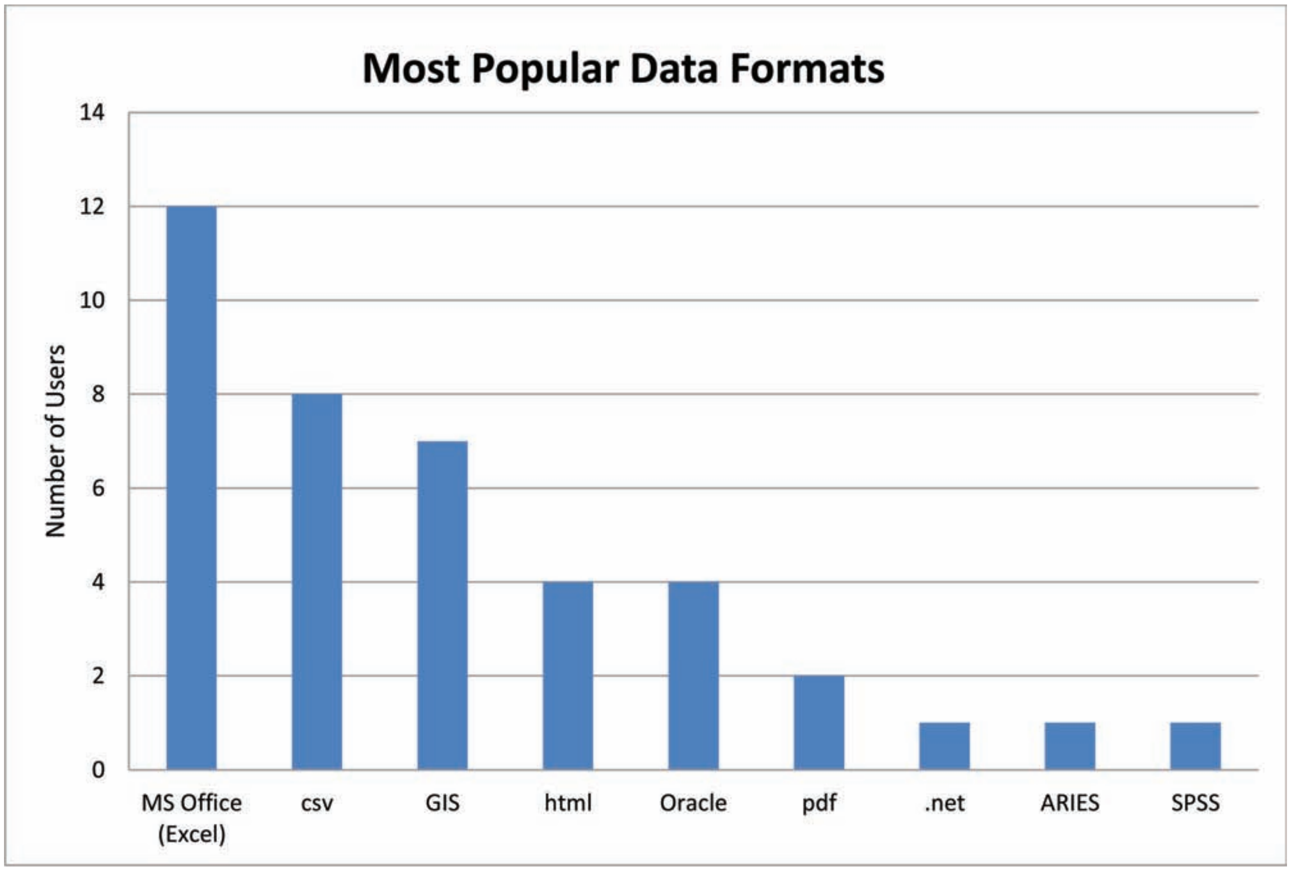


Figure G.4 Most popular data formats.

G.3 Recommendations

G.3.1 Accessibility

It is suggested that the portal have two areas. One public and one password restricted. The public area may contain not only reports and statistics for public consumption but also certain types of data which are already openly available, as for instance, the annual average daily traffic data.

The password protected area will contain data which INDOT may want to restrict from public view for a number of reasons, including contractual agreements.

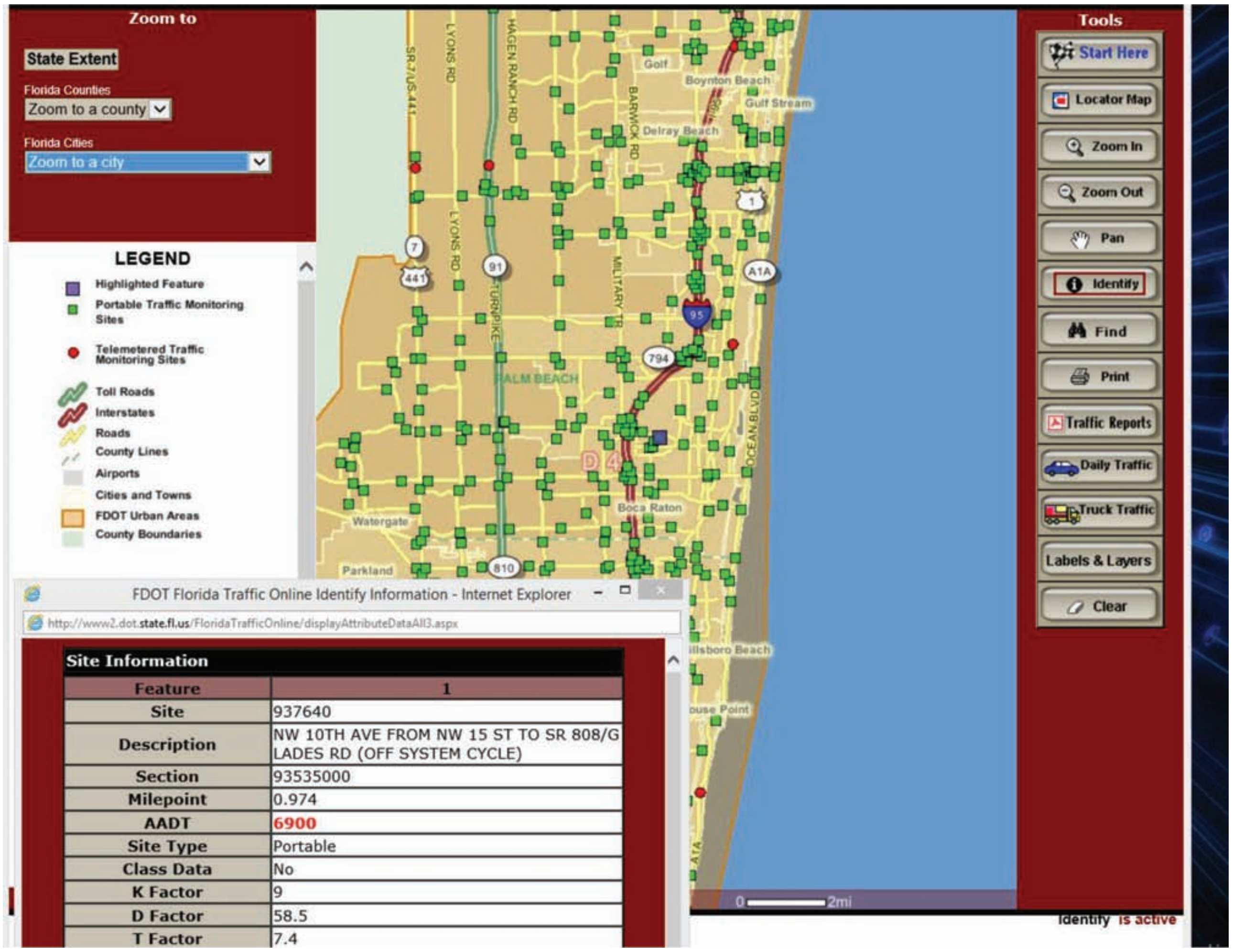
Depending on INDOT's needs, two password protected areas, rather than only one, might be desirable. One area would have data scrubbed from personal identifiers and one area would have personal identifiers intact. Those identifiers may be desirable in certain projects where the linkage of different databases might be necessary.

G.3.2 Datasets

As a data portal, the site should provide access to the datasets most frequently used and needed by INDOT associated transportation analysts, planners, data managers, researchers and designers. These were determined to be:

- Crash Data
- Road Network Representation
- Annual Average Daily Traffic
- Road Inventory Data
- Signs and Signals Inventory
- Pavement Data
- Bridge Data
- E-Citations
- Detector Data
- Weight in Motion Data
- Road Barriers Inventory
- Travel Time Data

Ideally, for the benefit of the user, the content of different datasets should be linked in a way that it would facilitate the extraction and processing of all related tables and variables. Under this premise, a crash should contain the index number of the associated segment or intersection in the road network representation, as well as links to the corresponding records in the road inventory database, the pavement dataset, the signs and signals inventory dataset, etc.



FDOT Florida Traffic Online Identify Information - Internet Explorer

http://www2.dot.state.fl.us/FloridaTrafficOnline/displayAttributeDataAll3.aspx

Site Information

Feature	1
Site	937640
Description	NW 10TH AVE FROM NW 15 ST TO SR 808/G LADES RD (OFF SYSTEM CYCLE)
Section	93535000
Milepoint	0.974
AADT	6900
Site Type	Portable
Class Data	No
K Factor	9
D Factor	58.5
T Factor	7.4

Figure G.5 Geographical visually interactive refinement of scope.

This linked storage would allow the user to have all needed information at the time of the extraction, rather than having to post process the data and make the proper associations.

G.3.3 Interface

For the purpose of data extraction, it is important that the interface have a number of elements:

- Selection of the scope for the extraction. The scope may be of geographical nature (state, district, county, township, and municipality) or it may be a roadway corridor defined by road segments and intersections within specific coordinates/mileposts. Pull down menus with the available options for the different scope options should be available.
- Geographical view and refinement of the selected scope. On part of the screen, an interactive map that reflects choices made via menus can be used both as feedback for the user choices and also as a means to interactively refine the scope choices (by using the mouse to delimit areas).
- Time period. The user should be able to define the time period used to extract yearly or periodic data.
- Choice of Datasets. Once the scope is selected, the user should then be able to select from among the available databases, those which are needed for his project. These databases could be available in a check list or a pull down menu with selectable items.
- Choice of variables. For each dataset selected, the user should have the option to select which variables are needed for the extraction.

The previously mentioned concept of having datasets linked will facilitate the extraction of related sets of records across the multiple datasets. And because the records extracted will contain the corresponding indexes to other matched records in different database tables, the user will be able to later integrate the data for analysis in a more efficient and error-free way.

G.3.4 Aggregation

The user should be given the option to tabulate and aggregate crashes according to a number of criteria.

TABLE G.2
Data portals in the US and abroad.

Locality (linked to the online source)	Range	Access		Aggregation				Display Format			Download Format				
		Public	Restricted	Crash Level	By Severity	Geographical	Road Level	Graphics	Tables	Maps	Csv	Excel	Pdf	Text	Other
Austin Texas	City	Y		Y					Y	Y	Y	Y	Y		Y
San Francisco	City		Y	Y	Y				Y						
Great Britain	Country	Y		Y			Y		Y		Y	Y		Y	
European Union	Multiple Countries	Y			Y	Y		Y	Y	Y	Y	Y	Y	Y	
USA	Country	Y			Y	Y			Y			Y		Y	
England	Country	Y		Y			Y	Y	Y	Y	Y	Y		Y	
California	State	Y			Y	Y			Y			Y	Y		
California	State	Y			Y	Y			Y			Y	Y		Y
Florida	State	Y	Y	Y		Y			Y	Y			Y		
Georgia	State	Y			Y	Y	Y		Y	Y					Y
Illinois	State		Y			Y		Y	Y	Y					
Maryland	State	Y		Y					Y	Y	Y	Y	Y		Y
Massachusetts	State	Y		Y		Y			Y	Y		Y	Y		
Missouri	State	Y			Y	Y		Y	Y			Y	Y		
New Jersey	State	Y			Y	Y		Y	Y	Y			Y		
New York	State	Y			Y	Y		Y	Y	Y			Y		
Ohio	State	Y	Y	Y		Y		Y		Y			Y		
Oregon	State	Y			Y	Y	Y	Y	Y	Y			Y		
Virginia	State	Y		Y	Y				Y	Y			Y		
Indiana	State		Y	Y			Y		Y		Y				Y

At a minimum, these tabulation criteria should include crash severity, light conditions, manner of collision, truck involvement, weather conditions, and single vs multiple vehicle crashes. Ideally all crash variables should be available for selection.

Besides the tabulation by crash characteristics, aggregation should also be available at geographical or road element (segment/intersection) level.

G.3.5 Exporting Formats

A number of standard formats should be adopted for the different types of output generated at the portal:

- Tables of data extracted from datasets: csv, xlsx (excel), html
- Tables with aggregated results: csv, xlsx (excel), html
- Preprocessed reports and statistics for public consumption: pdf
- Road Network Representation Layer: ArcGIS shape (shp) file

About the Joint Transportation Research Program (JTRP)

About This Report

An open access version of this publication is available online. This can be most easily located using the Digital Object Identifier (doi) listed below. Pre-2011 publications that include color illustrations are available online in color but are printed only in grayscale.

The recommended citation for this publication is:

Tarko, A. P., Ariyur, K. B., Romero, M. A., Bandaru, V. K., & Lizarazo, C. G. (2016). *TScan: Stationary LiDAR for traffic and safety studies—Object detection and tracking* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2016/24). West Lafayette, IN: Purdue University. http://dx.doi.org/10.5703/1288284316347