JOINT TRANSPORTATION
RESEARCH PROGRAM
INDIANA DEPARTMENT OF TRANSPORTATION
AND PURDUE UNIVERSITY

TScan–Stationary LiDAR for Traffic and Safety Applications: Vehicle Interpretation and Tracking

Andrew P. Tarko, Mario A. Romero,
Vamsi Krishna Bandaru, Cristhian Lizarazo

SPR-4102 • Report Number: FHWA/IN/JTRP-2021/31 • DOI: 10.5703/1288284317402

RECOMMENDED CITATION

Tarko, A. P., Romero, M. A., Bandaru, V. K., & Lizarazo, C. (2021). *TScan–Stationary LiDAR for traffic and safety applications: Vehicle interpretation and tracking* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2021/31). West Lafayette, IN: Purdue University. https://doi.org/10.5703/1288284317402

AUTHORS

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

Mario A. Romero, PhD
Research Scientist and Program Manager for the Center for Road Safety
Purdue University

Vamsi Krishna Bandaru
Graduate Research Assistant
Lyles School of Civil Engineering
Purdue University

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

JOINT TRANSPORTATION RESEARCH PROGRAM

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

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

NOTICE

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. FHWA/IN/JTRP-2021/31	**2. Government Accession No.**	**3. Recipient's Catalog No.**
4. Title and Subtitle TScan–Stationary LiDAR for Traffic and Safety Applications: Vehicle Interpretation and Tracking		**5. Report Date** November 2021 **6. Performing Organization Code**
7. Author(s) Andrew P. Tarko, Mario A. Romero, Vamsi K. Bandaru, and Cristhian Lizarazo		**8. Performing Organization Report No.** FHWA/IN/JTRP-2021/31
9. Performing Organization Name and Address Joint Transportation Research Program Hall for Discovery and Learning Research (DLR), Suite 204 207 S. Martin Jischke Drive West Lafayette, IN 47907		**10. Work Unit No.** **11. Contract or Grant No.** SPR-4102
12. Sponsoring Agency Name and Address Indiana Department of Transportation (SPR) State Office Building 100 North Senate Avenue Indianapolis, IN 46204		**13. Type of Report and Period Covered** Final Report **14. Sponsoring Agency Code**

15. Supplementary Notes

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

16. Abstract

To improve traffic performance and safety, the ability to measure traffic accurately and effectively, including motorists and other vulnerable road users, at road intersections is needed. A past study conducted by the Center for Road Safety has demonstrated that it is feasible to detect and track various types of road users using a LiDAR-based system called TScan. This project aimed to progress towards a real-world implementation of TScan by building two trailer-based prototypes with full end-user documentation. The previously developed detection and tracking algorithms have been modified and converted from the research code to its implementational version written in the C++ programming language. Two trailer-based TScan units have been built. The design of the prototype was iterated multiple times to account for component placement, ease of maintenance, etc. The expansion of the TScan system from a one single-sensor unit to multiple units with multiple LiDAR sensors necessitated transforming all the measurements into a common spatial and temporal reference frame. Engineering applications for performing traffic counts, analyzing speeds at intersections, and visualizing pedestrian presence data were developed. The limitations of the existing SSAM for traffic conflicts analysis with computer simulation prompted the research team to develop and implement their own traffic conflicts detection and analysis technique that is applicable to real-world data. Efficient use of the development system requires proper training of its end users. An INDOT-CRS collaborative process was developed and its execution planned to gradually transfer the two TScan prototypes to INDOT's full control. This period will be also an opportunity for collecting feedback from the end user and making limited modifications to the system and documentation as needed.

17. Key Words LiDAR-based tracking, intersection traffic analysis, traffic conflicts, TScan	**18. Distribution Statement** No restrictions. This document is available through the National Technical Information Service, Springfield, VA 22161.

19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 103 including appendices	**22. Price**

EXECUTIVE SUMMARY

Introduction

To learn about traffic operations and improve traffic performance and safety, the ability to accurately measure traffic, including motorists and other vulnerable road users, at road intersections for sufficiently long periods is needed. The efficiency of collecting and analyzing intersection data is particularly important given the expectation of a growing presence of autonomous and connected vehicles in traffic. Project SPR-3831, conducted by the Center for Road Safety, has demonstrated that it is feasible to detect and track various types of road users (e.g., trucks, cars, pedestrians and bicycles) using a LiDAR-based system called TScan. The mentioned study provided a set of specifications recommended to build a trailer-based prototype. This project, SPR-4102, was intended to progress towards a real-world implementation by further importing the design and performance of the original TScan unit and building two trailer-based prototypes with full end-user documentation.

Findings

The previously developed detection and tracking algorithms have been modified to increase their tracking accuracy, and the software has been converted from the research code written in MATLAB to its implementational version written in the C++ programming language. This conversion increased the real-time execution capabilities of TScan.

Two trailer-based TScan units have been built. Although they follow the operational specifications, due to the unavailability of the LiDAR sensor recommended in specifications, a two-sensor setup was selected as an alternative. To decide the types of sensors and their relative positioning in the TScan head, we simulated the data points' density in the individual fields of view of the candidate sensors. The design of the prototype was iterated multiple times to account for component placement, ease of maintenance, etc. The part of the prototype containing all the sensors is detachable from the trailer for easy storage in a secure location.

The expansion of the TScan system from a one single-sensor unit to multiple units with multiple LiDAR sensors necessitated transforming all the measurements made in their local reference systems into a common spatial and temporal reference frame. This process also included a GPS-based time synchronization method.

One of the envisioned uses of the TScan system is detecting traffic conflicts, which requires sufficiently accurate positions and velocities of objects moving within the system's field of view. The signal processing algorithms developed as part of SPR-3831 were modified or replaced to significantly enhance tracking quality. This modification has significantly reduced the false positives conflicts detection. The evaluation results are provided in the research report.

Engineering applications for performing traffic counts, analyzing speeds at intersections, and visualizing pedestrian presence data were developed. The limitations of the existing SSAM for traffic conflicts analysis with computer simulation prompted the research team to develop and implement its own traffic conflicts detection and analysis technique applicable to real-world data.

Implementation

Two trailer-based prototypes were built as part of this project. The algorithms were implemented in C++ with extensive use of parallel processing and careful memory management to achieve real time tracking. A post processing module was implemented to further improve tracking accuracy and also allows data from multiple trailers to be combined into one set of results, thus making the TScan system scalable. The report also provides user manuals for setting and operating the TScan research unit and for the various engineering applications mentioned above.

Efficient use of the development system requires proper training of its end users. An INDOT-CRS collaborative process was developed and its execution planned to gradually transfer the two TScan prototypes to INDOT. This process involves a sequence of informational meetings and joint field data collection exercises of growing comprehensiveness. The role of CRS personnel will be gradually reduced in these exercises, while involvement of the INDOT end users will increase. This period will be also an opportunity for collecting feedback from the end user and for making limited modifications of the system, where possible, and in the documentation as needed.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This report presents a research project (SPR-4102) aimed at improving the existing Traffic Scanner (TScan) unit for intersection traffic data collection and analysis and developing a second TScan unit integrated with the first one to make the system applicable to large intersections. The project was funded by the Joint Transportation Research Program (JTRP) sponsored by the Indiana Department of Transportation and the Federal Highway Administration. The previous project SPR-3831 had proven the concept by deploying a single LiDAR sensor technology. The previous and current units identify stationary objects and areas (background) and track moving objects within a 200-ft range from the sensor location and estimate the objects' positions, speeds, and acceleration rates ten times per second. They also estimate the objects' dimensions and classify them into cars, trucks, and two-wheelers.

To facilitate the TScan data use in traffic and safety analysis, several engineering applications have been developed including counting traffic movements, measuring vehicles' speeds within an intersection, and identifying traffic conflicts.

The interest in further advancing and implementing the TScan system was expressed by INDOT in early 2016 after presenting the SPR-3831 results. In response to the positive feedback the JTRP executive committee approved Phase II of the TScan development process to build one TScan prototype for testing on the road, SPR-3831 was viewed as Phase I. After a successful experience with the built prototype and after confirming the benefit of integrating two units, Phase III was approved and executed. It included busing the second TScan prototype and integrating it with the first one.

This report covers the work and results of Phases II and III.

- Designing and building the first TScan prototype unit and its evaluation.
- Improving the first prototype based on the feedback from INDOT.
- Designing and building the second prototype.
- Coordinating in time and integrating spatially the results from the two prototypes.
- Incorporating traffic conflicts detection via data post-processing.
- Developing selected engineering applications.
- Developing end user's documentation for the TScan and engineering applications.

The following are the deliverables of this project.

1. Two trailer-based hardware prototypes tested and integrated.
2. Software for TScan setting and for supporting real-time data collection, processing, and handling.
3. Two engineering applications for basic traffic and traffic conflicts analysis.
4. Documentation including this report, TScan user manual, and engineering applications manuals.

The report is organized in a manner that emphasizes the various aspects of this project. The organization of this report is as follows.

Chapter 2 introduces the initial concept of TScan Prototype as envisioned and for which specifications were developed at the end of previous project, explains its evolution and outlines its final form that is implemented in the two prototypes built as part of this project.

Chapter 2.1 explains in light detail the evolution of TScan prototype design which includes the rationale behind the shift from the previously envisioned single LiDAR unit to a multi-LiDAR unit.

In Chapter 4, the overall process that a user goes through when trying to collect data with TScan prototype is presented.

A detailed description of the various changes made to the algorithms since the previous TScan feasibility report is provided in Chapter 5. A comparison is made to show the improvements brought about by these changes in Chapter 6.

The various engineering applications developed to take advantage of the rich microscopic simulation, like trajectories of real vehicles to perform targeted analysis, are showcased in Chapter 7 and detailed in a separate user manual.

SSAM or Surrogate Safety Assessment Model (SSAM) is an application developed by Siemens ITS with FHWA funding to automatically identify, classify and evaluate traffic conflicts or close encounters between two vehicles given a list of vehicle trajectories (Gettman et al., 2008). A discussion on the shortcoming of SSAM and the need to adopt a new conflicts identification technique is explained in Chapter 8.

A brief discussion on the knowledge gained during this project and recommendations for future work is presented in Chapter 9 and Chapter 10 concludes the report.

Appendices A, B, and C provide selected details of the system while Appendix D includes the user manual that explains in detail how to operate the TScan trailer unit to collect data with screen captures of the software used. Appendix E details the engineering applications developed by the Center for Road Safety to facilitate use of the TScan in engineering analyses.

2. CONCEPT

The feasibility of a LiDAR based traffic monitoring system named Traffic Scanner or TScan was shown in the JTRP project SPR-3831. This chapter explains the user-oriented concept of the TScan system, its evolution and current capabilities.

2.1 An Initial Concept—End User's Perspective

The first TScan system was conceptualized around the 64-laser LiDAR—a state-of-the-art sensor at the time of developing the first concept. The proposed

system was to be applied at road intersections and relatively short road segments in daylight and nighttime conditions; and even during atmospheric precipitation (light rain, snowfall, and fog) of intensity that allows the light beams to travel without excessive dispersion. The range of the LiDAR was such that it could cover a broad range of intersections.

Research was conducted using the Center for Road Safety's (CRS) Mobile Traffic Lab (MTL) shown in Figure 2.2. Based on the findings, specifications for a standalone prototype was developed.

The trailer-based prototype had to be easy to set up by a single operator and be sustainable for several days with limited supervision. The assumed primary source of information for this system was a single LiDAR unit and the video cameras included in the design were used only for inspecting traffic in short periods selected by the user after data collection to confirm the validity of the collected numeric data.

The processing of data had to be executed in real-time, including classification and tracking objects. The accuracy should be sufficient without human involvement and data processing, which was the primary source of savings and the increased practicality of the proposed TScan system.

The TScan output file should include all the traffic characteristics of individual vehicles in a convenient format which would then make 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 positions of the moving objects at a rate of at least 10 instants per second in addition to things like type of vehicle and item dimensions.

Furthermore, the format of the outcome of the tracking was to follow the SSAM format for simulated traffic with modifications that reflect various sources (measurement vs. simulation). For example, the "wire-based" motion of vehicles in typical simulation would be 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 recommended included four main components (Figure 2.1).

1. A trailer with a power supply. A trailer with stabilizing legs to transport the unit to the data collection site, set up the equipment in a suitable position, and provide sufficient physical support and protection against tempering and a power supply to provide continuous power for the duration of data collection with the option of using electrical energy if available at the site.
2. A telescoping mast to raise the sensors to desired height.
3. Sensors installed on a mount. A LiDAR sensor such as the HDL 64E by Velodyne Inc. to provide sufficiently dense and accurate data, a set of Inertial Measurement Units (IMU) to account of motion of the sensor due to vibrations and wind, a surveillance video camera to record the traffic flow for later inspection by the user if needed. These sensors should be mounted on a pan/tilt mechanism to allow for better coverage of the intersection or road segment under surveillance.
4. A computer with communication component and data storage. A computer with sufficient processing power to process the vast amounts of data generated by the LiDAR in real time, specific communication equipment to facilitate remote operation of the computer and sufficient data storage capacity to allow continuous data collection for at least 1 week.

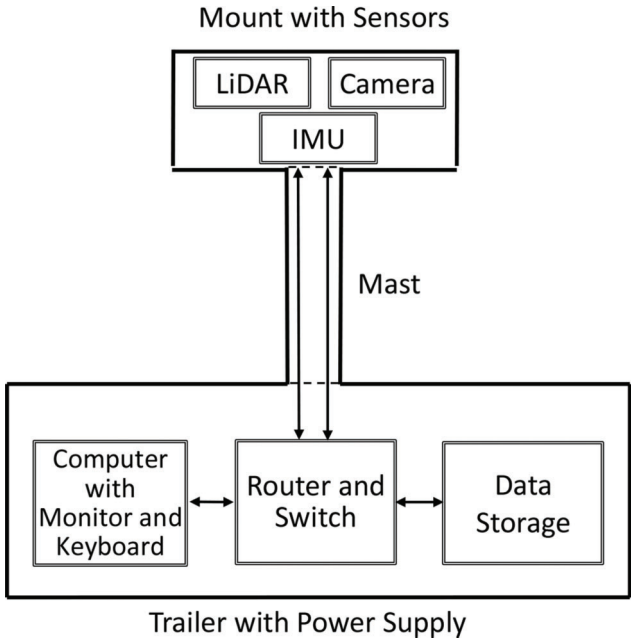


Figure 2.1 A general concept of the portable LiDAR-based system–TScan.

2.3 Concept Modification

2.3.1 Hardware Component Location Changes

One major concern identified in the early stage of developing the prototypes was security of the unit when operated unsupervised for prolong periods. To avoid tampering with the computer and data storage if placed in the trailer, these hardware components were placed in a separate enclosure and attached to the top of the mast together with the sensors. The self-contained unit was to be attached to the mast by the end user before the mast unfolding. The power supply and communication signal were provided via cables running from the trailer to the top of the mast. These cables were to be properly shielded with a steel conduit to protect the cables from being cut.

2.3.2 Move to Multiple LiDARs in a Single Unit

Most of the commonly available LiDAR sensors including the Velodyne HDL 64E had an advertised range of 100 m. This was only achievable if the target was

highly reflective in a Lambertian sense ($\pm$ Lambertian reflectance is the property that defines an ideal "matte" or diffusely reflecting surface. The apparent brightness of a Lambertian surface to the observer is same regardless of the observer's angle of view—e.g., unfinished wood exhibits roughly Lambertian reflectance whereas wood finished with glossy coat of polyurethane does not.). The effective range for tracking of 60 meters (Tarko et al., 2018) was sufficient to collect data with a single trailer unit at compact intersections of undivided roads. The effective range was not sufficient at large channelized intersections with medians on crossing roads. To overcome this challenge, the unit had to be equipped with multiple sensors and/or sensors with a longer range. Even then, the need for two units properly placed at an intersection seemed to be a practical solution under the shortage of long-range sensors that could cover wide areas. Two units would also help mitigate the occlusion problem difficult to solve with a single vantage point. Furthermore, multiple units could be deployed along road segments upstream of an interstation to monitor and analyze long queues. Thus, the scalability of the system became a core design objective.

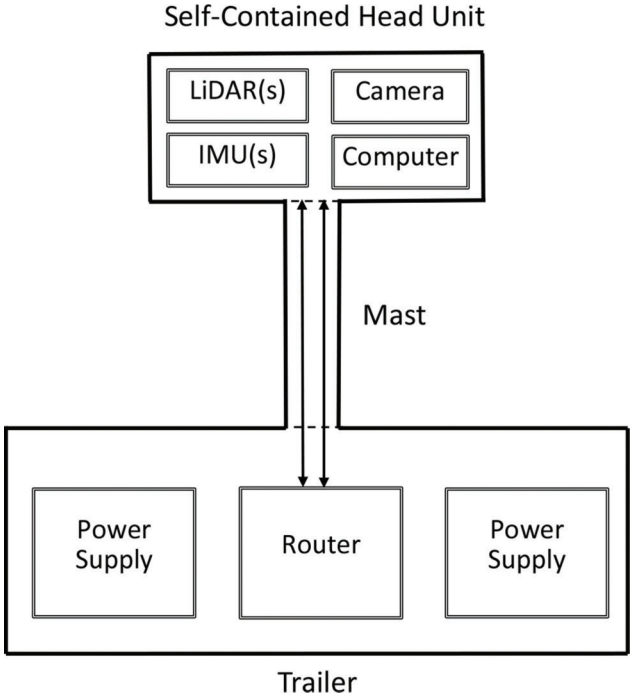


Figure 2.2 A modified general concept of the TScan prototype.

2.3.3 SSAM's Shortcomings and Need for Custom File Format

As stated in Section 2.1, the results of tracking were to follow SSAM format (Appendix B). The exitance of SSAM application developed by Siemens ITS with FHWA funding was the primary reason the file format was adopted. Given a list of trajectories, the application was capable of identifying traffic encounters. several shortcomings were discovered when this application was used extensively for TScan results during prototype development. The lack of support for different road user types was one such problem. SSAM does not distinguish between pedestrians and vehicles for instance and assumes that all road users are vehicles. Pedestrians move at exceptionally low speeds and have the ability to instantaneously stop, this is not the case for vehicles like cars. Thus, pedestrians need to be treated differently and SSAM does not do so and has no provision to even mention the classification of the road user. Problems with SSAM's methodology in detecting traffic conflicts are detailed in Chapter 8.

To overcome the above shortcomings, a two-pronged solution was adopted. A new file-format, namely TScan File Format or TSFF (refer to Appendix C) that preserves a lot more information than what SSAM does and a new conflicts identification method explained in detail in Chapter 8 was also developed.

2.4 Integration of Data Collection and Processing with Multiple Units

A block diagram of the final concept implemented in the prototypes is shown in Figure 2.2. While the hardware requirements of the system largely remained the same, some components of the hardware were relocated within the system.

The computer (along with data storage) was moved to a self-contained head unit. This change provided another benefit that the self-contained head unit could be made detachable and thus allow for secure storage of the more expensive components like LiDARs inside a building when not in use.

Including multiple sensors in a single unit introduced new challenges. The measurements with these units had to be reconciled within a common temporal and spatial reference baseline since each LiDAR sensor has its own frame of reference. The absolute orientation of the LiDARs in 3D is computed by using measurements from the LiDARs' built-in Inertial Measurement Unit and the feedback from the pan servo motor. In order to be able to effectively cover the intersection, the sensors are mounted on top of a base that can be panned and tilted. The position of sensors has to be accounted for when computing the transformation matrix to translate points from one reference frame to another. A complete description of the transformation method is provided in Section 5.3.

Once the common spatial reference frame is established for points, the measurement times must be adjusted to a common temporally reference as well. Initially, the rotation of the LiDAR was the base for the synchronization. LiDARs are known for not keeping a constant rotation frequency that leads to a time drift in the order of 5 seconds over a period of 1 day. Yet this method was acceptable so long as only a single unit was involved. One LiDAR sensor would be designated as the primary sensor. Data collected from

all the other sensors within one rotation of the primary sensor constituted a "frame." For the sake of tracking, it is assumed that the data within a frame is measured in the same instance of time.

While this method worked for a single unit, it was insufficient for multiple masts. Instead, a GPS-based time reference was used. In both the developed units, a GPS receiver delivers a Pulse Per Second (PPS) signal. It is used as a reference timer to which the primary LiDAR's internal timer/clock is synchronized. Data obtained from all the sensors within a specified unit of time (for instance 0.1 seconds when the rotation of the LiDARs are set to 10 Hz) as observed by the primary LiDAR constitutes a "frame."

Each unit generates about 35 gigabytes of data per hour. Transferring wirelessly this amount of data in real time was deemed infeasible given the power and space constraints, especially if the system is to be expanded beyond two units. Currently each individual unit collects data, performs tracking and stores the intermediate results in the TSFF format. The GPS time is recoded at the onset of data collection by each unit. Then, the results from each unit are copied to a single location in an office computer. The post processing module combines the tracking results from multiple masts and produces one set of results in the TSFF format. Section 5.8 describes the post processing procedure in further detail.

3. TSCAN PROTOTYPE DESIGN EVOLUTION

This chapter talks about the design process the CRS team underwent in developing the two prototypes.

3.1 Mobile Traffic Lab (MTL)

The previous project showed the viability of tracking road users with a LiDAR. Since it was an exploratory work, it used CRS's mobile traffic laboratory (MTL) (see Figure 3.1).

3.2 Move to Multiple LiDAR's in a Single Mast

Due to the extended lead time, the large size and the high cost of the Velodyne HDL 64 LiDAR, a decision was made to redesign the TScan hardware by using modern but smaller sensors. Simulations were made to ascertain the viability of using multiple cheaper sensors instead of the single large bulky sensor.

The simulation results are shown in Figure 3.2. While the coverage for the proposed combination of HDL 32 and VLP 16 (both by Velodyne) is less than that of a single HDL 64E, the density of points arranged along lines (marked in blue) in each polygon was higher than the threshold needed for successful identification of objects. The new sensors configuration was decided with the approval by the sponsor in December 2016.

The development of the prototype hardware was continued to incorporate the new multiple sensor configuration (see Figure 3.3). One of the design requirements was that the new sensor assembly (TScan head) was capable of working with the van-based MTL as well as with the new trailer-based unit. There were two issues to address when developing the first version of the TScan head.

1. The pan motor at the bottom of the unit was not able to bear the entire weight of head unit during the lift process of the mast.

Figure 3.1 Mobile traffic laboratory with TScan hardware.

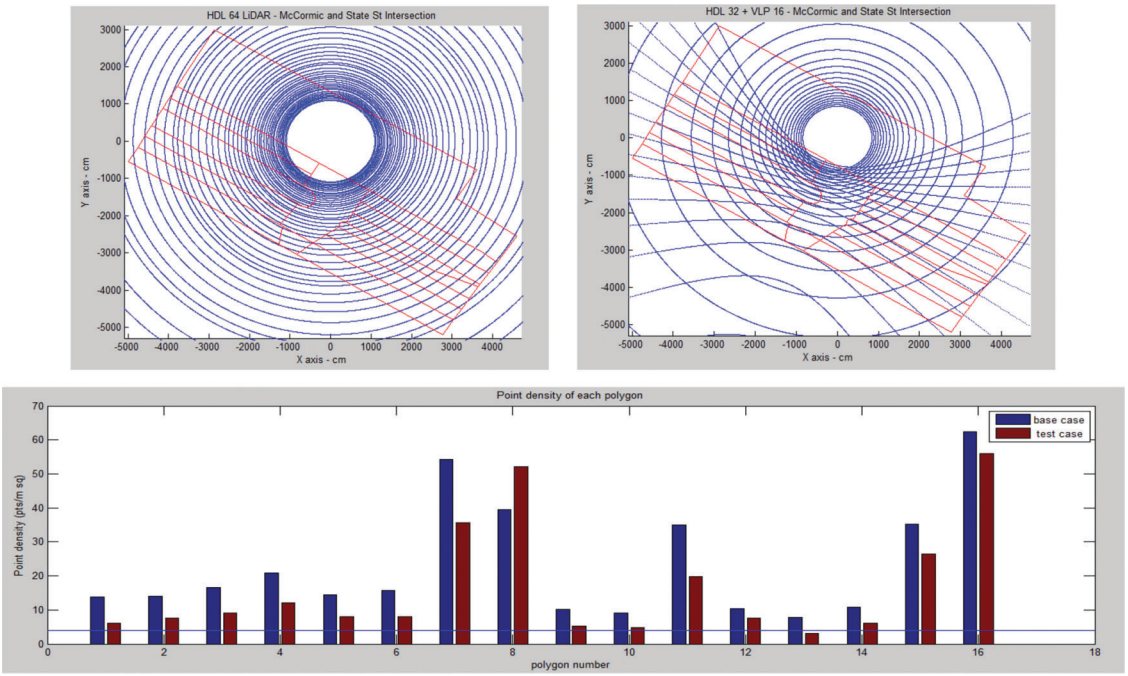


Figure 3.2 Simulation of point cloud distribution.

Figure 3.3 Prototype 1–version 1.

2. The unit was not self-contained. The computer to do all the computation in real time was not part of the head unit and this posed a security issue as multiple days' worth of data collection could be jeopardized if the computer was tampered with.

The first issue was resolved by using a trailer whose mast itself could pan, thereby panning the entire TScan head unit. The second issue was solved by moving the computer and the data storage for the results on the head unit itself (see Figure 3.4). This change provided another benefit that the self-contained unit could be made detachable and thus allow for secure storage of the more expensive components like LiDARs inside a building when not in use.

At the time of purchase, the vendor of Velodyne sensors had a long delivery time (16–32 weeks). It was around the time when a new manufacturer of LiDAR sensors, Ouster, started offering its products. Ouster had a lead time of only 6 weeks and, more importantly, the characteristics of Ouster OS1-64 were similar to those of Velodyne HDL 64-E except for the vertical field of view.

The new Ouster OS1-64 was integrated into the prototype after thorough testing. It required installing

transformer to convert the 12V power supply to 24V required by Ouster sensor. Once integration was completed, additional challenges appeared that the team had to resolve before the prototype was ready for use. The first serious challenge was bringing readings from the multiple LiDAR sensors into a single reference frame (Section 5.2). Using the reported internal IMU angles of the LiDARs was insufficient and an automatic self-alignment method was developed (Section 5.3).

Eventually, the list of hardware components necessary to reach the desired functionality of Prototype 1 was finalized. With this knowledge, Prototype 2 design followed the design of Prototype 1 except the size of the enclosure, which was slightly enlarged to improve an access to the components for rapid testing and troubleshooting not only during the unit's development but also during its future use and maintenance. Figure 3.5 shows a side-by-side comparison of both prototype head units. Another significant change was an addition of another plate to house the LiDAR interface boxes apart from the sliding one that contains the computer.

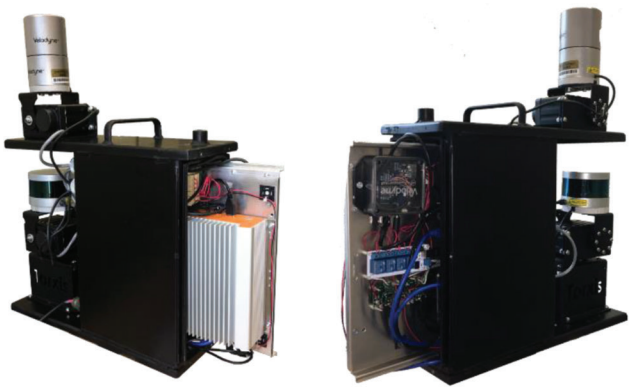


Figure 3.4 Prototype I–version 2.

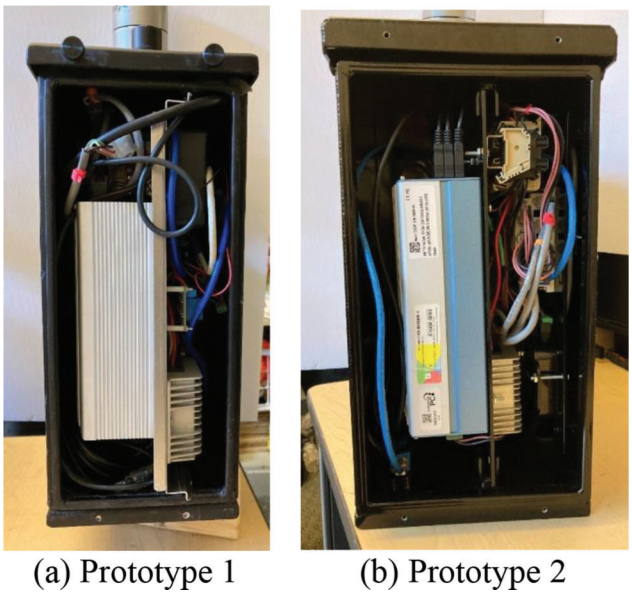


Figure 3.5 Prototype 1 head alongside Prototype 2.

3.3 Final Hardware

The hardware for the TScan Prototypes consists of two primary components: (1) Solar Tech's Trailer with a telescopic mast, and (2) TScan head. The prototype head was designed in such a way that a single person can attach the head to the tip of the mast and then raise it to the desired height.

The TScan head has the following two components: (1) the C-Shape Assembly with the LiDAR sensors attached to the motorized pen-tilt (PT) mechanisms, and (2) the enclosure with the computer and other electronics (see Table 3.1). Two wires, one for power and one of communications must be connected by the user to the TScan head. Detailed descriptions of all the major components used in the prototypes are described in Appendix A. The major hardware components of both prototypes are seen in Figure 3.6.

Velodyne HDL 32E and Ouster OS1-64 are the two LiDAR sensors included in each of the two prototypes. An Inertial Measurement Unit (IMU) is placed at the bottom of the motors on the base of C shape (Figure A.1) in each prototype to compensate for the

TABLE 3.1
Components of TScan Head

C Shape Assembly	Enclosure
Velodyne HDL32E	Fanless Industrial PC
Ouster OS1-64	4 Polulu Jrk 21v3 Servo Controllers
Vivotek Fisheye SF8174V Camera	Velodyne HDL 32 Interface Box
Vectornav VN-100 IMU	Ouster OS1-64 Interface Box
	USB Controlled 4-Channel Switch Relay
	Gigabit Ethernet Switch

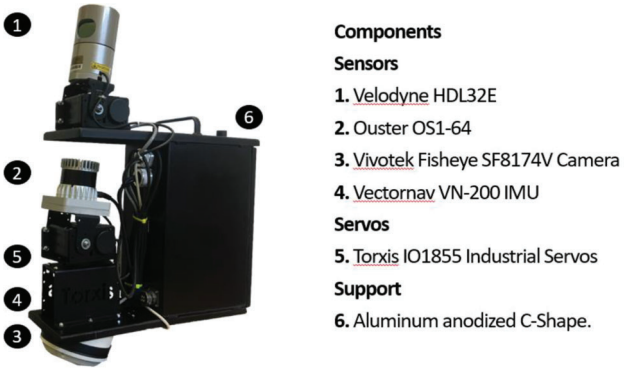


Figure 3.6 Major hardware components of both prototypes.

Figure 3.7 Final version of Prototype 1 (background) along with Prototype 2.

motion of the mast during windy conditions. The SF8174V Fisheye camera by Vivotek is included to provide the opportunity of recording traffic in the field of view of LiDAR sensors during data collection. In the current version of the system, recorded video is used for inspecting traffic events by the end user if needed. Figure 3.7 shows the sensor setup used in both the prototypes.

Servo motors of the PT mechanisms enable precise positioning of LiDAR sensors during field setup of the system. The industrial fan-less computer with installed application is used to set up the system in the field and to perform real-time tracking. A gigabit router is used to handle all the internal communications within the TScan head. Motor controllers to translate the user commands to voltages required by the servo motors round up the major hardware of the system.

The second prototype, built later than the first one, is equipped with a newer computer with better processing capabilities than Prototype 1. Both the computers are more than adequate to perform the necessary computations in real time.

4. TSCAN DATA COLLECTION AND PROCESSING

This chapter describes the TScan process as experienced by the end user of the system. The TScan process involves five phases, some of them with multiple steps, outlined in Table 4.1. The phases include off-site preparation, on-site setup, data collection, data post processing, and user applications. Note that batteries of the trailer have to be sufficiently charged before going to the site for data collection.

TABLE 4.1
TScan Process Overview

1.	Offsite preparation
	1.1 Background image
	1.2 Intersection layout
2.	On-site setup
	2.1 Hardware setting
	2.2 LiDAR and image alignment
	2.3 Sensor coverage
	2.4 Background identification
	2.5 Results inspection
	2.6 Repeat 2.2 to 2.5 if necessary
3.	Data collection
4.	Data post processing
5.	User applications

4.1 Off-Site Preparation

During the off-site setup, the user should obtain an orthogonal image from an external source such as Google Maps (see Figure 4.1). Then, the provided software is used to divide the intersection into several enclosed areas called polygons—entry and exit lanes, intersection area, sidewalks, medians, parking lots, etc. This information is saved and used in the next phase of the process.

Polygons have the following important functions.

- They define the area were the background is expected and identified.
- They help classify moving objects into categories (cars, pedestrian, etc.).
- They used to define traffic movements in the counting application.
- They help identify conflict-free trajectory in the conflict detection application.

For information and guidelines on how to define polygons at an intersection, refer to Appendix Section D.3.5. For information on how to use the TScan off-site setup, refer to Appendix Section D.3.

4.2 On-Site Preparation

Activities in this phase are performed at the site of data collection. For detailed information on this process refer to the Appendix Section D.4.

4.2.1 Hardware Setting

The user hauls the TScan trailer to the intersection selected for data collection. The head unit containing the sensors is transported in a separate protective case. Once at the desired location, the trailer base is properly positioned, stabilized, leveled, and then set up (see Figure 4.2). Stabilization is achieved with the four jacks in the trailer's corners. A spirit level for guidance is present at the center of the trailer base. The head unit is then mounted on the mast and the mast is raised to the desired level.

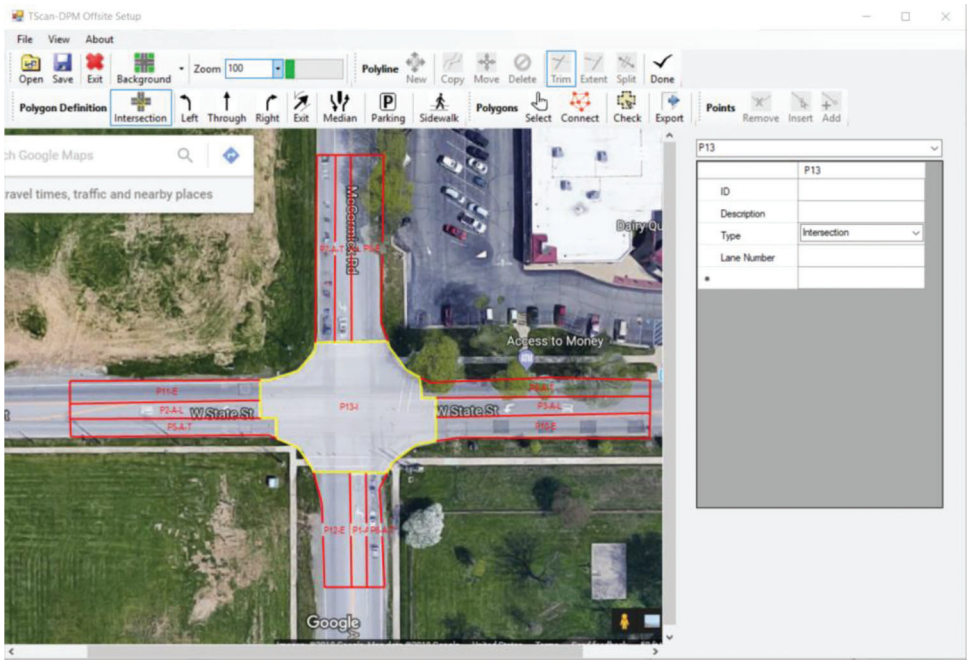


Figure 4.1 TScan off-site setup.

4.2.2 Sensor Coverage

Once the mast is raised, the power to the head is turned on. The user connects to the computer in the head via windows remote desktop. Once the system initializes, the LiDAR points are displayed as shown in Figure 4.3. Using the provided software controls (Figure 4.3) the LiDAR sensors are directed at the intersection by manipulating the various pan/tilt motors. Greater coverage provides better tracking capabilities of the system.

4.2.3 LiDAR and Image Alignment

Once the mast is raised, the power to the head is turned on. The user connects to the computer in the head via windows remote desktop. Using the provided software, the LiDAR points are displayed (Figure 4.4). The ortho image saved during off-site preparation, must be aligned with the LiDAR points. Refer to Appendix Section D.4.6 for details.

4.2.4 Background Identification

Once the system is set up, the user initiates the background identification phase. The system collects sample data for a few minutes to analyze and identify stationary background that includes objects like buildings, road pavement, foliage, etc.

4.2.5 Results Inspection

Once background is identified, the sample data is processed through the real time tracking module and the results are presented to the user through the Trajectory Visualizer application (Appendix Section E.6).

4.2.6 Repeat 4.2.3 to 4.2.5 if Necessary

The tracking results may be found unsatisfactory due to misalignment between the LiDAR points and the ortho-image or due to poor sensor coverage. In such a case, the user has to repeat steps 4.2.3 to 4.2.5. Data collection is finally initiated once the user is satisfied with the sample tracking results.

4.3 Data Collection

The user initiates the data collection phase by entering the amount for data collection. Then the real time tracking algorithm takes over and there is no user involvement needed during this phase.

4.4 Post Processing

The results of real-time tracking of moving objects during the data collection period is stored in a custom TSFF format developed by CRS. The rationale for using TSFF format is explained in Section 2.3.3. The quality of tracking results is further improved in a post processing step that is performed at the office after data collection. Each object is analyzed with respect to itself and to other nearby objects to ensure that a single Object ID represents one physical object from the moment the object enters the intersection till the moment it leaves the LiDAR's field of view. Classification of road users takes place during this phase. Refer to Section 5.8 for more details on this phase.

Figure 4.2 TScan trailer on-site setting.

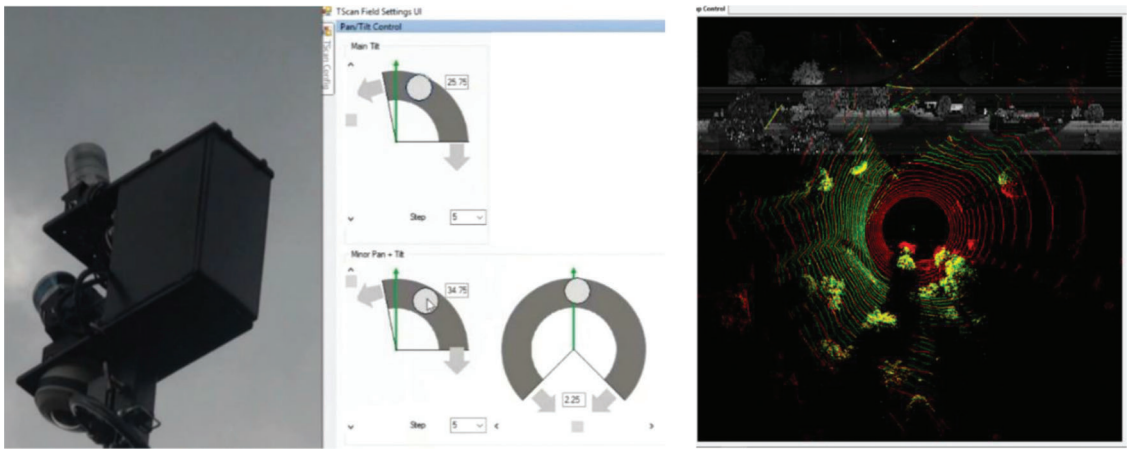
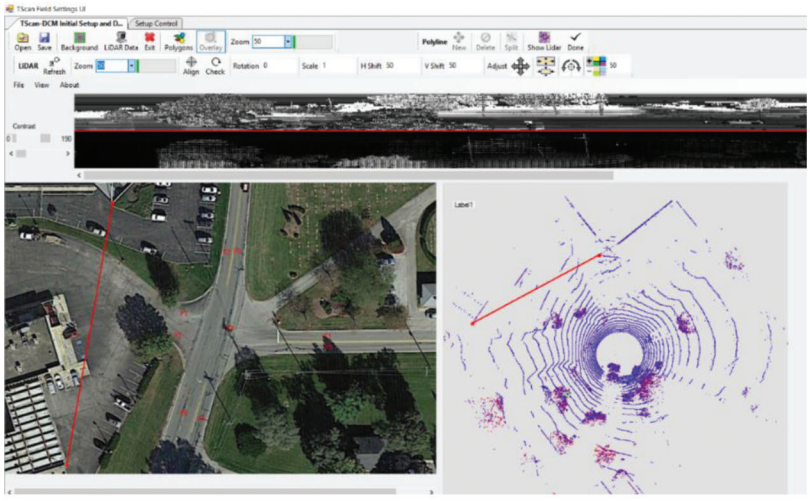


Figure 4.3 Sensor coverage.

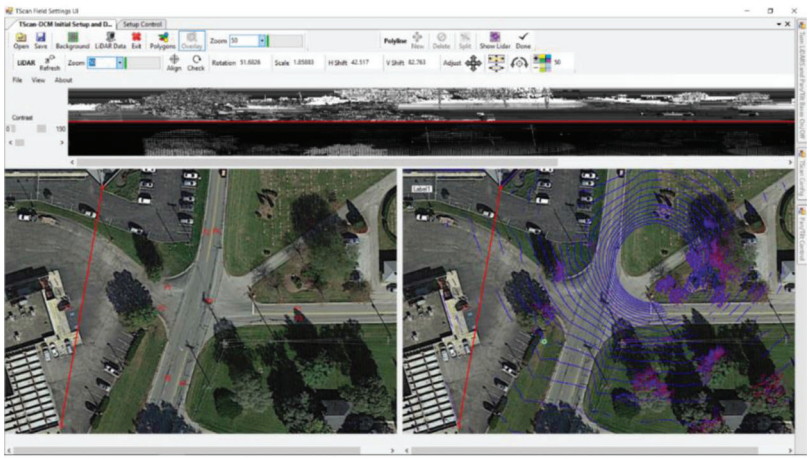
In addition to improving tracking results, post-processing step also does preliminary computations for engineering applications such as pedestrian occupancy and traffic conflicts. For the pedestrian occupancy application, counts are performed at 1-minute intervals and all other time periods are aggregation of these 1-minute counts. Instantaneous time to collision or iTTC is also computed during the post processing.

4.5 TSFF File Format

The SSAM file format was inadequate to carry all the necessary information to support the various applications that TScan enables. For instance, SSAM format does not have a field to indicate the class of road user the trajectory belongs to. It assumes all the trajectories are vehicle trajectories. Thus, a new file format



(a) Selecting the points for alignment



(b) After alignment

Figure 4.4 Aligning LiDAR points with ortho-image.

was developed, namely TScan File Format (TSFF). The results of tracking and post processing are stored in this format. Refer to Appendix C for detailed description of the new file format.

The results of post processing are also in TSFF format, though in a separate folder. These results are used by all the applications developed by CRS—traffic counts, speed analysis, pedestrian occupancy, and traffic conflicts. Chapter 7 describes these applications in further detail.

5. REVISED ALGORITHM

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 an 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.

The overall progression of data is shown in the flowchart in Figure 5.1. Although the overall steps involved in the processing remain the same as presented in the previous report (SPR-3831), each step has been improved either through refinement or by replacing it with a better performing algorithm. The improved elements are highlighted in Figure 5.1. The process has been modified to support the TScan unit's multiple LiDARs.

Background identification and elimination is applied independently on each LiDAR. During the clustering phase, the data points from multiple LiDARs are combined before detecting objects. Kalman filter-based algorithm is used to track objects across multiple frames. Once the tracking of an object is finished, the object's entire history is used to estimate its dimension.

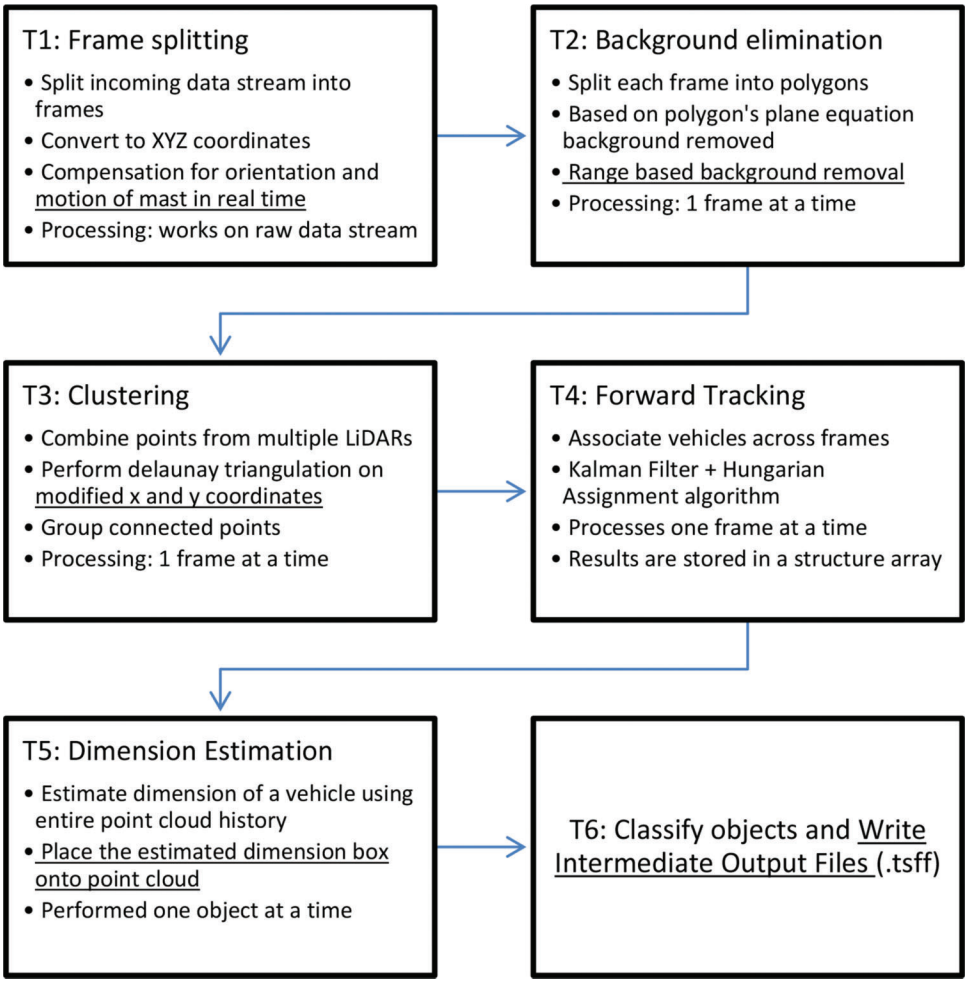


Figure 5.1 Schema for real time processing after background identification.

The results are then stored in a custom TSFF format (Appendix C).

5.1 Setting Up of the System

As described in the previous sections of this report, few preparation steps must be performed before the actual data collection begins. Part of this preparation is performed in the user's office (off-site–Section 4.1) and the remainder at the intersection (onsite–Section 4.2).

During the off-site 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 that the user is interested in tracking road users (Appendix Section D.3.5). 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 ortho image data and selecting the position and orientation of the LiDAR system.

5.2 Alignment of Image and LiDAR Data Points

The TScan spatial referencing uses four coordinate systems. The first major task in the processing pipeline is to bring all these coordinate systems into a single one. The following are different coordinate systems.

1. Geographic coordinate system (cm on the ground).
2. Image coordinate systems (number of pixels from top left corner).
3. Mast coordinate system (cm, origin at the base of the TScan head).
4. Individual LiDAR coordinate system (cm, origin within the sensor).

Both Velodyne HDL 32 and Ouster OS1 64 LiDAR's have their sensor coordinate frame's origin within their sensor body. The respective manuals provide details on the exact position of their origin. Both the sensors also

have in-built IMUs. This allows for easy estimation of their orientation.

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:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \quad \text{(Eq. 5.1)}$$

Similarly for the y and z axis,

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad \text{(Eq. 5.2)}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{(Eq. 5.3)}$$

If u represents a fixed reference frame and v represents a frame rotated by α, β, and γ about the x, y, and z-axis of the fixed frame respectively, then the transformation can be represented by Equation 5.5.

$$R_r = R_z(\gamma) \times R_y(\beta) \times R_x(\alpha) \quad \text{(Eq. 5.4)}$$

$$v = u \times R_r \quad \text{(Eq. 5.5)}$$

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

$$u = v \times {R_r}^T \quad \text{(Eq. 5.6)}$$

The above set of equations assume that there is no translation involved i.e., all the axes about which the rotations occur intersect each other. In reality the sensors are mounted on top of motors. The motors' axis may or may not be aligned such that their axes of rotation intersect. Moreover, the sensors are typically placed on a platform attached to the axis of rotation. This platform is a fixed distance from the axis and this has to be considered. Therefore, a 4×4 homogeneous transformation matrix is used that incorporates rotation and translation.

Thus, given a vector u, its transformation v is represented by

$$H_T = \begin{bmatrix} R & T^T \\ 0 & 1 \end{bmatrix} \quad \text{(Eq. 5.7)}$$

$$v = H \times u \quad \text{(Eq. 5.8)}$$

Where,

H is the homogeneous transformation matrix;
R is the rotation component which can by any combination of R_x, R_y and R_z; and
$T=[T_x, T_y, T_z]$ represents the translation components along x, y, and z axes, respectively.

The mast's origin is taken to be at the front end of the bottom plate of the C shape. The X axis is aligned with the body of the head with the positive direction being towards the box.

A series of rotations and translations are made such that the points are transformed from the sensor coordinate frame within the body of the sensor to the mast origin (Figure 5.2). The governing equation is given as:

$$\begin{aligned} C = {} & R_z(MountAngle) \times R_x(rollAngle) \\ & \times T(mountTrans) \times T(tiltTrans) \\ & \times R_y(tiltAngle) \times T(translBetwAxes) \\ & \times R_z(panAngle) \\ & \times T(panTransl) \times T(servoPos) \end{aligned} \quad \text{(Eq.5.9)}$$

Note that in the above equation each transformation shown is used in its 4×4 homogeneous form.

This process is applied individually to each LiDAR sensor present in the mast, bringing them all to same reference frame.

Next set of transformations is to move from mast coordinate system to geographic coordinate system. The mast coordinate system is rotated such that the +Y axis aligns with cardinal north and translated such that center of the image is the origin.

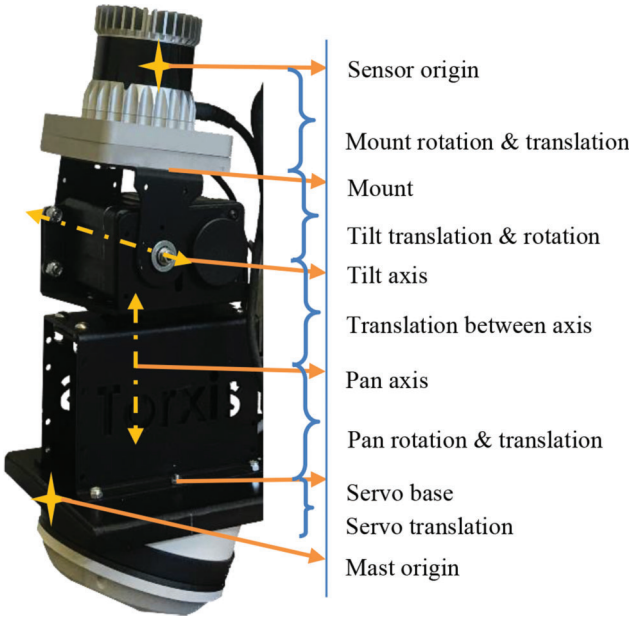


Figure 5.2 Sensor alignment.

For visualization, the XYZ coordinates have to be displayed on top of an ortho image. For this an additional scale factor is needed. Note that the origin of the image coordinate system is the first pixel, or the top left corner of an image. The governing equation to move from XYZ coordinate to image coordinates is given as:

$$x_p = \frac{1}{2}w + x * s$$
$$y_p = \frac{1}{2}w - y * s \quad \text{(Eq. 5.10)}$$

Where,

x_p and y_p are pixel coordinates along x axis and y axis, respectively;
x and y are geographic coordinates along x axis and y axis, respectively; and
s is the scale factor to convert from pixels to cm.

If ortho image is obtained from Google Maps, the extraction process gives us latitude, longitude, distance spanned in meters on the ground along the height of the image. Google Map images have the same scale both along the height and width of the image.

If the image is provided by the user and the above information is typically missing, then the scale factor is determined based on the points selected by the user for alignment.

The image captured should contain the regions spanned by all the masts used for that site. Image should have some distinctive features to help with alignment such as trees, buildings, electric poles.

5.3 Multiple LiDAR Self-Alignment

Both of the TScan prototypes have two LiDARs working together to provide enough coverage of the intersection. Each LiDAR has its own internal coordinate system in which it makes measurements. When using multiple LiDARs it is imperative that all the points generated are brought to one common coordinate system.

As a first step, the angles reported by the servo motors were used to align both the LiDARs. The servo controller's estimate of the angles was not precise enough to accurately calculate the orientation of each LiDAR. Next, the IMU readings form the two LiDARs themselves were used to align the points. Upon testing, even this method did not produce accurate enough results.

The two LiDARs are usually set up in such a way that there is a considerable amount of overlap between their coverage of the intersection. The self-alignment method looks for these common areas that are flat in nature (for instance road pavement) where points from both LiDARs are available. Two planes are then fit individually for each LiDAR's points. Then, the transformation matrix to align the two planes is computed. This transformation matrix is used to bring both sets of points from the two LiDARs on to a single reference frame.

An illustration of the performance of all the three methods discussed is shown in Figure 5.3. The first set of comparisons shows the raw LiDAR points after compensation. The second set of figures show the background plane identification results.

5.4 Conversion of Coordinates

The first step in the processing pipe line is to convert the incoming raw data into 3D points in cartesian coordinates. Each LiDAR sensor has its own internal frame of reference according to which the measurements are made. Data from both the LiDARs used in the prototype is comprised of the following pieces of information for each laser measurement.

- The angle about z-axis of the sensor at which it is oriented.
- The distance reading recorded by a laser.
- The intensity of return of the laser.
- The timestamp when the laser was fired.

Some pieces of the above may not be present for every laser fired but can be calculated based on the sensors' model and the information given. For instance, it is common to provide only one angle value for a set of laser readings. In such a case, it is implied that that angle is to be used for every reading in the set.

The sensor model is used to convert the reported data into usable XYZ coordinate points. The sensor model can be imagined as follows. There are a certain number of individually aimed laser diodes that all lie in a single vertical plane and the rays emanating from them intersect at the origin of the sensor coordinate frame. The following parameters (shown in Table 5.1) are defined by the manufacturer for each laser scan line to model the deviation from said ideal conditions.

Though the intended meaning of these parameters is same for both sensors, the sign conventions vary based on the sensor used. The values of 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.

5.4.1 Velodyne HDL-32

The Velodyne HDL 32 as the name implies has 32 laser scan lines. More importantly the FOV is such that it can cover areas closer to the base of the mast than Ouster OS1-64. The computation of the local scanner coordinates (x, y, z) for laser i of the Velodyne scanner is given by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (R_i) \times cos(\delta_i) \times sin(\varepsilon) \\ (R_i) \times cos(\delta_i) \times cos(\varepsilon) \\ (R_i) \times sin(\delta_i) \end{bmatrix} \quad \text{(Eq. 5.11)}$$

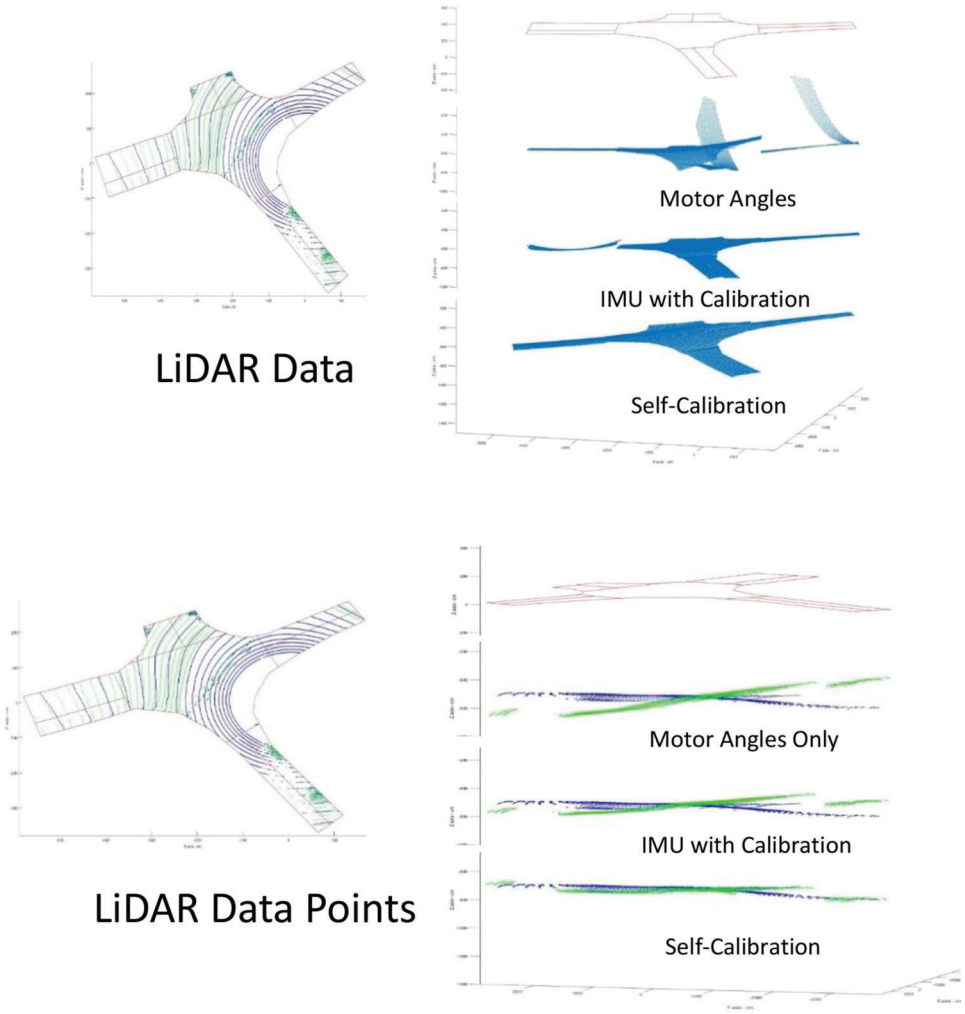


Figure 5.3 Self calibration for aligning the two LiDARs.

TABLE 5.1
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 or vice versa.
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 or vice versa.

Where,

R_i is the raw distance measurement from laser i;
ε is the encoder angle measurement; and
δ_i is the vertical correction parameters pertaining to laser i as explained in Table 3.1.

As shown in Figure 5.4, the origin of the sensor is not at the base of the sensor but 3.575 inches or 9.0805 cm above it. This is taken as mount translation when applying the transformation described in Section 5.2.

5.4.1.1 Data Structure of Communication Packets. The HDL-32E 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 is output in the form of UDP packets. Each packet contains a data payload of 1,206 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 (see Figure 5.5).

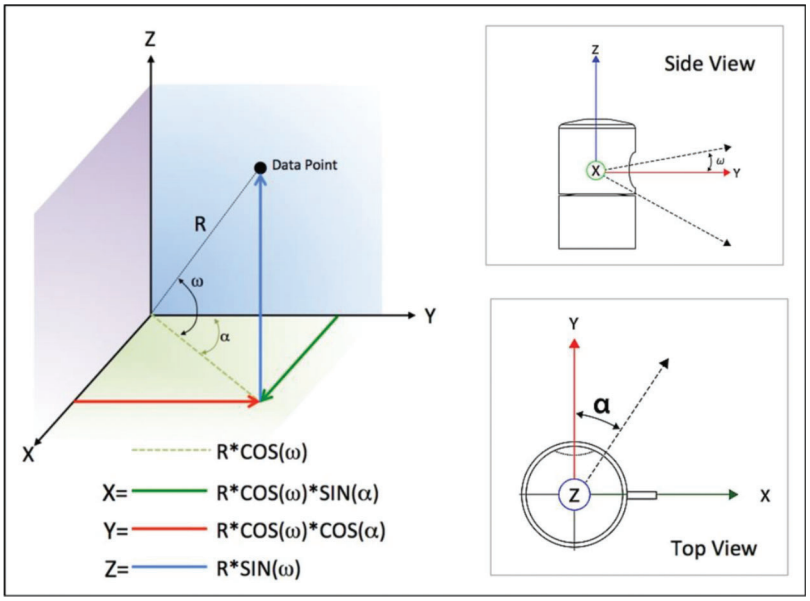


Figure 5.4 Velodyne HDL-32 sensor coordinate system.

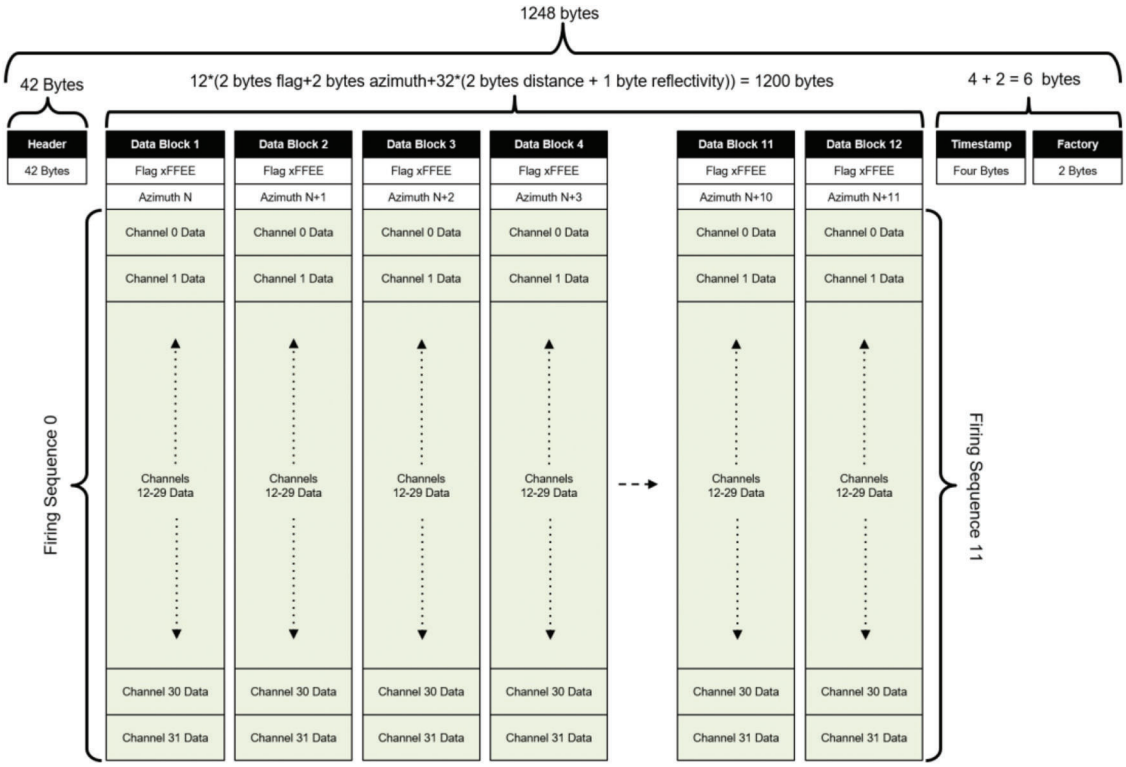


Figure 5.5 Velodyne HDL-32 packet structure.

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 $0 \times$ EEFF and the lower block has a header of $0 \times$ DDFF (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.). For HDL-32 it only $0 \times$ EEFF applies.
- *2 bytes of rotational information*
  - This is an integer between 0 and 35,999. Dividing this number by 100 produces values in degrees. Only one azimuth value is given per data block.

- *32 laser return info of 3 bytes each*
  - Each return contains *2 bytes* of distance information, in 0.2-centimeter increments, and *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.
- *4 bytes of time stamp information*
  - The four-byte time stamp is a 32-bit unsigned integer marking the moment of the first data point in the first firing sequence of the first data block. The time stamp's value is the number of microseconds elapsed since the top of the hour. The number ranges from 0 to 3,599,999,999 the number of microseconds in 1 hour.
- *2 bytes of rotational information*
  - The first byte is called return mode byte. It takes one of the following values depending on the mode selected by the user. By default, they are configured for strongest return mode or value 0×37. The other modes are Last return (value 0×38) and dual return (0×39).
  - The next byte indicated a product ID which is used to identify how the lasers are arrayed vertically. For a full explanation refer to the sensor user manual (Velodyne, 2021).

5.4.2 Ouster OS1 64

The ouster sensor is placed on top of a pan tilt base below the Velodyne LiDAR as shown in Figure 3.6. The computation of the local scanner coordinates (x, y, z) for laser *i* of the Ouster scanner is given by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (R_i) \times cos(\delta_i) \times sin(\varepsilon) + cos(\varepsilon + \beta_i) \\ -(R_i) \times cos(\delta_i) \times cos(\varepsilon) + sin(\varepsilon + \beta_i) \\ (R_i) \times sin(\delta_i) \end{bmatrix} \quad \text{(Eq. 5.12)}$$

Where,

R_i is the raw distance measurement from laser i;
ε is the encoder angle measurement; and
δ_i, β_i are the parameters pertaining to laser i as explained in Table 3.1.

Similar to Velodyne HDL 32, the ouster sensor has its origin located 36.180 mm from its base. This has to be taken into account when performing the calculations explained in Section 5.2.

5.4.2.1 Data Structure of Communication Packets. Ouster OS1-64 communicates via a LAN by sending UDP packets. Each packet contains a data payload of 12,608 bytes. Each packets contains 16 azimuth blocks and are always 12,608 bytes in length. The packet rate is dependent on the output mode. Words are 32 bits in length and little endian. See Figure 5.6

Each azimuth block contains the following.

- *Timestamp [64 bit unsigned int]:* timestamp of the measurement in nanoseconds.
- *Measurement ID [16 bit unsigned int]:* A sequentially incrementing azimuth measurement counting up from 0 to 511, or 0 to 1,023, or 0 to 2,047 depending on LiDAR mode.
- *Frame ID [16 bit unsigned int]:* Index of the LiDAR scan. Increments every time the sensor completes a rotation, crossing the zero point of the encoder
- *Encoder Count [32 bit unsigned int]:* An azimuth angle as a raw encoder count, starting from 0 with a max value of 90,111—incrementing 44 ticks every azimuth angle in 2,048 mode, 88 ticks in 1,024 mode, and 176 ticks in 512 mode.
- *Data Block [96 bits]:* 3 data words for each of the 16 or 64 pixels. Refer to Table 5.2 for full definition.

Word	Azimuth Block 0	Azimuth Block 1	...	Azimuth Block 15
(Word 0,1)	Timestamp	Timestamp	...	Timestamp
(Word 2[0:15])	Measurement ID	Measurement ID	...	Measurement ID
(Word 2[16:31])	Frame ID	Frame ID	...	Frame ID
(Word 3)	Encoder Count	Encoder Count	...	Encoder Count
(Word 4,5,6)	Channel 0 Data Block	Channel 0 Data Block	...	Channel 0 Data Block
(Word 7,8,9)	Channel 1 Data Block	Channel 1 Data Block	...	Channel 1 Data Block
	.	.		.
(Word 193, 194, 195)	Channel 63 Data Block	Channel 63 Data Block	...	Channel 63 Data Block
(Word 196)	Azimuth Data Block Status	Azimuth Data Block Status	...	Azimuth Data Block Status

Figure 5.6 Ouster OS1-64 packet structure.

TABLE 5.2
Ouster OS1 64 Data Block Format

Word	Byte 3	Byte 2	Byte 1	Byte 0
(Word 0)	unused[31:24]	unused[23:20] range_mm[19:16]	range_mm[15:8]	range_mm[7:0]
(Word 1)	signal_photons[31:24]	signal_photons[23:16]	reflectivity[15:8]	reflectivity[7:0]
(Word 2)	unused[31:24]	unused[23:16]	noise_photons[15:8]	noise_photons[7:0]

- *Azimuth Data Block [96 bits]:* Indicates whether the azimuth block contains valid data in its channels' Data Blocks. Good = $0 \times$ FFFFFFFF, Bad = 0×0. If the Azimuth Data Block Status is bad (e.g., in the case of column data being dropped), words in the data block will be set to 0×0, but Timestamp, Measurement ID, Frame ID, and Encoder Count will remain valid.

5.5 Background Identification and Removal

Background identification happens during the setup process and background removal happens during real time data collection. Once the system is set up, sample data are collected for 15 minutes to identify the background. Previously two different approaches, based on type of polygon, were used to identify background. For polygons of a road pavement section surface equations were used to approximate the background for other polygons such as medians and sidewalks, a spherical coordinate-based thresholding was applied (Tarko et al., 2016).

On extensive testing, the following drawbacks were observed about the background identification process.

- The choice of background identification method depended on polygon type as designated by the user.
- Thus, if the polygons were not properly aligned, fixed objects like walls, barriers would be present in road pavement polygons instead of sidewalks or medians. This would lead to improper background identification and hence removal.

To overcome the mentioned drawbacks, two new complementary methods were developed that would be applied to all the polygons in the intersection.

5.5.1 Surface Tile

While surface equations provided an accurate approximation for road pavement with simple geometries, it failed when dealing with more complex geometries. In a four-legged intersection such as the intersection of Morehouse Road and W 350 N in West Lafayette, where the main Morehouse Road is on a vertical curve and W 350 N Road joins at a steep angle the surface equation-based method failed to produce accurate representations of the surface.

To get a more generalized surface approximation of the road surface, the following algorithm is adopted.

Algorithm Overview

- Convert readings reported by a LiDAR into cartesian coordinates by performing the necessary transformations. Any compensation for orientation of the sensor and motion of the mast is applied.
- 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 not parallel to the gravity vector. 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 polygon, use RANSAC algorithm to fit a plane. A threshold of 20 cm is used to determine outliers which are then removed. Repeat the fitting process until there are little to no points (a 3% threshold value is used) to remove.
- Separate the remaining points based on their x and y coordinate values into tiles of 25 cm $\times$ 25 cm.
- The average "z" value of each "tile" with a significant number of points is computed.
- Perform Delaunay triangulation on tile averages.
- Plane equation of each triangle in the triangulation is computed.
- For tiles with insignificant number of points, compute the "z" value from the plane equation of the triangle it belongs to.

5.5.2 Percentile-Based Background

The concept of distance-based separation of background and moving objects in spherical coordinates remains the same as stated in the previous report. Instead of using averages and standard deviation of readings reported by a laser n at angle α, the threshold is determined according to the following algorithm.

Algorithm Overview

- Collect data for 3,000 frames or more when traffic is low.
- Group distance readings by laser and angle.
- Separate the distance readings of each group into bins of desired size (5 cm bins are used).
- Identify the farthest set of three consecutive bins that combined has more than 10% of the distance readings for that group.
- The bin edge with the smallest distance reading is the desired threshold. Any reading less than this threshold is considered foreground.

5.6 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 the following 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.

Algorithm Overview

This phase consists of the following steps.

- All the points in the frame post background elimination is converted from cartesian to a spherical coordinate system.
- Delaunay triangulation (Delaunay, 1934) is performed for all the points in the frame post background elimination.
- The lengths of all the connections obtained from triangulation are computed.
- All connections that are greater than NEIGH_RADIUS is 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 of SPR-3831 report).
- 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 clouds of points are combined and a new bounding box is computed.

This process similar to the previous to the one previously reported. The change is that the triangulation is performed on transformed points (x^*, y^*) rather than only the x and y coordinates. The transformed system takes into account how the points are arranged in 3D.

5.7 Forward Tracking

The next step is to associate clusters belonging to the same physical object across frames. For this a Kalman Filter (Kalman, 1960) based multi object tracking system is used. A constant acceleration model is used to predict the motion of objects. Section 3.5 of SPR-3831 describes the Kalman Filter and the governing equations of motion used in detail.

5.8 Dimension Estimation and Box Placement

After associating clusters across frames and identifying objects, the next step is to estimate dimensions of the object. A critical element of estimating the dimensions is the angles reported by minErrorRect procedure (Section 3.4.3 in SPR-3831 report). But these angles are insufficient to place the estimated box back on the 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 (MBF) smoother (Section 3.6.1 in SPR-3831 report).

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 sensor has inherent and somewhat troublesome behavior of not hitting the same physical point in space after each rotation. This results in a non-stationary centroid of a stationary object.

A lower bound of speed is proposed to account for this non-stationarity. An object is assumed stationary if its measured velocity is lower than the assumed lower bound. This method relies on a single threshold and it works satisfactorily for vehicles sufficiently close to the LiDAR but it fails in the following two cases.

1. For stationary objects far from the sensor, the false motion induced by the LiDAR sensor is so large that the measured velocity is sometimes greater than the assumed threshold. It is the case where the signal to noise ratio is small.
2. During occlusion in progress, the visible portion of an occluded object increases or decreases. It causes the speed of the occluded object underestimated or overestimated. Consequently, a stationary object may appear as moving and moving object as nearly stopped (Figure 5.7). In this instance, the noise is low but the signal is false.

To better identify the portion of the trajectories where the vehicle is stopped regardless of occlusion, the input to MBF smoother is itself smoothed. A robust local regression using weighted least squares and a 2nd degree polynomial is used to smooth the centroid after tracking before applying MBF smoother. Using this two-pronged approach, regions where the object is stationary is better identified (Figure 5.7) which leads to better box placement in those regions. The estimation of orientation of the object has been improved considerably when it is stationary.

5.9 Post Processing Algorithm

At completion of data collection in the field, the user is expected to retrieve the storage devices from each of the trailer units and assimilate the data in to a single folder in an office PC. Then, the post processing part of the program can be launched.

5.9.1 Combining Data from Multiple Masts

The first step in the post processing phase is combining data from all the trailer units used in the data collection. The results of the field data collection from each of the trailer units are synchronized in time using a GPS and they are in the same spatial reference frame thanks to the reconciliation (data transformation) process outlined in Section 5.2.

The process of combining points is as follows. For each frame of data (single full rotation of a LiDAR sensor), raw points that make up the clusters (objects) in the TSFF files are loaded from all the TScan units that has data for that frame. Note that each TScan unit starts collecting data at different points in time as the user has to individually start the real time data collection one mast at a time. Thus, especially for the initial period before all the units start collecting data, only some of the units have data for that frame or time

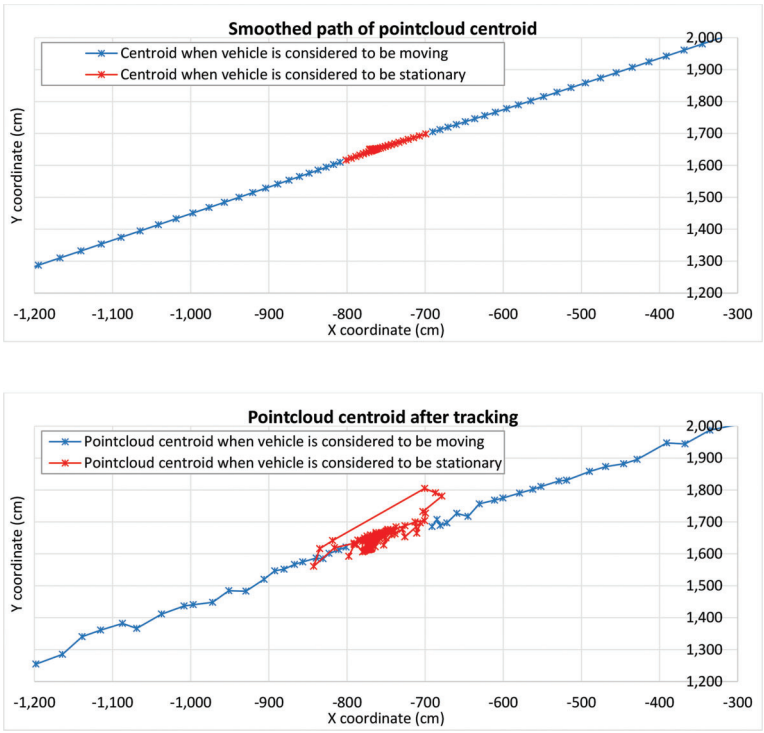


Figure 5.7 Sample trajectory of a vehicle during occlusion.

instant. Once the points are loaded, clustering (Section 5.6) is performed on this combined set of points, followed by tracking (Section 5.7), dimension estimation and box placement (Section 5.8) in that order. Thus, at the end of this process, the data from all the trailer units are combined into a single set of clusters.

5.9.2 Refining Cluster IDs

In order a series of algorithms are performed to ensure that each cluster ID at the end of tracking represents one physical object. These are outlined in Section 3.7 of SPR-3831 report, a brief description is provided below for completeness.

Shadows (areas behind objects not visible from the sensor's point of view) may be present in the field of view of a given trailer unit due to stationary objects like poles, trees, etc., or dynamic objects such as large vehicles, for example, a truck casts its shadow on a passenger car. A vehicle passing through the shadow is split into two clusters due to the nature of the clustering algorithm. These two clusters are given separate cluster IDs. Both the clusters must be combined into one cluster as they represent a single object.

In some cases of split clusters, one cluster carries the correct ID while the other cluster may have its ID incorrectly associated with a nearby vehicle. In order to account for the possibility of such events, the trajectories of each of the cluster IDs must be analyzed and the incorrect associations have to be corrected by splitting the incorrect tracks into parts that represent separate vehicles and combining the correct ones.

When the cluster IDs in the results correctly represent actual objects (one-to-one match) then further analysis including dimension estimation is much more reliable and accurate. Although multiple LiDAR units vastly reduce the occlusion (shadowing) problem thanks to multiple viewpoints, the problem may still persist. Thus, the iterative process of splitting and merging cluster trajectories is still performed (Section 4.8 of SPR-3831 report).

5.9.3 Recomputing Dimensions and Final Smoothing

After trajectories are split and some rejoined as explained in the previous, it is assumed 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. Dimensions of objects whose trajectories have been modified are estimated once again. The boxes with improved dimensions are placed back on top of the point clusters using the algorithm explained in Section 5.8. The centroids of these boxes are then smoothed using local linear second order regression. The positions of boxes are adjusted accordingly and the final trajectory estimate of the object is obtained. The results are saved in TSFF format.

5.9.4 Calculations for Other Engineering Applications

Certain additional calculations performed in the post processing module may considerably reduce the execution time of some engineering applications. The calculations performed at this stage include the following.

- Pedestrian occupancy is calculated for 1-minute aggregation interval to be used in the pedestrian occupancy engineering application (Appendix E.9).
- Instantaneous time to collision or iTTC is calculated for every object pair in the data set and information regarding those pairs whose iTTC values are below a certain pre-defined liberal threshold (currently at 5 seconds) are saved. This information is used by the Traffic Conflicts application (Appendix E.11).

6. EVALUATION

The following chapter describes the procedure applied to evaluate TScan's performance in detecting traffic conflicts after improving the algorithm. The baseline for comparison is the set of algorithms reported in the *2016 TScan Report*.

6.1 Data

To have a reliable comparison, the same data sets were used to evaluate the performance of the improved algorithm described in this report and the old algorithm reported in the previous report. By using the same data, the effect of the new sensor arrangement is not taken into consideration and the sole effect of the algorithmic improvements on the computational performance is estimated. The data used in the evaluation were collected at intersections presented 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 was 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 (City of West Lafayette, 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

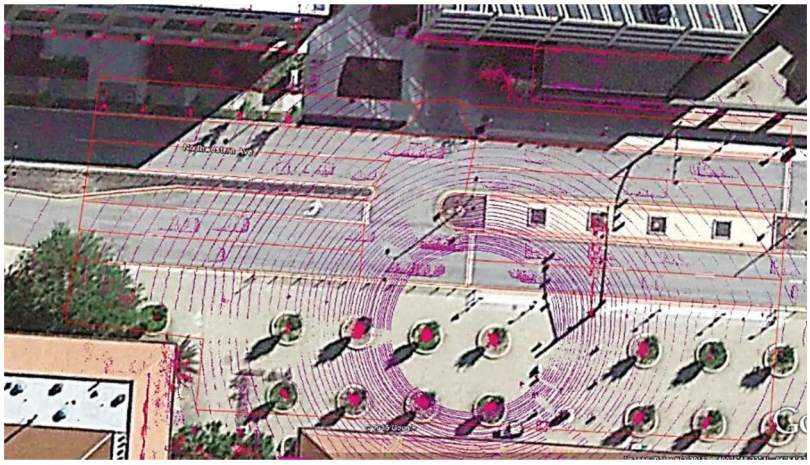


Figure 6.1 Pedestrian crossing at 504 Northwestern Avenue.

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

6.2 Result Comparison

As stated in the previous chapters, the improvements done to the TScan algorithm primarily targeted at better identification of conflicts and eliminating as much false positives as possible. A comparison between these revised algorithms and the ones previously reported in SPR-3831 report is presented in Table 6.1. Both sets of results were processed by SSAM included 60 minutes of Northwestern traffic, 30 minutes of McCormick traffic, and 25 minutes of Morehouse traffic. Only interactions that involved at least one vehicle are reported.

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

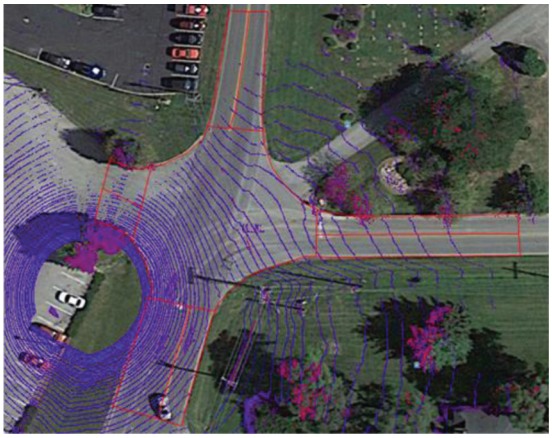


Figure 6.3 Intersection at Morehouse Road and West 350 North, Indiana.

For the pedestrian crossing at 504 Northwestern Avenue, SSAM previously reported 13 interactions that involved at least one vehicle during the analyzed 1 hour. By contrast the current set of algorithms only report 3 conflicts. Issues regarding improper clustering and pedestrian with overestimated dimensions have been eliminated by the improvements made to the algorithms. Box placement especially when occlusion occurs is still the primary reason for the remaining conflicts. The conflicts reported at the intersection of McCormick and State Street can also be attributed to box placement.

The true conflict reported in Morehouse Road intersection was not reported by SSAM in the current version. Upon further investigation, it was observed that due to better box placement and dimension estimation, the iTTC was over a 1.5-second threshold. Visual inspection only revealed that the two vehicles indeed got close to each other but not necessarily the exact iTTC.

7. TSCAN TOOLBOX–ENGINEERING APPLICATIONS

In order to facilitate the TScan use in INDOT traffic and safety analysis, several engineering applications were developed for the visualization of TScan data, counting moving objects, evaluating speeds inside intersections, detecting traffic conflicts, and measuring pedestrian presence on roadways. A program called *TScan Toolbox* combines these applications in one software tool. Figure 7.1 shows the TScan Toolbox with the engineering applications. Detail information about the engineering applications can be found in Appendix E.

7.1 Trajectory Visualizer

The Trajectory Visualizer is meant to help graphically inspect the TScan tracking results and the motion details of selected vehicles. Moving rectangles that represent objects (vehicles, bicycles, or pedestrians) are displayed together with the intersection layout.

TABLE 6.1
Traffic Interactions at the Studied Intersections–SSAM Extracted

	Northwestern		McCormick		Morehouse	
Case	**Collisions**	**Conflicts**	**Collisions**	**Conflicts**	**Collisions**	**Conflicts**
Results from 2016 Report						
True positives	0	0	0	0	0	1
False positives	10	3	5	1	1	0
Results from Revised Algorithms						
True positives	0	0	0	0	0	0
False positives	1	2	2	0	0	0

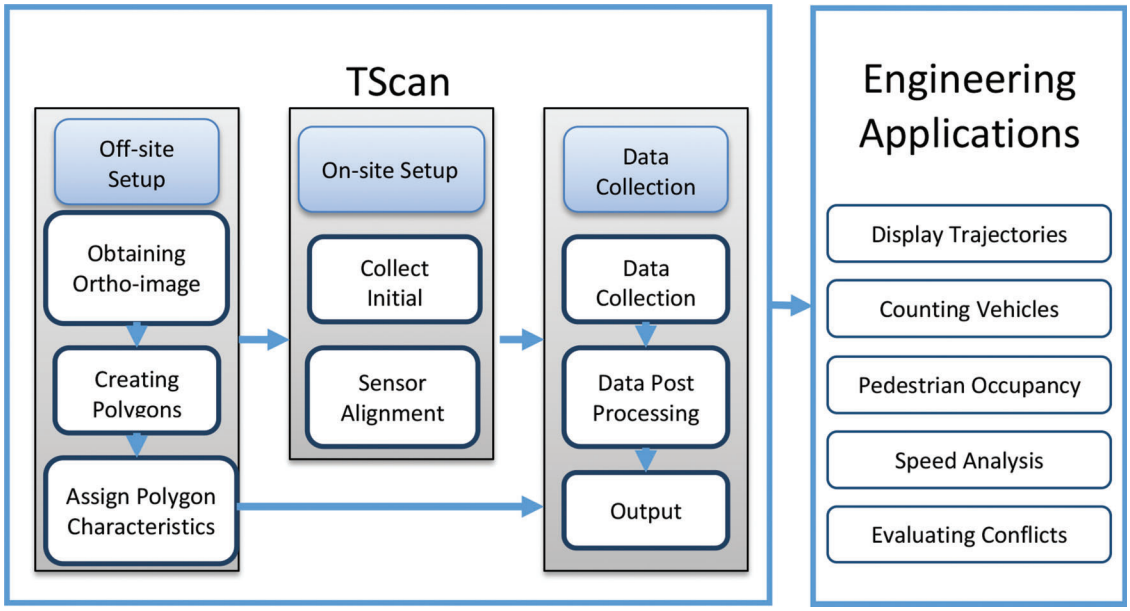


Figure 7.1 Self calibration for aligning the two LiDAR units.

The visualization can be progressed frame by frame or as a continuous video controlled with the play, pause, and stop buttons. A particular frame can be selected either by using the scroll bar at the bottom or by typing the desired frame number. Also, the zoom level of the image can be adjusted by selecting it in the dropdown list, as well as the distance, speed, and acceleration units.

Finally, a particular vehicle can be selected for a more detailed analysis by clicking on the vehicle's box or by typing in the vehicle ID. Once a vehicle is selected, its location, speed and acceleration in the current frame is displayed as well as a chart that represents its entire trajectory. The representation of the objects can be changed to show the LiDAR points that compose it, its bounding box, and its path by selecting the corresponding checkboxes.

7.2 Counting Vehicles

The Counting Vehicles module, named "counts" allows the user to set up the directional counting based on the TScan output files. And to obtain statistics about the directional counting using user-defined counting periods. It includes creating the Origin-Destination matrix based on the created reference polygons of the intersection and the ability to export the results to a csv file. A set of tools to assign polygons (Appendix Section D.3) to the O_D matrix, adjust counting interval and to count vehicles was created. The Output can be saved as a comma separated values .csv file.

7.3 Speed Analysis

The speed analysis module can be used to calculate the user-selected percentiles and the average speed of vehicles during their passage through the intersection zone for each traffic movements defined in the origin-destination matrix.

The end user may set a spot speed threshold to detect and filter out vehicles that at some point during their passage through the intersection slow down considerably or stop to yield to the opposing traffic or pedestrians. This filter helps estimate the average speed and speed percentiles of vehicle motion not disturbed by other road users. Such speed information is crucial for designing traffic signals correctly.

In addition, the desired speed percentile can be selected and the results can be saved as a comma separated values .csv file.

7.4 Pedestrian Presence

The pedestrian presence in a small area during a certain period can be measured with the occupancy rate calculated as percent or portion of time when a pedestrian is present in the area. The field of view is divided with a grid into small squares. The occupancy rate of each of the squares is graphically presented with color intensity, the more intense color, the higher occupancy. The time interval for calculating the occupancy is selected from the pulldown list, which includes 1-, 5-, 15-, 30- and 60-minute options. Color intensity levels assigned to various occupancy ranges are controlled by the end user.

7.5 Traffic Conflicts

TScan post-processing generates a list of interactions between vehicles that could be traffic conflicts. A set of conditions define if an interaction includes an evasion action or not. These conditions include instantaneous

time to collision, speed, acceleration rate, and jerk (longitudinal and lateral). The user applies these conditions to detect evasion actions in the interactions. The end user can modify the provided default values.

To complete determination if an interaction is a conflict, each of the events that have passed the first filter, is checked if it turns into a collision if no evasion action is present. The part of the vehicle's trajectory identified as evasion is replaced with a non-evasion sequence found among apparently non-evasive trajectories of vehicles that perform the same turning movement maneuver as the analyzed vehicle. Then, the program checks if a collision happens using the assembled non-evasive trajectory. If it does, the time to collision for that interaction is calculated.

The results are saved as comma separated values in a .csv file. These files are compatible with the Conflict Diagram Builder program that can be used to analyze the conflicts patterns.

8. IMPROVED IDENTIFICATION OF TRAFFIC CONFLICTS

This chapter provides more explanations of the traffic conflict identification process applied to traffic interactions and its connection with the existing theory. Following the theory postulated in Tarko (2018), an interaction between road users is considered a traffic conflict if there is a failure that produces a crash if one of the road users does not respond timely. Short proximity in space and time has been extensively used as an indication of failure in traffic conflicts.

Multiple indicators of extraction and characterization of traffic conflicts include time-to-collision (TTC), post encroachment time (PET), time to accident, gap time, time to departure, braking time, among others. Ismail et al. (2011) broadly categorized these indicators of traffic conflicts into the following two groups: (1) conflict measures requiring the presence of a collision course and (2) indicators based on the temporal proximity of road users. In the first case, one of the most representative measures refers to TTC, and in the second case, the most representative measure corresponds to PET. As extensively discussed by Tarko (2018), a counterfactual definition of traffic conflict requires the following two conditions: (1) very short separation of road users indicating failure and (2) the collision would happen if no evasion were performed. In the case of the application of PET, there is no need for a collision course, and conflicts could be cleared by themselves. It violates the postulate of the counterfactual case of a crash. Hence, a proximity measure without a collision point cannot be applied in the proposed method. This method supports the application of time to collision for the identification of traffic conflicts in this research project.

When considering methods requiring a collision point, there has been considerable heterogeneity in terms of the spatial or temporal separation values of TTC applied for specification of traffic conflicts. This difficulty has been translated into the definition by users of rather subjective thresholds applied in software programs, such as FHWA's Surrogate Safety Assessment Model (SSAM). SSAM was structured with the primary objective of providing identification, classification, and evaluation of traffic conflicts that occur in simulation models (Gettman et al., 2008). The tool includes an open-standard vehicle trajectory data format that could be obtained from a variety of simulation model vendors/developers including PTV(Vissim), TSS (AIMSUN), Quadston (Paramics), and Riosux Engineering (Texas). TScan expanded the methodology by including real, extracted trajectories into SSAM. The following procedure is applied by the algorithm for the extraction of traffic conflicts.

1. A zone grid is created based on the dimension information given by the header name of the input file, whose individual square zones cover 50-ft by 50-ft (15.25-m by 15.25-m) areas. This split procedure can effectively reduce the number of vehicle-vehicle comparisons necessary for identifying prospective conflicts.
2. Draw an appropriately projected path for each vehicle in the analysis region. One of the most simple and reasonable strategies is to predict the future path of the vehicle based on its current speed and detailed trajectory information. The forward distance is defined as the distance that vehicle *A* can travel at the current speed for the MaxTTC specified by the user in an interval equal to V1 * MaxTTC.
3. When the projected paths for the vehicles of interest are confirmed for the following MaxTTC time interval, the overlapping between two vehicles within each square zone then can be identified after the rectangular perimeter of each vehicle is known. Thus, some conflicting vehicle-pairs are identified for every time-step, and all the previously found conflicting vehicle-pairs would be maintained until the last time-step for analysis.
4. Based on the steps above, a more detailed analysis for each conflicting vehicle-pair can be conducted. This analysis refers to a more accurate estimation of TTC by iteratively shortening the future projection timeline by 0.1 seconds and reprojecting them as before until there are no overlaps.

An important definition of the method is the MaxTTC to input for the projection of the path. The analysis considers all the interactions with values lower than MaxTTC as traffic conflicts regardless of whether or there was a failure involved or dynamic characterization before the crash. However as reported in Tarko and Lizarazo (2021), experimental validation has shown, the heterogeneity of the drivers, risky driving habits, and sensor noise undermine the proximity measures as sufficient. Additional limitations from SSAM includes handling vehicle and vulnerable road users encounters. Hence, a more redefined methodology is applied in the conflicts' module for proper identification in line with the failure-based definition from Tarko (2018).

Tarko and Lizarazo (2021) identified two conditions must be met to properly claim observed traffic events as traffic conflicts: (1) analysis of longitudinal and lateral acceleration profiles for identification of avoidance due to failure and (2) estimation of the time-to-collision as

the period between the end of the evasion and the hypothetical collision. Extrapolating user behavior in the counterfactual scenario of no evasion is applied for identifying the hypothetical collision point. Also, it was found for vehicle-vehicle encounters that the anticipated impact speed should be sufficiently high to induce drivers' fear of collision. In general, the following procedure is applied for the identification of traffic conflicts in this framework.

1. Identify potential pairs in a conflicting situation based on extrapolation of the trajectory using an assumption of constant speed and direction (instantaneous time-to-collision). Extraction of potential pairs based on liberal thresholds (<5 seconds).
2. Among all the detected pairs, find the ones with a significant change in lateral or longitudinal acceleration (jerk, indicating potential avoidance action).
3. Find another vehicle free of evasion whose trajectory is close to the trajectory of the vehicle performing evasion.
4. In the pair of vehicles formed in step 1, replace the evading vehicle with the vehicle free of evasion found in step 3 and check if the two vehicles would collide.
5. If so, then estimate redefined time-to-collision based on the time between the ending of evasion and hypothetical collision point. Perform steps 2–5 for all the remaining potential pairs of vehicles in conflicting conditions from step 1.

This general procedure is implemented when the user presses the button of "Extract" conflicts in the user interface. More information about the engineering applications can be found in Volume 2 of the *TScan User Manual*, conflicts section.

9. DISCUSSION, RECOMMENDATION, AND FUTURE WORK

The recommendations for future improvement discussed in Chapter 7 of SPR-3831 report regarding improving tracking performance have been explored and appropriately implemented. The evaluation results in the previous chapter quantifies those improvements.

This chapter presents some areas where the current prototypes could be further improved. These improvements all apply to several aspects of the current system—convenience and reliability, hardware, algorithm execution speed, and engineering applications.

The most difficult step in the process of setting up TScan system is the alignment of 3D point cloud data from the LiDARs with the orthographic image in the field. As things stand, the user needs considerable training to properly align the two within a reasonably short time. If there are standing out and easily recognizable physical structures such as buildings or electric poles, the alignment process gets a bit easier. In rural areas and on county roads, where such objects do not exist, identifying the same spots in the visualized data cloud and on the orthographic image becomes challenging.

One area worth exploring to solve this issue is to use a magnetometer in combination with GPS signal. The magnetometer tells the system which direction cardinal north is and the GPS can help with the precisely locating (within 2–3 meters) the trailer unit, and therefore the origin of that unit's reference frame within the orthographic image. This provides a good starting point in the process of alignment and the user might just has to make some small adjustments to ensure good alignment.

Another possibility is to use obtain digital elevation models (DEM). A similar method to LiDAR self-calibration can be used to align the 3D DEMs with the LiDAR points automatically. Given that these DEMs are geotagged, the ortho image can be automatically aligned with the LiDAR points. Once aligned, DEMs provide the road surface "background" thus improving the background identification process and in turn significantly improving road user detection accuracy.

Adding cellular communication to TScan units would allow them to be remotely monitored. This would significantly increase its ease of use. Currently security is a big risk despite most of the expensive components being on top of the mast. The following are some examples of the types of notifications a user can receive once the TScan unit is fitted with cellular communication.

- Data collection status (time remaining) and statistics line (number of objects tracked).
- Battery level status.
- Any large vibration can be detected by the IMU and alert can be immediately sent to the user. Such large vibrations can occur due to the following.
  - Heavy wind.
  - Someone tampering with the system.
  - An accident involving the stationary trailer (i.e., someone collides into the trailer).

Regarding hardware, Ouster, the company behind the OS1-64 sensor used in the trailers has developed new long-range (~200 m) and short-range (~50 m but crucially with 45-degree downward firing angle) sensors with only downward firing lasers. A combination of these new generation sensors can potentially eliminate the need for pan tilt motors there by greatly simplifying initial setup process and may even provide better coverage than the prototypes developed as part of this report.

The post processing step currently is single threaded. This algorithm can be redesigned to be multi-threaded (i.e., use parallel processing) and improve its execution speed by a factor of 3 or 4. With TSFF providing detailed trajectories, engineering applications to analyze traffic operations like gap acceptance can be developed.

10. CLOSURE

The primary objective of this project was to develop two prototypes and accompanying engineering applications as envisioned in SPR-3831. As planned, two prototypes accompanied by a suite of engineering applications including one for identifying traffic conflicts have been developed.

The final design of the prototypes is the significantly revised initial vision developed in the SPR-3831. The primary reason of the change, as outlined in this report, is the change of a sensor configuration from a single Velodyne HDL 64 to a combination of Velodyne HDL 32 and Ouster 64. This change in architecture necessitated adding a mechanism to independently move the sensors. This modification allows a better coverage of intersections with the combined field of view of the two LiDAR sensors.

A new architecture had to be developed that would combine points from multiple sensors into a single frame of reference while benefiting from the added degrees of freedom of the sensor assembly movement. Information, such as readings from IMU sensors built in each LiDAR, feedback from servo motors, and surface terrain of an intersection seen by each LiDAR, were fused together in this new architecture to get an accurate estimation of each LiDAR's orientation in 3D space. This information was used to bring measurements from multiple LiDARs in the same unit into one common reference frame. Then, the additional use of a common orthographic image allowed readings from multiple masts to be brought to the same frame of reference.

A GPS based time synchronization method was implemented. Not only did this allow data from multiple sensors to be perfectly aligned in time but also data from multiple trailer units to be synchronized in the post processing step. Eventually, all the measurements and processing results from multiple sensors and masts are spatially and temporally aligned.

The use of multiple LiDAR sensors in a single mast increased the amount of data handled from a previously estimated 10 GB/hour to 30 GB/hour. Considerable time and effort were spent on converting the MATLAB code into the C++ code to be implemented while keeping memory usage within the limits and accomplishing real-time tracking with limited computational capabilities available. These efforts include developing new background identification algorithms, multithreaded processing architecture, and use of more efficient algorithms.

A new file format named TScan File Format (TSFF) has been created to replace the SSAM file format, which support only vehicles and other road user types. The new TSFF format also supports off-line merging of data produced by multiple TScan units.

The SSAM application computes instantaneous time to collision (iTTC) only for vehicles. A new conflicts detection method is based on time to collision (TTC) and not iTTC. It is accomplished by replacing the part of the trajectory affected by a conflict (evasion maneuver) with a matched conflict-free trajectory from trajectories collected by TScan at the same intersection during the same data collection period.

In addition to traffic conflicts, other engineering applications have been developed as part of this project. They include vehicle counting by turning movements, speed analysis within an intersection area, and pedestrian presence. TSFF format allows storing a quite wide range of information about each object, such as 3D dimensions, classification, position, velocity, acceleration, bounding box, and a cluster of measurement points that makes up the object. The stored data can be used in other traffic studies such as gap acceptance and other.

The TScan system captures 3D data and tracks vehicles and other objects moving across intersections by combining data from multiple units. It includes a novel conflicts identification technique that has an opportunity to be the world's first system that fully automatically implements traffic conflicts method to the engineering practice. As in a case of any research product, a number of improvements will possibly be identified in the implementation phase to make the TScan better and more convenient.

REFERENCES

City of West Lafayette. (2012). *Seasonally adjusted average daily traffic* [Tippecanoe County ADT]. Plan commission staff.

Gettman, D., Pu, L., Sayed, T., & Shelby, S. (2008). *Surrogate safety assessment model and validation: Final report* (Publication No. FHWA-HRT-08-051). Siemens Energy & Automation, Inc. https://www.fhwa.dot.gov/publications/research/safety/08051/08051.pdf

Ismail, K., Sayed, T., & Saunier, N. (2011). Methodologies for aggregating indicators of traffic conflict. *Transportation Research Record*, *2273*(1), 10–19. https://doi.org/10.3141/2237-02

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*(1), 35–45.

Tarko, A. P. (2018). Estimating the expected number of crashes with traffic conflicts and the Lomax Distribution–A theoretical and numerical exploration. *Accident Analysis and Prevention*, *113*, 63–73. https://doi.org/10.1016/j.aap.2018.01.008

Tarko, A., 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. https://doi.org/10.5703/1288284316347

Tarko, A. P., & Lizarazo, C. G. (2021). Validity of failure-caused traffic conflicts as surrogates of rear-end collisions in naturalistic driving studies. *Accident Analysis and Prevention*, *149*, 105863.

Tarko, A., Romero, M., Ariyur, K., Bandaru, V., & Lizarazo, C. (2018, May 16–18). *Detecting and tracking vehicles, pedestrians, and bicyclists at intersections with a stationary LiDAR* [Paper presentation]. 18th International Conference Road Safety on Five Continents (RS5C 2018), Jeju Island, South Korea. http://urn.kb.se/resolve?urn=urn:nbn:se:vti:diva-12915

Velodyne. (2021). *HDL32-E user manual* [PDF file]. Retrieved March 16, 2021, from https://velodynelidar.com/products/hdl-32e/#downloads

APPENDICES

Appendix A. Prototype Hardware

Appendix B. SSAM File Format Specification

Appendix C. TScan Output File Format

Appendix D. TScan User Manual

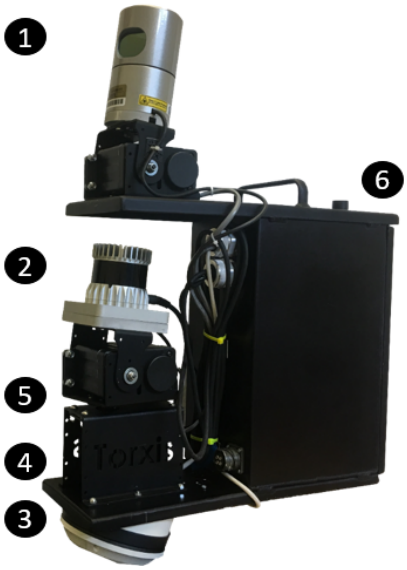
Appendix E. TScan Engineering Applications

APPENDIX A. PROTOTYPE 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 3D 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 and Figure A.1.



Components

Sensors

1. Velodyne HDL32E

2. Ouster OS1-64

3. Vivotek Fisheye SF8174V Camera

4. Vectornav VN-200 IMU

Servos

5. Torxis IO1855 Industrial Servos

Support

6. Aluminum anodized C-Shape.

Figure A.1 Prototype II head.

A.1 Sensors and Cameras

The sensors and cameras installed on the TScan Heads are the two LiDAR units, an IMU, a GPS receiver and a fish eye camera.

A.1.1 LiDARs

The two LiDARs used in the prototypes are Velodyne HDL-32E and the Ouster OS1-64 (Figure A.2). The important specifications of both LiDARs are shown in Table A.**1**.

Table A.1 Sensor Manufacturer Specifications for Both LiDARs

Sensor	Velodyne HDL 32	Ouster OS1 64
Number of lasers	32	64
Horizontal FOV	360	360
Vertical FOV	+10.67° to -30.67° (41.33°)	+16.6 to -16.6
Range–10% reflectivity	2 m–50 m	0.8 m–40 m
Range–80% reflectivity	2 m–100 m	0.8 m–105 m
Angular resolution (Vertical)	1.33°	0.6°
Rotation rate	5–20 Hz	10–20 Hz
Points per second	~695,000	~1,310,720

Figure A.2 Velodyne HDL-32E and Ouster OS1-64 LiDARs.

A.1.2 IMUs

TScan will require an IMU that accurately measure the orientation of the head unit. The IMU used is a Vecotrnav VN-100 whose specifications are given in the Table A.2.

Table A.2 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:	±2,000 °/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	

A.1.3 GPS

Taking into account that the LiDAR sensor can synchronize its data with precision GPS-supplied time pulses over a dedicated port, the GPS device 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.
- The sync pulse length is not critical (typical lengths are between 20 ms and 200 ms). Start the $GPRMC record between 50 ms and 500 ms after the end of the sync pulse.

The GARMIN GPS 18x LVC is the GPS sensor (Figure A.3) 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.

Figure A.3 Garmin GPS receiver.

A.1.4 Camera

In order to cover the entire intersection and the base of the trailer at the same time a fisheye IP camera with a field of view of 360 degrees should be used (Figure A.4). A 5 megapixel sensor is recommended with a 30 fps at 1,080 p Full HD.

Figure A.4 Vivotek SF8174V 5MP 360° fisheye camera.

A.2 Computer and Communications

A.2.1 Computer

The minimum computer specifications required to run the real time data collection process are given in the following table (Table A.3). The computers used in the trailer units either match or exceed the performance of the specifications provided in the table below.

Table A.3 Computer Specifications

Processor	4th Generation Intel Core i7-4770T Processor or better but with a TDP of 35W and max power consumption of 45W
Operating System	Windows 10 or higher, 64-bit
Memory	32 GB, 1,666 MHz or higher, DDR4
Hard Drive	SSD 512 GB
Graphics Card	Intel HD 4600 or better
Ports	1 USB 3.0 ports 5 USB 2.0 Ports 1 HDMI or Display Port 2 RJ-45 (10/100/1000Base)

A.2.2 Communication Devices

Two standard Gigabit Ethernet switch with at least 4 ports are used to split the communications into two independent networks. One present at the top of the mast connecting the LiDARs and camera with the computer. The other with Wi-Fi present below the mast to allow the user to remote control the computer in the head unit.

Table A.4 shows the power consumption estimate used to estimate the power requirements.

Table A.4 Power Consumption Estimation

Equipment	Power Consumption	Voltage
LiDAR	2 × 20 watts	12-24 VDC
IMUs	0.7 watts	5 VDC
Computer	65 watts	12 VDC
External storage	5 watts	USB powered
Camera	25 watts	12 VDC
Router	30 watts	12 VDC
Total	155 watts	

A.3 Infrastructure

A.3.1 Trailer with Mast

A portable tower trailer is used with batteries, charger and a 30-ft folding mast included. The trailer has stabilizers for normal operation (Figure A.5).

Figure A.5 SolarTech portable tower trailer.

The solar tech trailer used has stabilizing jacks to level the platform. It can sustain wind speeds up to 80 miles per hour and has enough battery capacity to power the TScan system for 2–4 days depending on external temperature.

A.3.2 Pan-Tilt Base

A pan tilt base to control the LiDAR orientation is included to individually orientation of the LiDARs (Figure A.6). The main LiDAR requires only tilt since the mast provides pan capabilities. The secondary LiDAR requires both Man and Tilt. The Pan-Tilt base specifications are shown in Table A.5

Table A.5 Pan-tilt base specifications

Rotation	0 to 270 deg
Tilt	-10 to 90 deg or more
Maximum load	1,600 oz-in (11.3 N-m)
Remote controller	USB servo controlled
Voltage	12 VDC

Figure A.6 Troxis servo controlled pan/tilt.

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 seen in Table B.1.

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

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 B.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.

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

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.

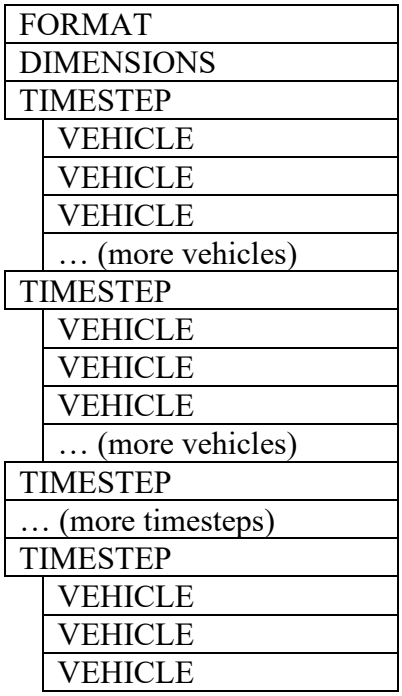


Figure B.1 Organization of records in the trajectory file.

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 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.

Table B.3 Dimensions Record Description

Field Name	Type	Value Description
Record Type	Byte	1 = DIMENSIONS record type
Units	Byte	0 = English (i.e., feet, feet/sec, feet/sec^2) 1 = 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.

Each time-step of vehicle data begins with a TIMESTEP 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.

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

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.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 binary value format. The file is organized as a list of objects. Each object has its time invariant properties listed followed by location of time dependent properties in time dependent file. The organization is shown in Table C.1.

Table C.1 Organization of Records in the Time Independent File

Interpolation byte
Object 1: Time independent properties (51 bytes) Object 1: Time dependent values location ($k_1 \times 8$ bytes)
Object 2: Time independent properties (51 bytes) Object 2: Time dependent values location ($k_2 \times 8$ bytes)
… more data records
Object n: Time independent properties (51 bytes) Object n: Time dependent values location ($k_n \times 8$ bytes)

The interpolation byte is set to 1 if the user requests for time dependent values to be estimated when the object is occluded in the field of view. It is set to zero by default.

The number of location values present depends on the number of frames the object was tracked which in turn can be calculated from the time independent values as shown in the following equation:

$$k_n = E_n - S_n + 1 \qquad \text{(Eq. C.1)}$$

Where,

k_n is the number of frames the object was tracked;

E_n is the last frame the object was present in the field of view of the system; and

S_n is the first frame the object was present in the field of view of the system.

If an object was occluded in a particular frame, and interpolation is turned off, the location value would be set to 0.

The time independent file contains the variables listed in Table C.2 for each object.

Table C.2 Time Independent Variables Description

Variable Name	Description	Type
ObjectID	Unique identifier of the object	uint_64
ObjecctTrackingID	Internal ID used for tracking	uint_64
Start Frame	First frame when the object entered the field of view	uint_64
End Frame	Last frame the object was within field of view	uint_64
Intersection Frame	First frame the object entered the intersection polygon	uint_64
Length	Measured object length in meters	uint_16
Width	Measured object width in meters	uint_8
Height	Measured object height in meters	uint_8
Tracking Status	Tracking status of object	uint_8
ObjClassification	Classification of the object	uint_8
Polygon Path	List of user defined polygons traversed by the object	5 × uint_8

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 Table C.3.

Table C.3 Organization of Records in the Time Dependent File

Interpolation Byte	
Object 1	Frame S_1 Time dependent values
	Frame $S_1 + 1$ Time dependent values
	Frame $S_1 + 2$ Time dependent values
	…
	Frame E_1 or $S_1 + k_1$ Time dependent values
Object 2	Frame S_2 Time dependent values
	Frame $S_2 + 1$ Time dependent values
	Frame $S_2 + 2$ Time dependent values
	…
	Frame E_2 or $S_2 + k_2$ Time dependent values
… (more vehicles)	
Object n	Frame S_n Time dependent values
	Frame $S_n + 1$ Time dependent values
	Frame $S_n + 2$ Time dependent values
	…
	Frame E_n or $S_n + k_n$ Time dependent values

The time dependent values file contains the variables listed in Table C.4.

Table C.4 Time Dependent Variables Description

Variable Name	Description	Number of Values	Units	Type
Speed	Instantaneous forward speed	1	0.5 cm/s	Int_16
Acceleration	Instantaneous forward acceleration	1	0.5 cm/s^2	Int_16
Angle	Orientation with respect to LiDAR	1	0.5 cm	Int_16
Box	5 × (X, Y) coordinates of corners	10	0.5 cm	Int_16
Centroid polygon ID	Polygon ID of the centroid of the bounding box	1		Int_16
Convex hull vertex count	Number of points in convex hull c	1		Int_16
Convex Hull vertex locations	c × (X, Y) coordinate of corners	$c \times 2$	0.5 cm	Int_16
Point cloud points count	Number of points in convex hull p	1		Int_16
Point cloud points location	p × (X, Y, Z, Polygon ID, Surface Z estimation)	$p \times 5$	0.5 cm	Int_16

The reported angle is computed as shown in Figure C.1.

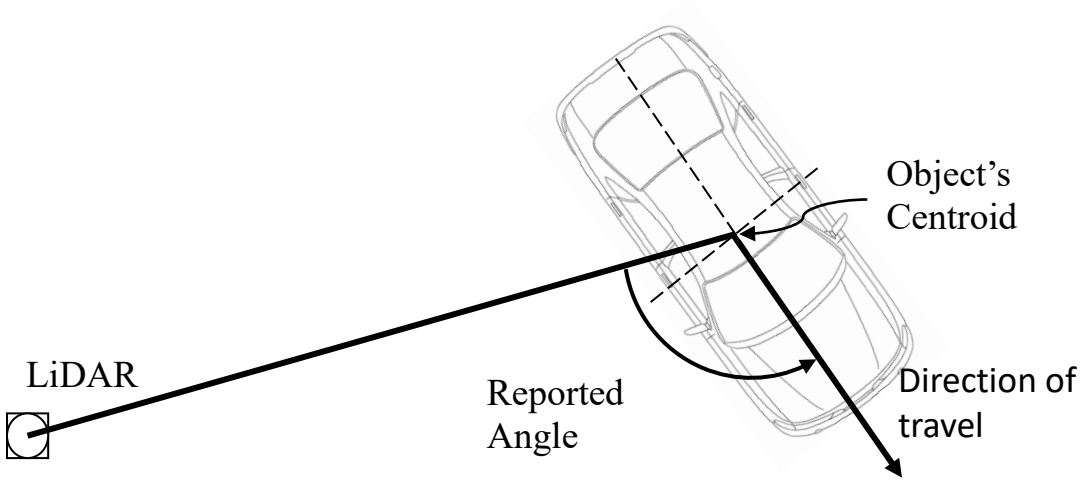


Figure C.1 Angle of object with respect to LiDAR.

APPENDIX D. TSCAN USER MANUAL

D.1 Introduction

TScan uses LiDAR technology that can detect and track various types of road users including buses, cars, pedestrians, and bicycles. 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 TScan prototype was developed by the Purdue Center for Road Safety as part of a research project funded by the Indiana Department of Transportation through the Joint Transportation Research Program. This user manual describes the overall process of preparing the system and then it focuses on the initial preparations executed in an office (off-site setup).

Setting TScan includes two phases (Figure D.1).

1. Off-site preparation.
2. On-site setup.

Once TScan data are collected, processed, and retrieved on site, they can be further processed off site with engineering applications. This first volume of the manual introduces the software TScan Offsite Setup and guides the user in creating, saving, and retrieving the intersection's characteristics to transfer it later to the field data collection. Part II: TScan Onsite Setup discusses the required steps to collect data on the field.

The off-site process allows the user to enter, save, and retrieve the intersection's characteristics in a graphic environment to be used by the onsite setup, the data collection and processing program and other engineering applications. The information required by TScan data collection and processing program is a list of polygons that define the road lanes, intersection areas, parking areas, sidewalks, and medians along with the corresponding intended maneuvers. In order to create the required polygons, the user selects or uploads an orthographic image that will be used to draw the polygon edges. This can also be used in other engineering applications. Although it is not necessary to create a new folder, it is advisable to do so in order to have all the study information regarding a particular site in a single folder.

In the off-site preparation, the user enters, edits, and saves intersection characteristics that are later transferred to the TScan Onsite Setup and data collection. *TScan Offsite Setup* software was developed to facilitate the following operations: (1) Selecting and uploading orthographic-images of intersections, (2) Drawing polygons that represent the intersection areas and (3) Entering polygon properties such as polygon type, and traffic maneuvers executed in the approach polygons.

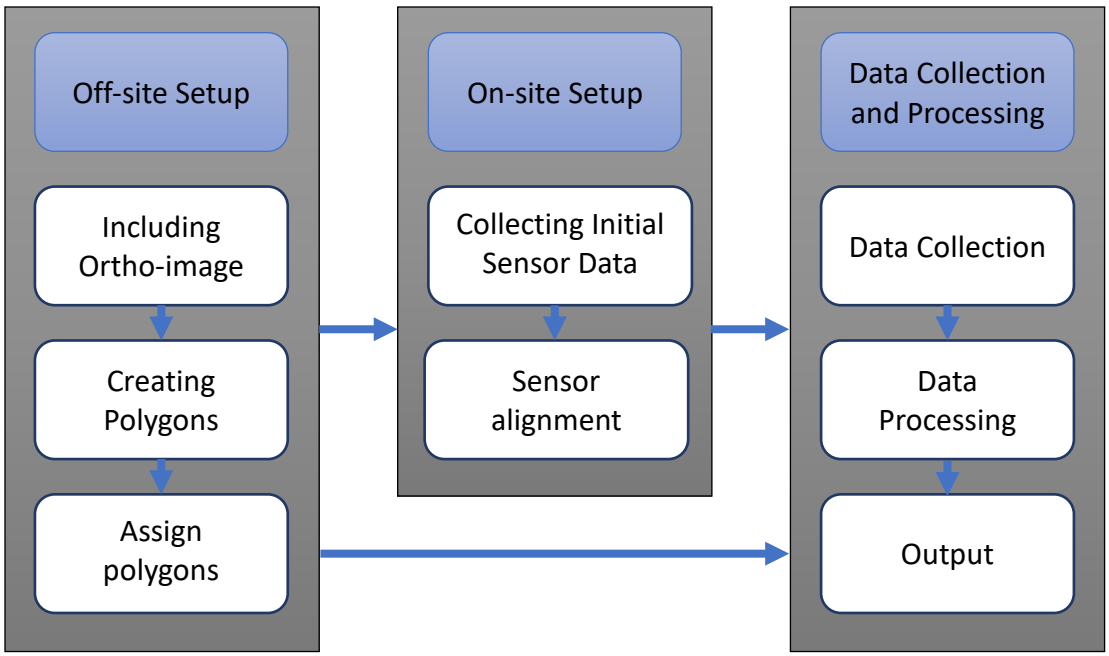


Figure D.1 TScan process overview.

TScan Offsite Setup software helps the user prepare site information needed to set the system in the field. The off-site preparation produces a setup file with the intersection polygons and other site characteristics. This file is needed in the on-site setup to establish a single spatial reference shared by the TScan sensors and the orthographic-image of the intersection to relate the collected data to the intersection and the GIS coordinates. It is accomplished by overlaying the LiDAR-collected data with the orthographic image of the intersection. Characteristic elements of the intersection recognizable on the image and in LiDAR data are aligned by the user in the field with help of the on-site software. The alignment information is saved and exported to the TScan Data Collection and Processing module.

After the setup information is obtained, the Data Collection and Processing module receives the data from the sensors, processes it, and produces an output file. No user intervention is required while collecting data unless there is an indication of malfunctioning.

The output file produced as the result of data collection and on-site processing is useful for many engineering studies including traffic volume and speeds, traffic performance (queues and delays), capacity analysis, warrants for control devices (signalization, stop signs, etc.), pedestrian studies, red signal compliance, traffic conflicts, etc.

D.2 Intersection Representation

The intersections in TScan are represented as a set of functional areas that determine both the potential users, the possible movements, and the operational function. These areas can be approaches, exits, intersection areas, parking areas, medians, or pedestrian areas. It is important to note that the areas must be interconnected.

- Approaches are areas that cover one vehicle arrival lane to the intersection. Each of these lanes is assigned the possible movements that vehicles can make, such as left or right turn, or trough movement. The approach zone should be at least the length of one vehicle but the length of three vehicles is preferable.
- An exit area is the set of lanes after vehicles leave the intersection area. In general terms, it is not necessary to divide the exit area by lanes. The exit area should be similar to the adjacent approach area.
- Parking area is a lane adjacent to and parallel with the travel lane of a roadway that is used for parking vehicles.
- The intersection area is the zone shared by different vehicular movements or pedestrians. Usually starting at the stop lines.
- A median area is the portion of the roadway separating opposing directions of the roadway.
- Pedestrian areas are zones adjacent to the roadway that are reserved for pedestrian-only. Usually, pedestrian areas are defined adjacent to the intersection polygon and an approximate width of two meters (7 ft.) is used.

This intersection representation is important for tracking objects. They define the areas of interest to select the lidar points for further analysis. Points outside of the defined areas are not considered. It also helps to define the intersection background and surfaces, to classify objects and to check if an object tracking is completed.

To define the intersection, an orthographic image is obtained on which the edges of the areas of interest are drawn. Then, the zones are selected, and the characteristics should be completed.

D.3 TScan Offsite Setup

TScan off-site configuration software helps the user to prepare the site information required to configure the system in the field. Off-site setup produces the configuration files corresponding to the data collection site that include the intersection representation for TScan, its characteristics, and an orthographic image.

D.3.1 Software Installation

TScan Offsite Setup is compatible with Windows 10. In order to run the program the MS .NET 4.5.2 Framework or later component must be installed. If the MS .NET 4.5.2 Framework is not present during the installation, TScan will attempt to install this component if the PC is connected to the Internet.

To install *TScan Offsite Setup*, follow the steps given below.

1. Extract the contents of the archived file to your local drive.
2. Click on the *setup.exe* file to initiate the installation.

After checking that the program is installed and 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.3.2 Launching TScan Offsite Setup

Once installed, the *TScan Offsite Setup* program can be launched using any of the following methods:

Method 1: Double click the shortcut on the desktop

Method 2:

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

Method 3:

1. Press *Start* button.
2. Start typing *TScanUI*, 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)/TScanUI/'*
3. Click on *TScanUI.exe*

D.3.3 Offsite Setup Interface

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

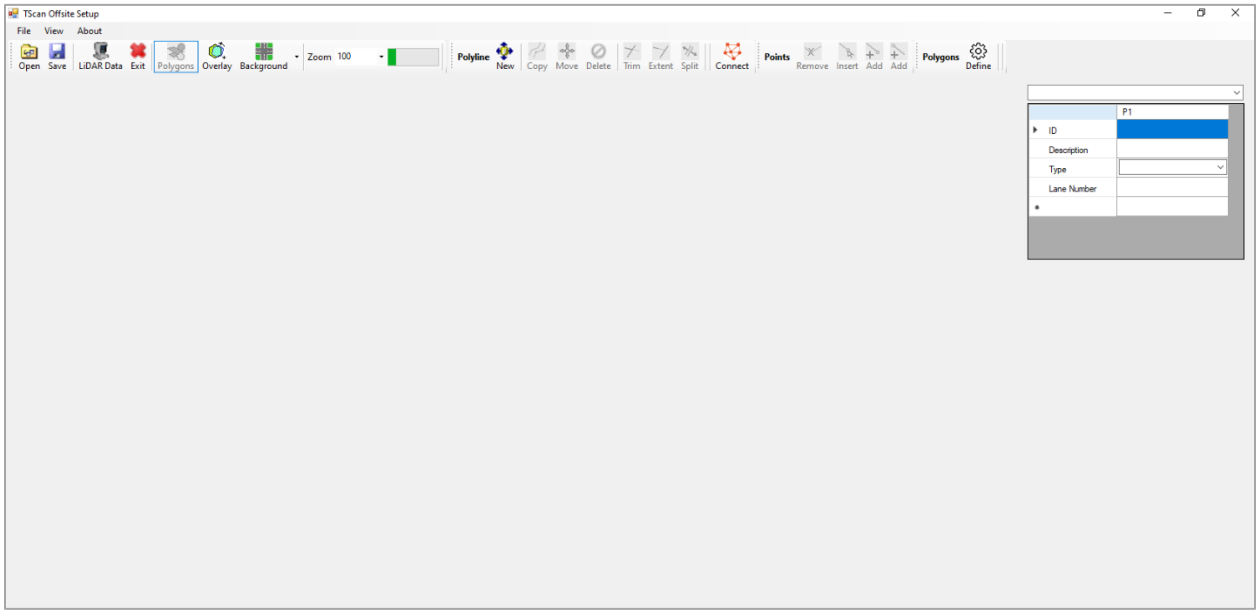


Figure D.2 TScan offsite setup process main interface.

The program interface includes a menu and command bar (Figure D.3). The menu includes tabs: *File, View*, and *About*. The *File* tab facilitates the process of saving or opening existing projects as well as exiting the program. The *View* tab allows the user selecting the option to visualize the command bar (second row) with only *icons*, *icons* + *text* or only *text*. The default configuration of the software provides the option of *icons* + *text*. The commands facilitate the operations on files, polylines, points, polygons, and LiDAR alignment (Figure D.3).

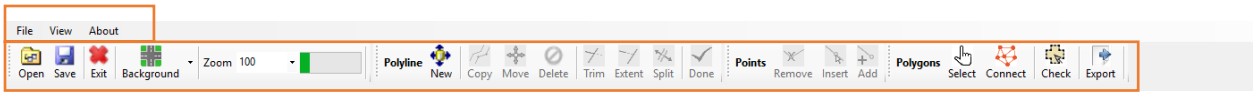


Figure D.3 Menu and command bars in the interface.

D.3.4 Background Image

A background image is needed to help the user develop an electronic representation of the intersection. *TScan Offsite Setup* offers three options of bring the background image: (1) capturing from Google Maps, (2) capturing from Bing maps, or (3) uploading from a file.

When capturing the images from Google or Bing maps, the original scale of the image must be preserved. To meet this condition, the computer *Display* settings must have the scale value in the *Scale and layout* option set at 100% (Figure D.4) .

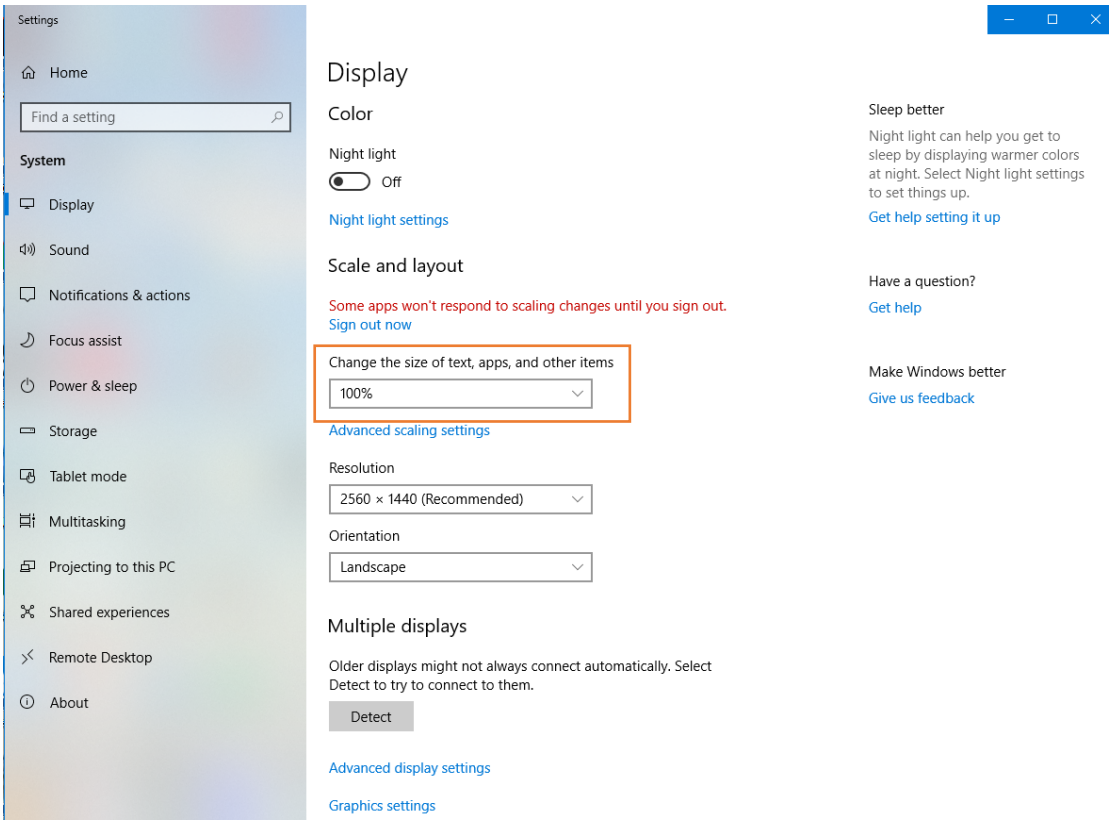


Figure D.4 Display settings: scale and layout 100%.

Select the desired source of the image from the drop-down menu in *Background*[1] of the *TScan Offsite Setup* (Figure D.5). Once the source is selected and the connection with either Google or Bing maps is established, look for the intersection of interest. Use the *Capture*[2] command button to store the background image as shown in Figure D.5. The user can quickly find the intersection by entering the latitude and longitude[3] of the intersection and pressing the *Go to*[4] button.

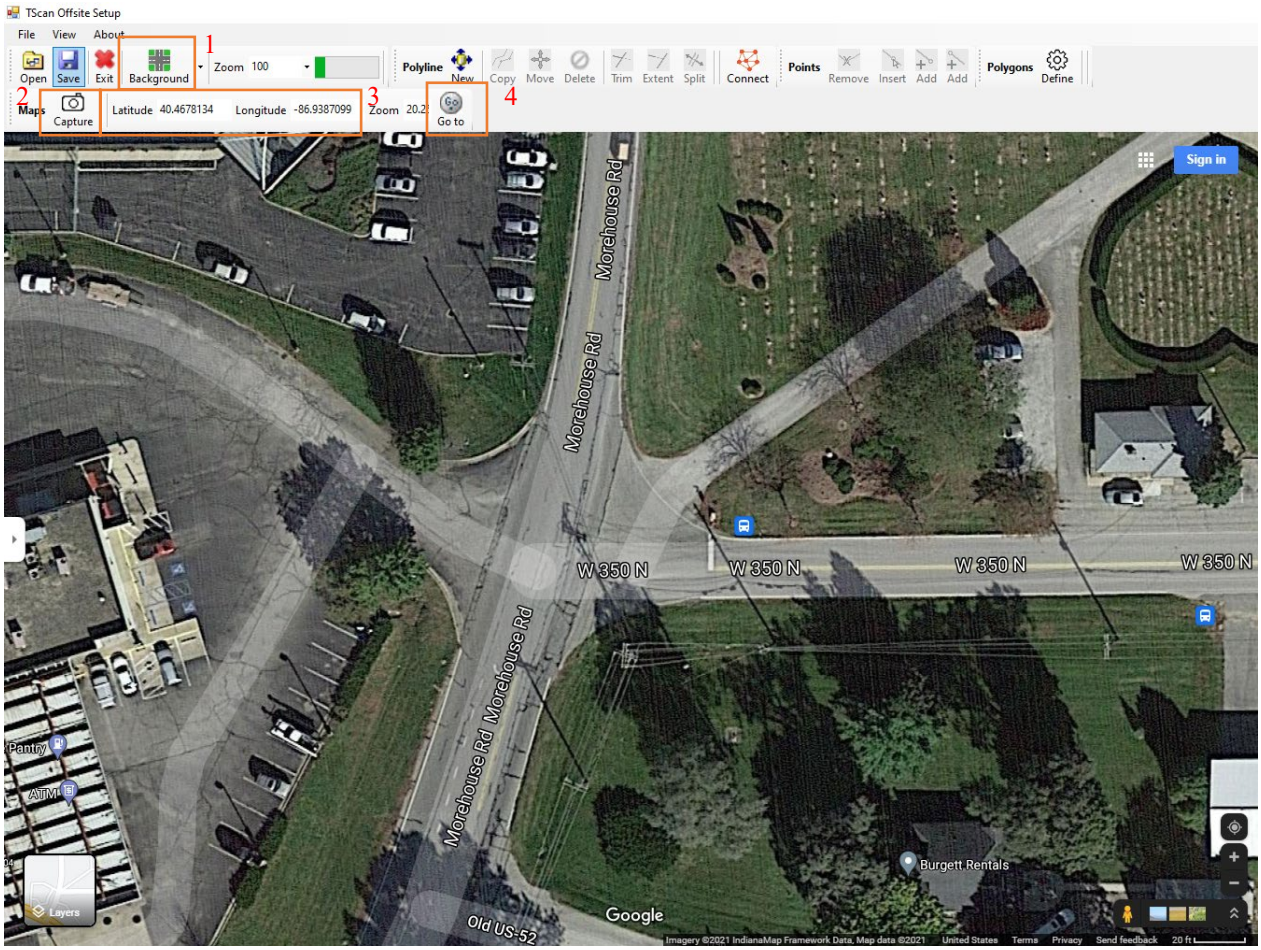


Figure D.5 Capture of orthogonal image to generate intersection layout.

The user has the additional option of uploading an image from a file. Select the option *From File* on the *Background* drop-down menu to open the Windows file selection window shown in Figure D.6 and select the folder where the file can be found and select the file.

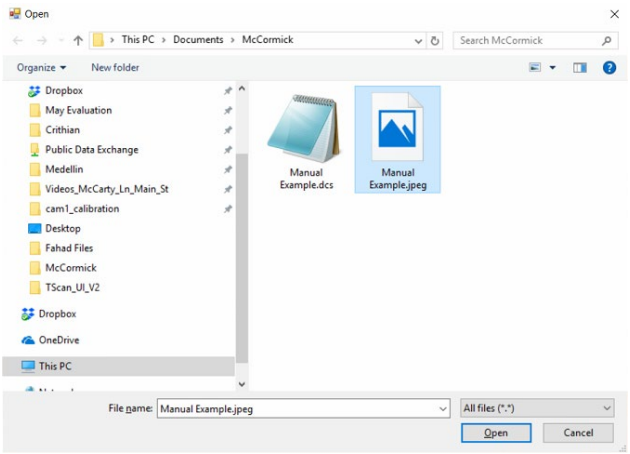


Figure D.6 Open background image from file.

It is important that when capturing the image some easily distinguishable nearby elements such as poles or buildings are visible in the image.

Once the image is uploaded, the scale of the displayed image can be adjusted by selecting one of the available zoom factors (Figure D.7).

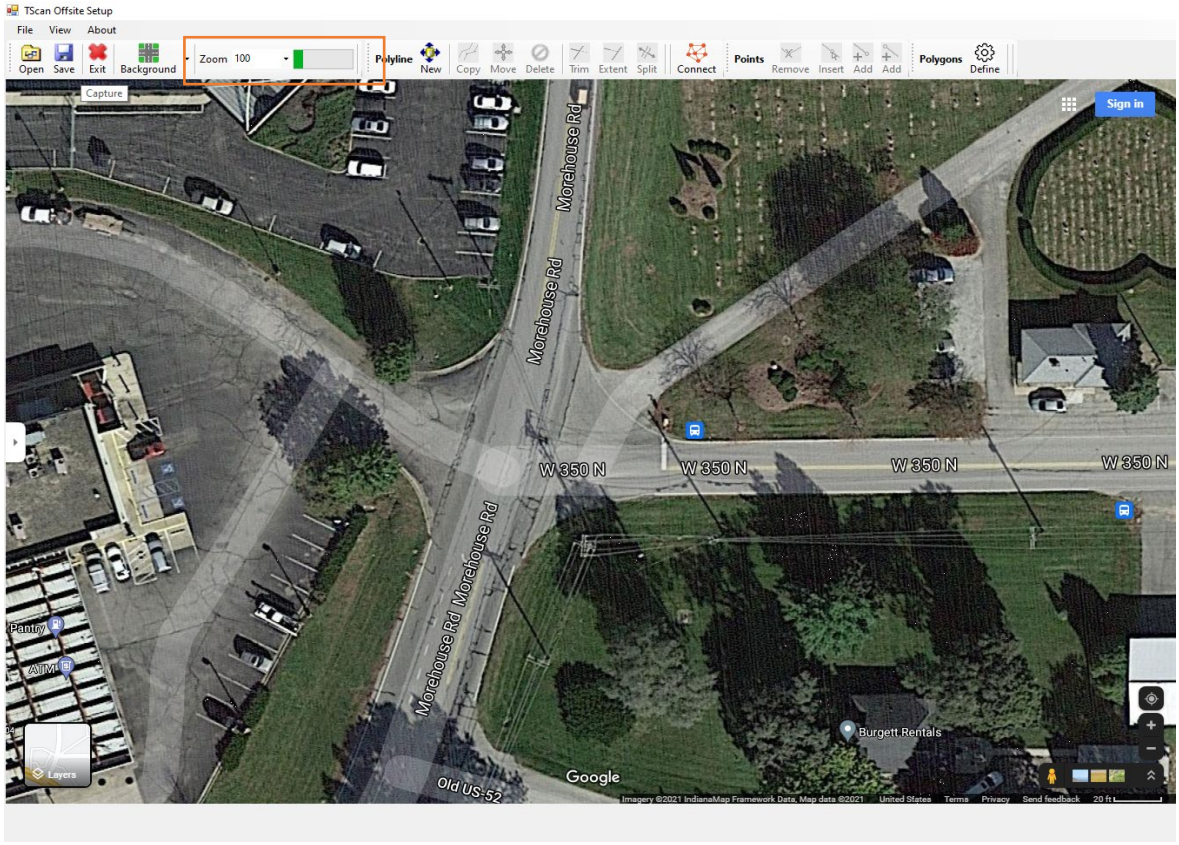


Figure D.7 Adjust background zoom factor.

D.3.5 Intersection Layout

Certain elements of the intersection need to be defined to help TScan detect objects moving within the intersection. Convenient drawing tools are available for this task in the polyline command bar and the point command bar. Table D.1 shows the supported operations including drawing polylines, adding points, and defining polygons.

Table D.1 List of Supported Operations for Various Polygon-Related Features

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

D.3.5.1 Polyline commands

Polyline is useful to represent the intersection layout. It is a continuous line composed of one or more straight segments. The polyline command bar includes tools for adding, deleting, copying, moving, trimming and, extending polylines as shown in Figure D.8.

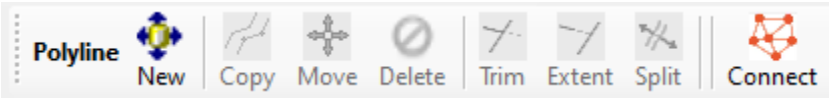


Figure D.8 Polyline tools in the command bar.

Adding Polylines

The first step of drawing the intersection layout is adding polylines. You can create a polyline by clicking on the *New* button and then specifying the vertices of the polyline by clicking at the desired points on the orthographic-image. Once all vertices are created, right-click on the image to finalize adding points to the polyline. The vertices of the new polyline will be deemphasized. Figure D.9 shows an example of adding polylines.

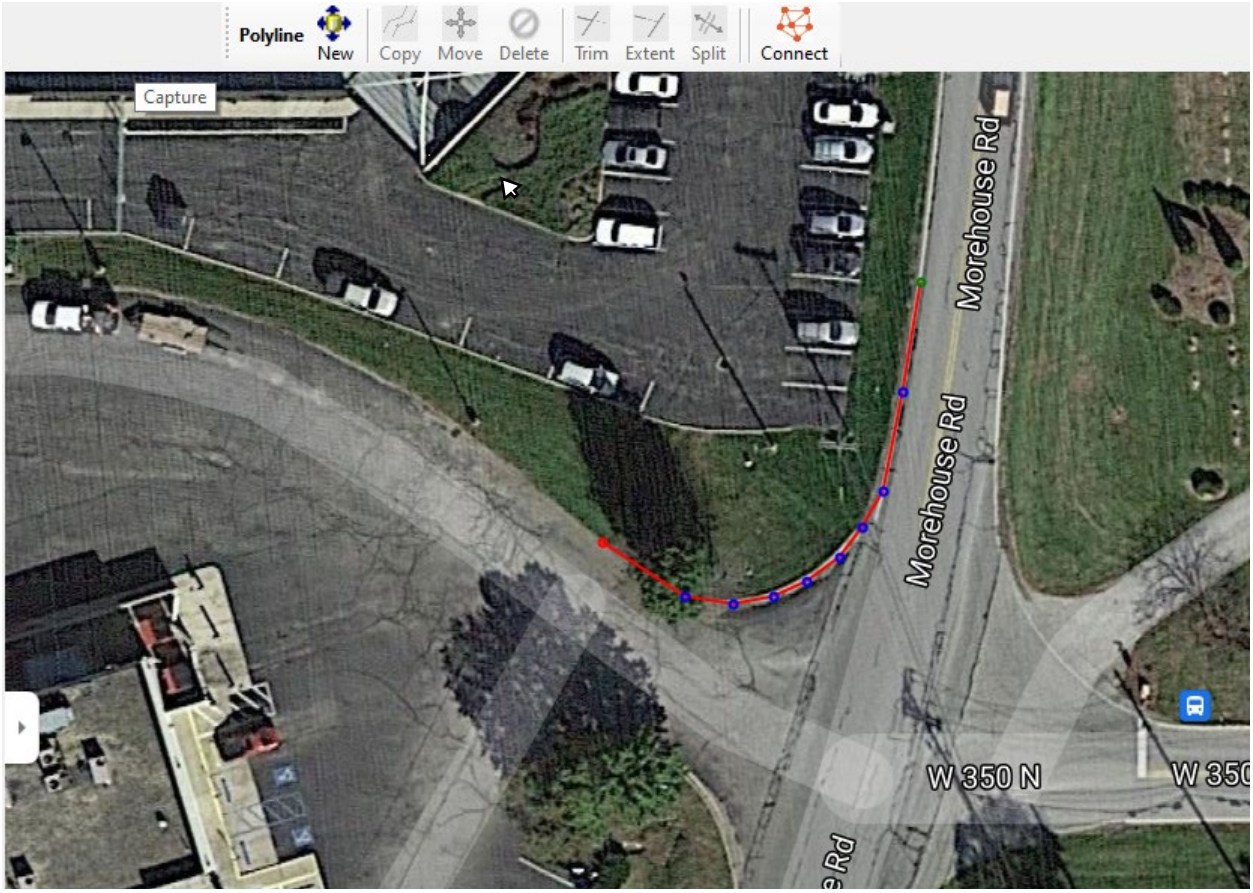


Figure D.9 Adding polylines.

POLYLINES SHOULD NOT INTERSSECT THEMSELVES. The program does not support drawing closed polylines. Closed areas can be created by allowing consecutive polylines intersect each other (Figure D.10).

(a) Incorrect–closed area with a single polyline

(b) Correct–no closed areas with a single polyline

Figure D.10 Avoid closed areas within a single polyline and ensure polylines to intersect.

Deleting Polylines

To delete a polyline, select the polyline by clicking on it. The vertices are displayed to confirm that a polyline is selected and ready for editing. Then, click on the *Delete* button to remove it from the image. Figure D.11 shows an example of deleting polylines.

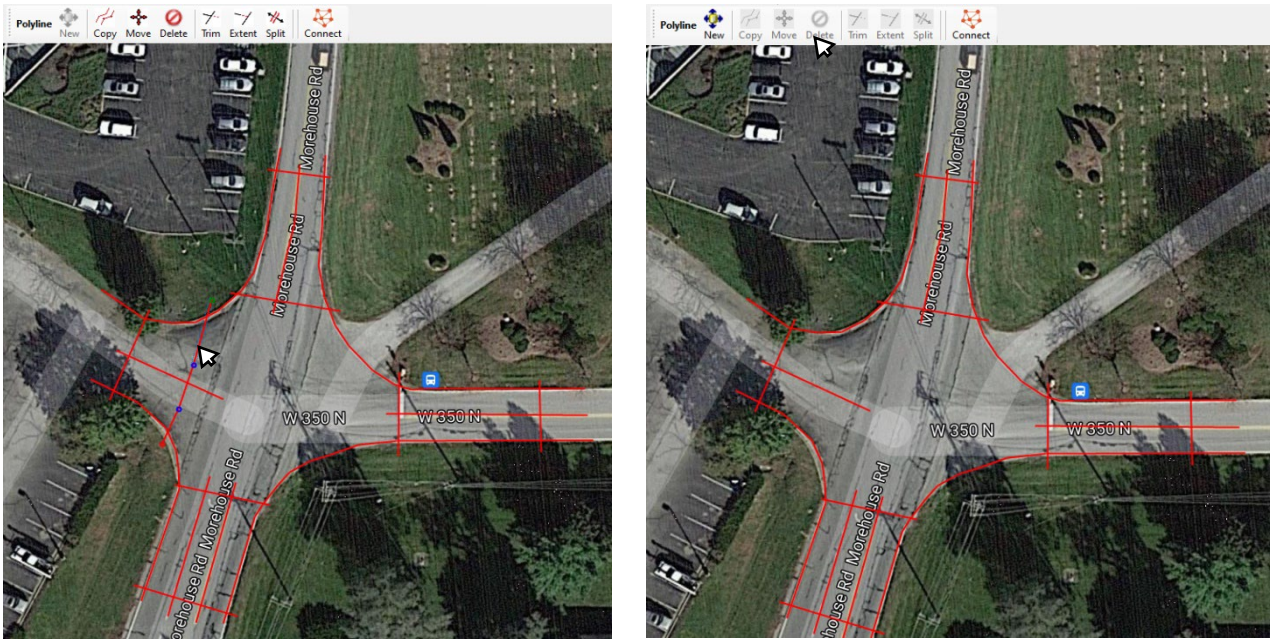


Figure D.11 Deleting polylines.

Copying Polylines

To create several parallel polylines, click on a polyline (vertices are emphasized), click on the *Copy* button, and then click on the image at the point where the new polyline is supposed to begin. Each next click generates another copy of the polyline. Right-click on the image to end copying polylines. Figure D.12 shows an example of a copied polyline.

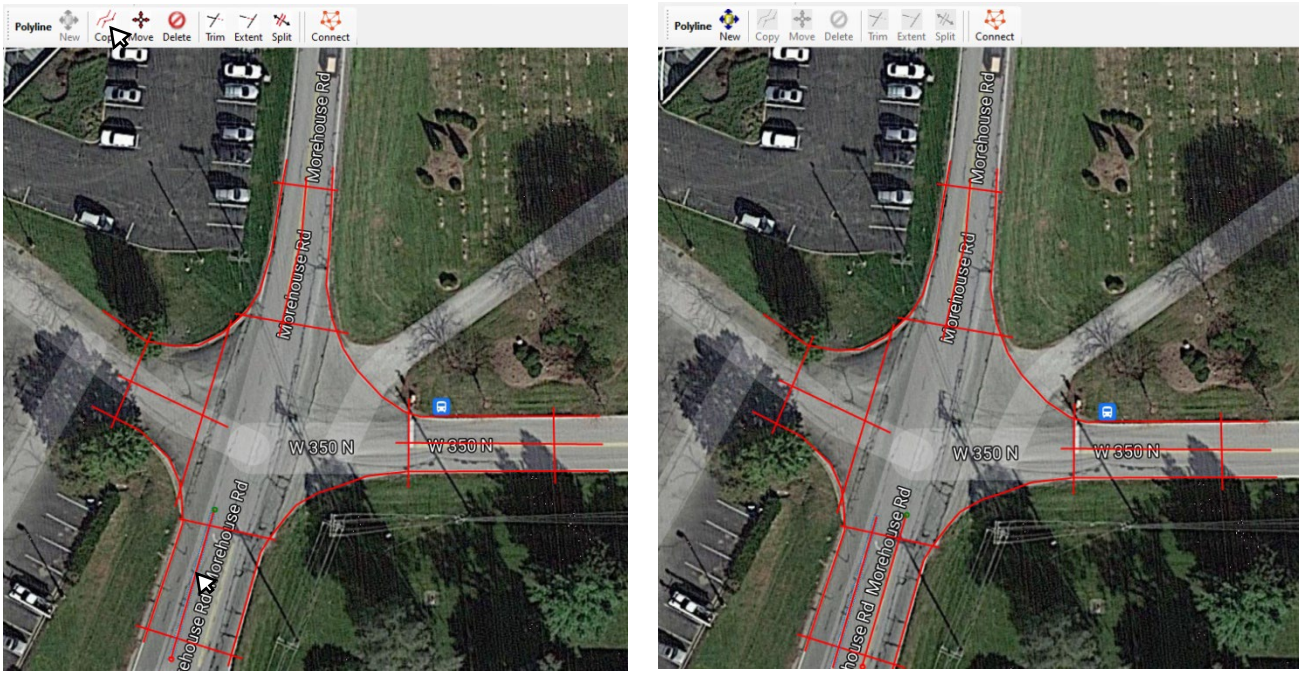


Figure D.12 Coping a polyline.

Moving Polylines

Select the polyline to be moved to highlight it. Then, click on the *Move* button and pressing the mouse left button, move the polyline to the desired location and release the mouse left button to finish. Right-click on the image to end moving the polyline. Figure D.13 shows an example of a moved polyline.

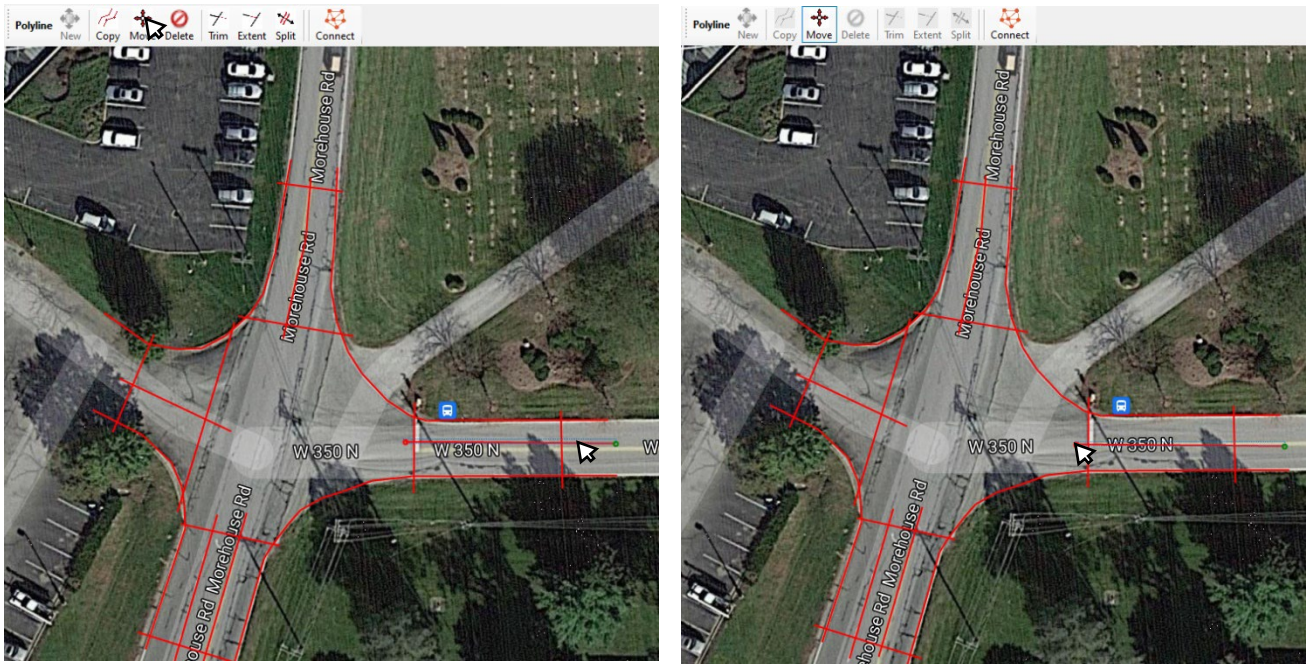


Figure D.13 Moving a polyline.

Trimming Polylines

To trim a polyline at the crossing point with another polyline, select the polyline, which will remain untrimmed (vertices will be emphasized). Click on the *Trim* button and the selected base polyline will change its color to yellow. Click on an intersecting polyline on the side to be removed. Repeat this operation to trim other intersecting polylines. Finally, right-click on the image to end the trimming operation. Figure D.14 shows an example trimmed polyline.

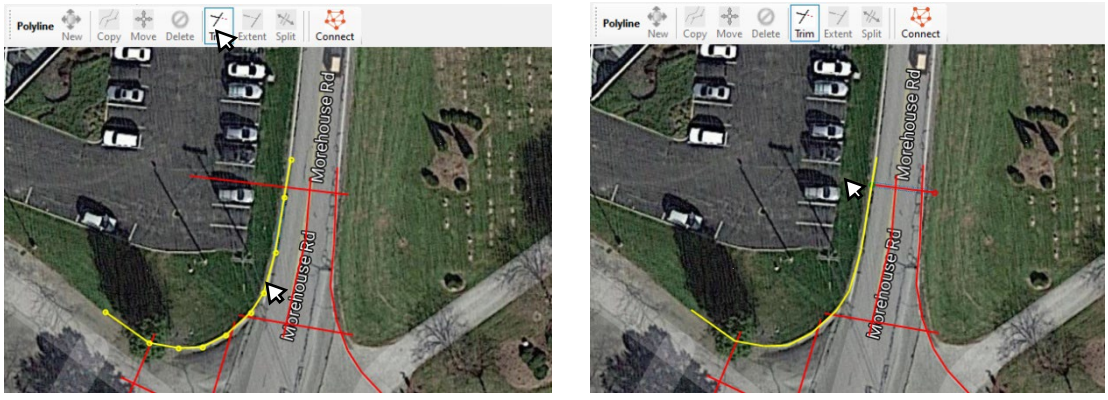


Figure D.14 Trimming a polyline.

Extending Polyline to Intersection with Another Polyline

To extend a polyline to an intersection with another polyline, it is necessary to select the other polyline, which is to remain unchanged. After selecting that polyline, its vertices become emphasized. Click on the *Extend* button. The selected base polyline will change to yellow. Then click on the polyline to be extended on the side where the extension needs to be made. Repeat this operation to extend other polylines. Finally, right-click on the image to end extending the polylines. Figure D.15 shows an example of an extended polyline.

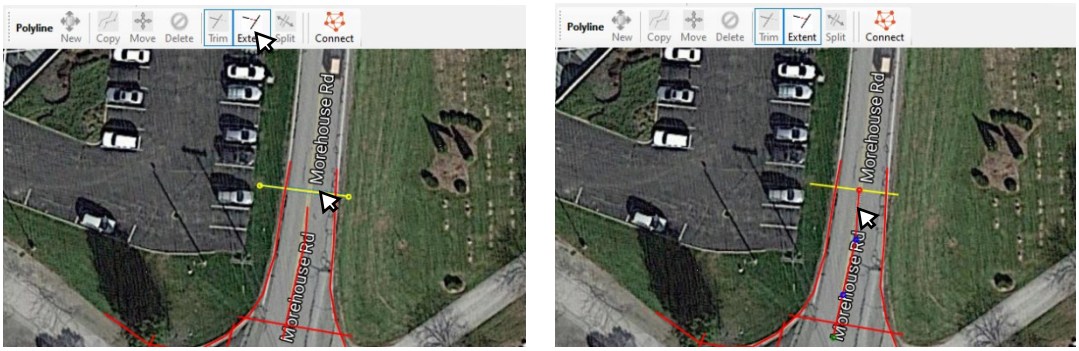


Figure D.15 Extending polylines.

D.3.5.2 Vertex commands

Another option to edit the polylines is to edit their vertices. The point command bar tools include functions for editing, removing, inserting, and adding polyline's vertices as shown in Figure D.16.

Figure D.16 Points tools in the command bar.

Edit Vertices

To edit the vertices, first select the polyline by clicking on it. After selecting, the vertices are displayed on the screen. To change the vertex's location, simply click on the vicinity of the vertex, drag it to the new position, and release the mouse button. Once all the changes of vertices' locations are executed, right-click on the image to finalize editing the vertices. Figure D.17 shows an example of moving vertices.

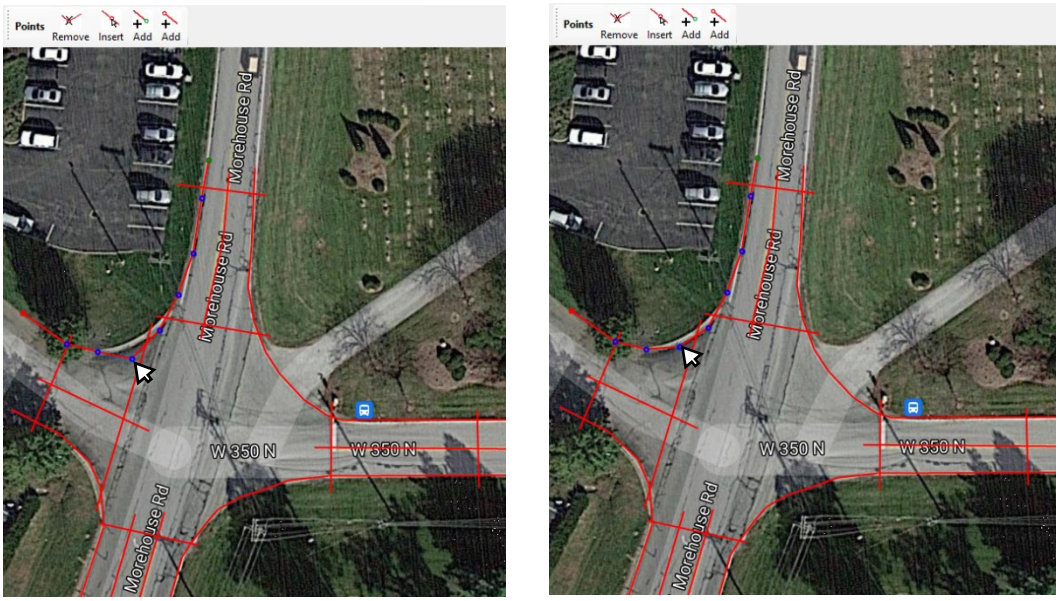


Figure D.17 Editing vertices.

Insert Vertices

To insert vertices, first select the polyline by clicking on it; the vertices are highlighted. Click on the *Insert* button to activate the function. Then, click on the polyline where the new vertices should be added. To end adding vertices, right-click on the image. Figure D.18 shows an example of inserting vertices.

Figure D.18 Inserting vertices.

Extending Polyline by Adding a Vertex

The difference between inserting and adding vertices is that inserting adds points between two consecutive vertices while adding vertices creates an extra vertex at an end of the polyline. To add vertices, select the polyline by clicking on it. After selecting, the vertices are displayed on the screen, the extreme vertices of the polyline are shown one in red and the other in green. The click on the *Add* with red dot button to activate the function that add vertices at the red end of the polyline or click on the *Add* with green dot button to activate the function that add vertices at the green end of the polyline. To create a new vertex, click on the location of the new vertex. Right-click on the image to finalize adding vertices. Figure D.19 shows an example of adding vertices.

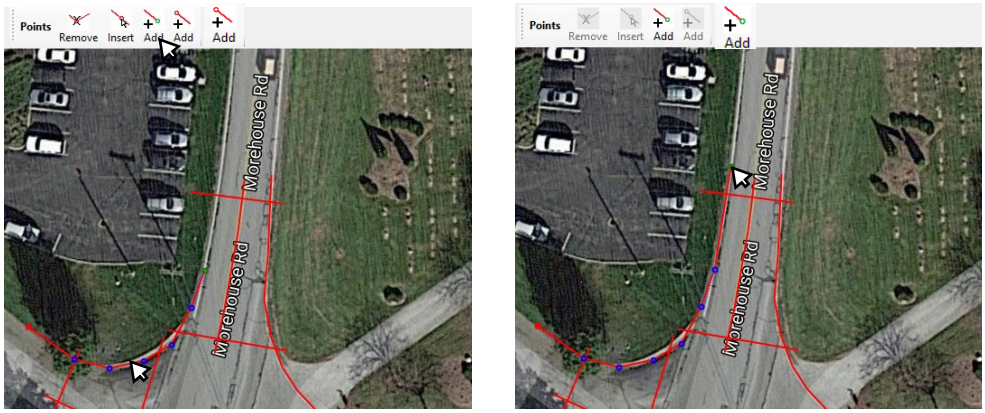


Figure D.19 Adding vertices.

Remove Vertices

To remove vertices, select the polyline by clicking on it to highlight its vertices. Click on the R*emove* button to activate the function. Click on the vertex to be removed from the selected polyline. Right-click on the image to finalize removing vertices. Figure D.20 shows an example of removing vertices.

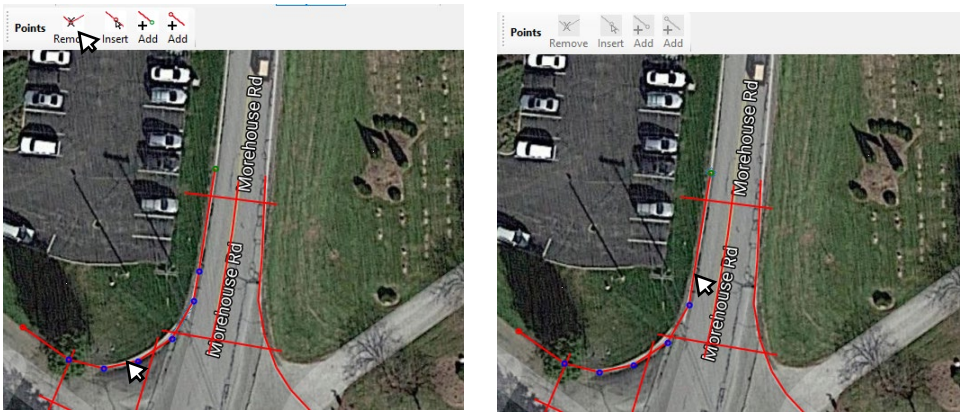


Figure D.20 Removing vertices.

Connecting Polylines

Use this function to trim all segments of the polylines and to add vertices at the intersection of any segment pairs. This command also removes all unconnected segments. Figure D.21 shows the result of the *Connect* function. The *Connect* polyline's function facilitates the process of drawing the intersection layout by allowing drawing boundaries of the intersection and lanes without paying attention to corners and then integrating and trimming redundant lines. The obtained intersection areas have limits and corners that meet the requirements needed to track objects inside the intersection effectively.

Before using this function, make sure that all polylines have at least two intersections with other polylines. Polylines that do not have at least two intersections will be deleted as a result of this process.

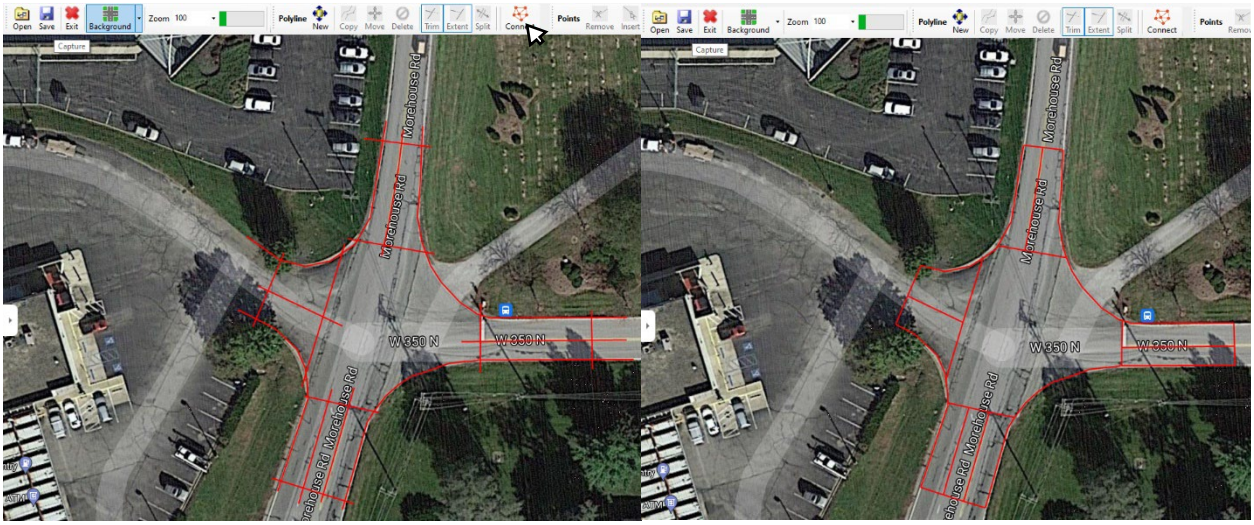


Figure D.21 Connect function before/after.

D.3.5.3 Creating polygons

Once the intersection layout is drawn, the next step is to create the polygons and set their characteristics. The polygon commands provide an easy way to convert polylines into closed polygons by executing several polyline operations at once.

Define Polygons

Use the *Define* command to select a polygon and to define its properties. After activating the *Define* command, the polygon definition bar appears (Figure D.22). Commands in this bar allows defining or redefining the function of the selected polygon. The definition includes type of polygon: approach, exit, intersection, parking, median and sidewalk. An approach polygon has associated traffic maneuver allowed from the polygon (typically a single lane): left, through or right. An exit polygon includes lanes after crossing the intersection (central area). A parking polygon is a lane used for curb parking. Sidewalk polygon is reserved for pedestrians and possibly for bicycles. An intersection polygon is the common area where vehicles leaving the approach polygons cross each other paths and leave this polygon by entering the exit polygons.

Once the command is activated, click on the image inside the polygon. The polygon area is highlighted. If the polygon already exists, the polygon information is displayed, otherwise, it has just been labeled by clicking on it is ready to be defined by entering its properties. Select one of the polygon definition command options to assign the function to the polygon.

Other polygon characteristics, such as zone ID, description, and lane number, if applicable, can be entered in the right-hand pane of the interface window (Figure D.22).

The polygons can be selected eighter by clicking in the image inside the polygon or by selecting it from the pulldown list at the top of the right-side panel. Note that only polygons selected at least once by the user are listed in this panel. Right-click on the image to end defining polygons.

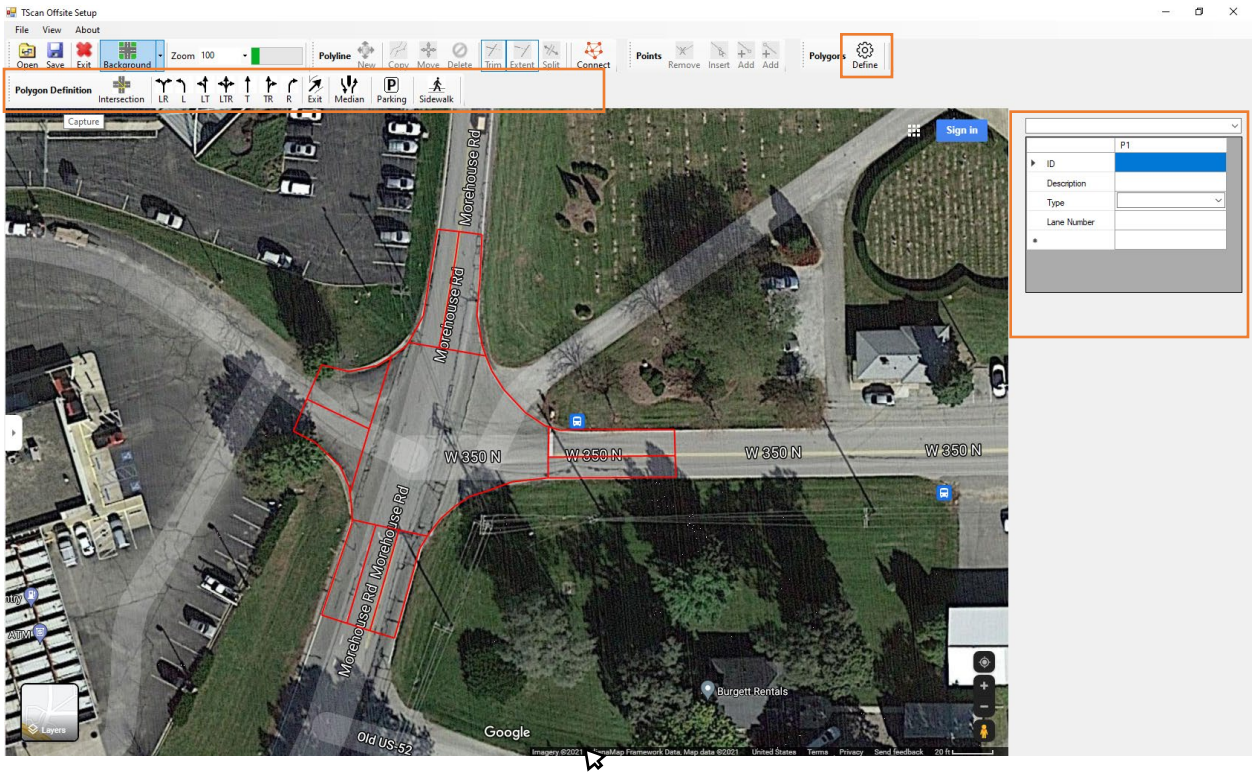


Figure D.22 Defining polygons.

D.3.5.4 Saving and retrieving layouts

TScan Offsite Setup allows saving the orthographic-image of the intersection as well as a ".dcs" file with the polygons layout and properties that are used during the on-site setup of the TScan system. Table D.2 shows the list of files produced and saved.

Table D.2 Files produced with the *TScan Offsite Setup*

File	Format	Description
<Orthographic-image>	Any image format such as bmp, jpg, jpeg, gif, png, tif, etc.	User can capture this image from the online mapping options or can provide a top view image of the intersection saved in image format. This 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.

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.23).

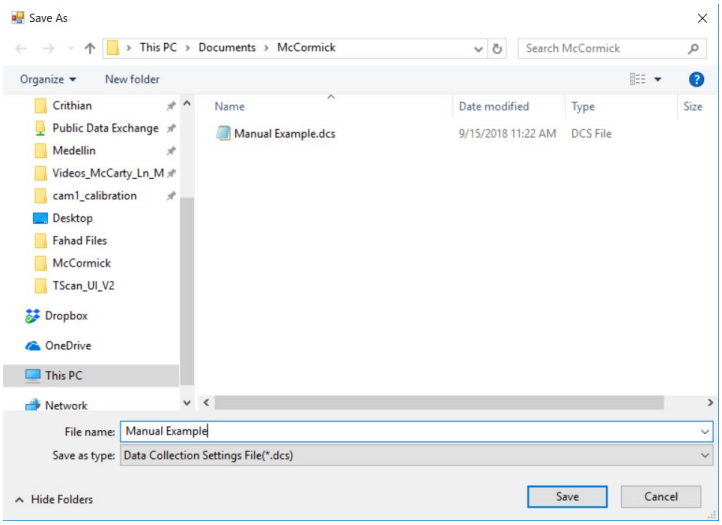


Figure D.23 Save layout.

Retrieving Layouts

After clicking on *Open*, the user is asked to select the file to be opened as shown in Figure D.24. 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.

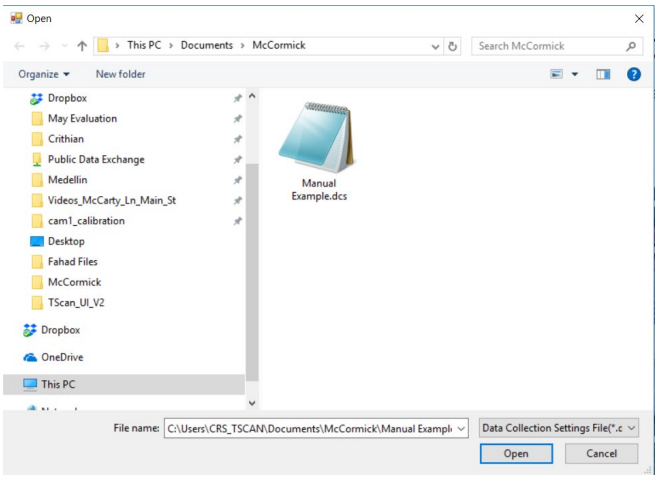


Figure D.24 Open layout.

D.3.6 Offsite Setup Final Remarks

The .dcs file generated by the *TScan Offsite Setup* program contains all the user information regarding the orthographic-image, the polygons information, and the associated coordinates. Please keep and take this file with you for the field data collection. This information is required in the next step of the TScan setup. The aforementioned step includes setting a common reference system for the TScan sensor and the orthographic-image in order to transfer the polygon characteristics to the TScan processing module. This requires an appropriate setup of the TScan hardware in the field.

D.4 TScan On-Site Setup

The *TScan On-site Setup* software assists the user in preparing the site information needed to set up the system in the field. The *TScan On-site Setup* program reads the files prepared in the Off-site setup which includes both the orthoimage and the intersection layout and characteristics. This file is needed to establish a single spatial reference shared by the TScan sensors and the ortho image of the intersection to relate the collected data to the intersection and GIS coordinates.

Since the TScan Head computer has no monitor or keyboard, the computer and program are controlled via remote access using the trailer's ethernet connection.

After the setup information is obtained, the Data Collection and Processing module receives the data from the sensors, processes it, and produces an output file. No user intervention is required while collecting data unless there is an indication of a malfunction.

The output file produced as the result of data collection and on-site processing is useful for many engineering studies.

D.4.1 Remote Desktop

After setting up the TScan hardware, connect the user laptop to the one located on the TScan head. To do so, connect the laptop to the WiFi Network TScan with password TScan. Once the laptop is connected to the specified network, launch the Remote Desktop Connection program in the laptop. This program can be launched by following these steps.

1. Press start button.
2. Start typing *Remote Desktop Connection*, the program shortcut should appear in the search results.
3. Single click in the shortcut.

In the dialog box of the program type *TScanMaster* under the Computer Tab. Once this computer is selected click connect. Enter TScan as the user credentials. When the connection is established you will be able to visualize the desktop of the computer located on the TScan head and the main window of the *TScan On-site Setup* software.

D.4.2 Launching TScan On-Site Setup

TScan On-site Setup software helps the user prepare site information needed to set the system in the field. The software is already installed in the TScan Head computer and is launched automatically when the system is powered up.

In case the *TScan On-site Setup* program is not running, it can be launched using any of the following methods.

Method 1: Double click the shortcut on the desktop.

Method 2:

4. Press *Start* button.
5. *TScanOnSite* should appear in the list of installed programs.
6. Single click on the shortcut.

Method 3:

4. Press *Start* button.
5. Start typing *TScan*, the program shortcut should appear in the search results.
6. Single click on the shortcut.

Method 4:

4. Open *My Computer.*
5. Browse to the location where the software was installed. Typically *'C:\Program Files (x86)/ TScanOnSite /'.*
6. Click on *TScanOnSite.exe.*

D.4.3 On-Site Interface

Figure D.25 shows the main interface.

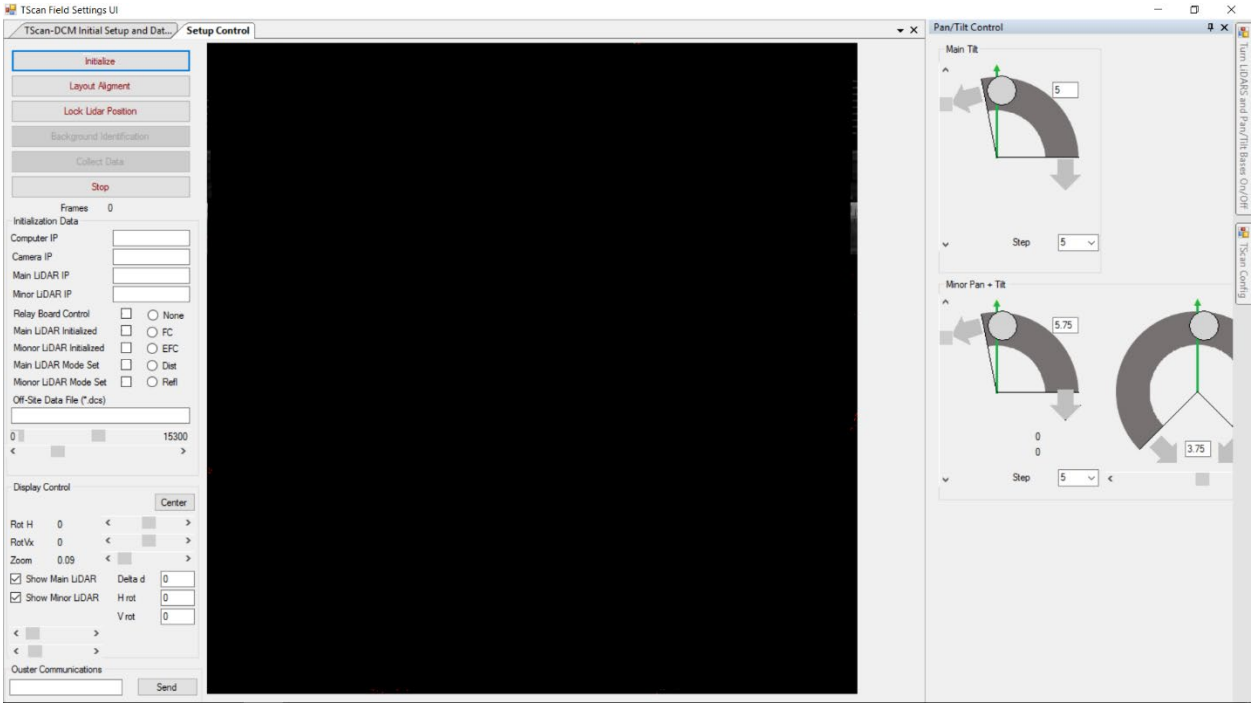


Figure D.25 TScan On-site Setup main interface.

The program includes several windows. The control and live display window is used to execute the different program functions and to display the sensor readings in order to ensure the intersection coverage; the orthoimage alignment window is used to overlay the orthoimage with the sensor readings; and the configuration window is used to indicate the ports for the different hardware elements in the TScan head.

D.4.4 Sensor Initialization

The first step is to initialize the sensors. To do so, click on the *Initialize* button to start the process (Figure D.26). This process can take few minutes. The IP addresses are displayed in the control panel as well as the checklist of the sensors that are initialized.

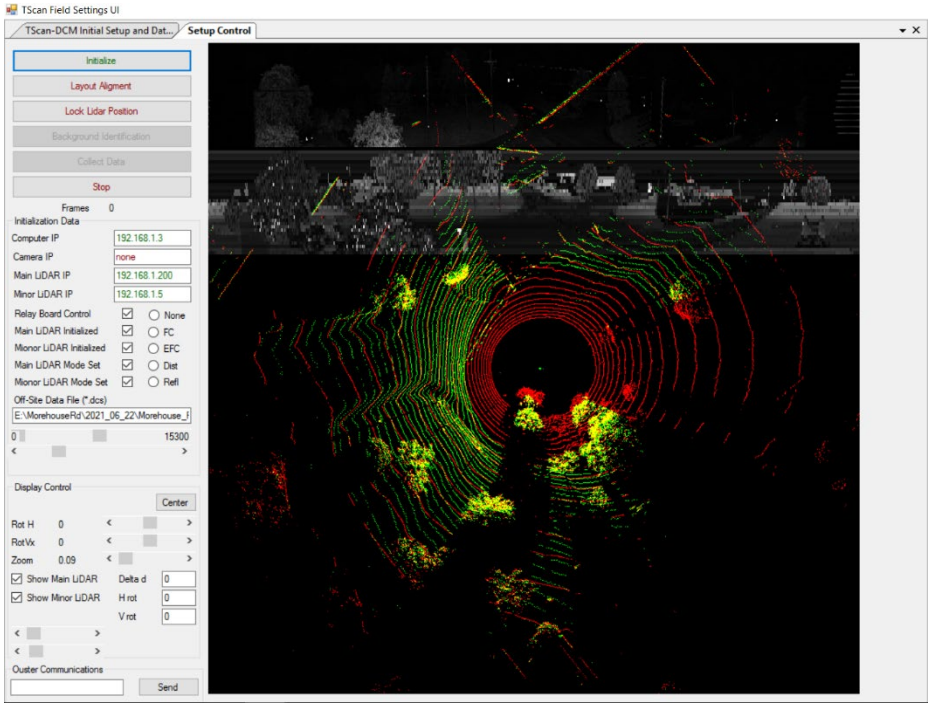


Figure D.26 TScan on-site setup control and live sensor readings display.

Once the sensors are ready, the program ask for the location of the .dcs files prepared during the off-site setup.

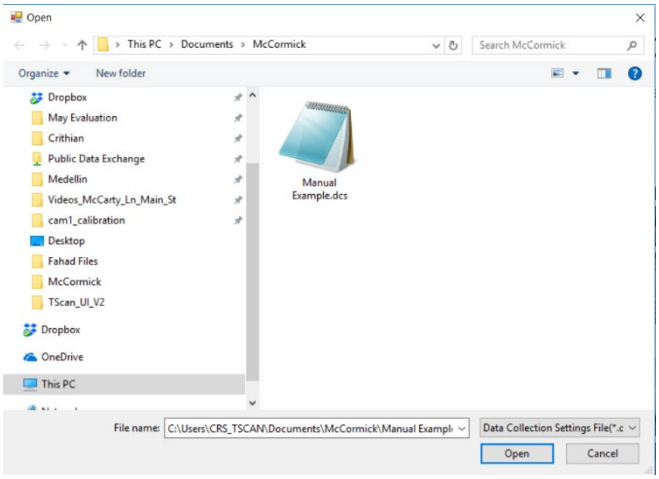


Figure D.27 Retreiving the off-site information.

Once the files are retrieved, the user can visualize the point cloud collected with the main sensor (located on top) in red color. The point cloud collected with the auxiliary sensor is shown in green color. A black and white image of the sensor's readings are also displayed.

D.4.4 Sensors Orientation

After initializing the sensors, the next task for setting up the system is to find an optimal position for the LiDAR sensors. The Pan/Tilt Control Tab allows the user to define the position of the main LiDAR (located on top of the TScan Head) and the Auxiliary or minor mechanism (located in the lower position). The user can control the pan/tilt mechanism using the scroll option or by defining as input the desired angle (Figure D.28). The reference zero is shown in green.

The appropriate position of the LiDAR involves maximizing the coverage at the intersection provided by the two sensors.

To maximize coverage, set up the mast and the head mechanism pointing at the center of the intersection. Tilt the main sensor that allows collecting points from vehicles from approximately 150 ft. The auxiliary LiDAR provide a pan and tilt mechanism in case the user requires a better detection of a specific approach of the intersection. If the user is interested in better detection on the intersection polygon, the pan mechanism should be positioned at the zero reference. Make sure to interlace the point cloud between the two sensors when selecting the position of the pan mechanism for the auxiliary sensor.

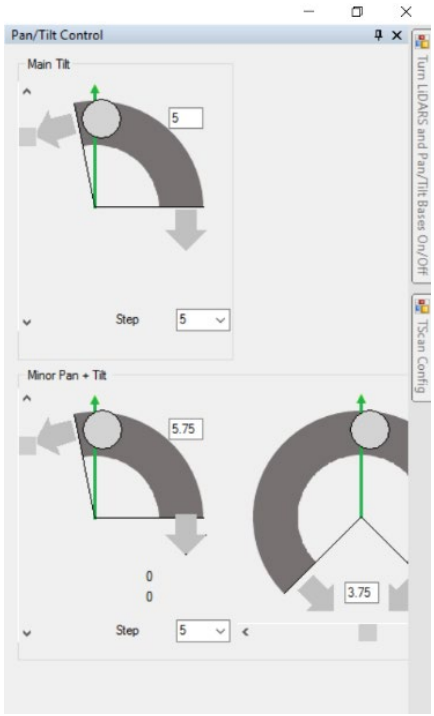


Figure D.28 Lidar orientation window.

D.4.5 Layout Alignment

The step 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 On-Site Setup. The initial alignment can be performed semi-automatically based on user defined reference points. Then fine manual adjustments can (also) be made.

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.29. 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 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.

If the .dcs file was not opened after initialization, then the .dcs file containing the information regarding the polygons needs to be opened. Press the button *open* and select the .dcs file corresponding to the layout of the intersection to analyze. Repeat this process to open the .jpg image using the button background.

D.4.5.1 Initial sensor data

Click on the *Overlay* button to display both the orthographic-image and few frames of the TScan sensors side by side along with a black and white image of the sensor's readings as shown in Figure D.29.

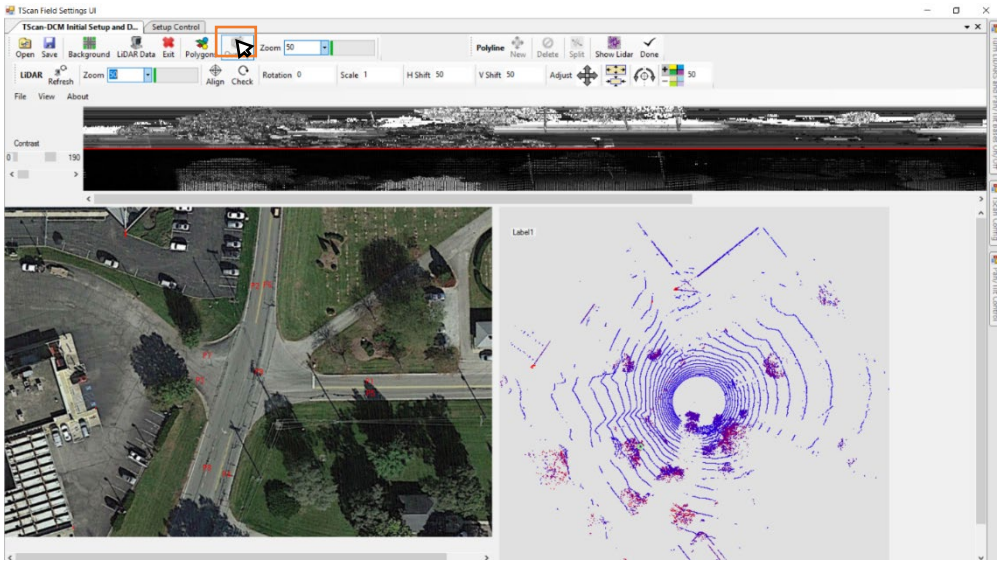


Figure D.29 Initial sensor data.

D.4.5.2 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 right-click on the image to finalize the creation of the reference points. 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.30.

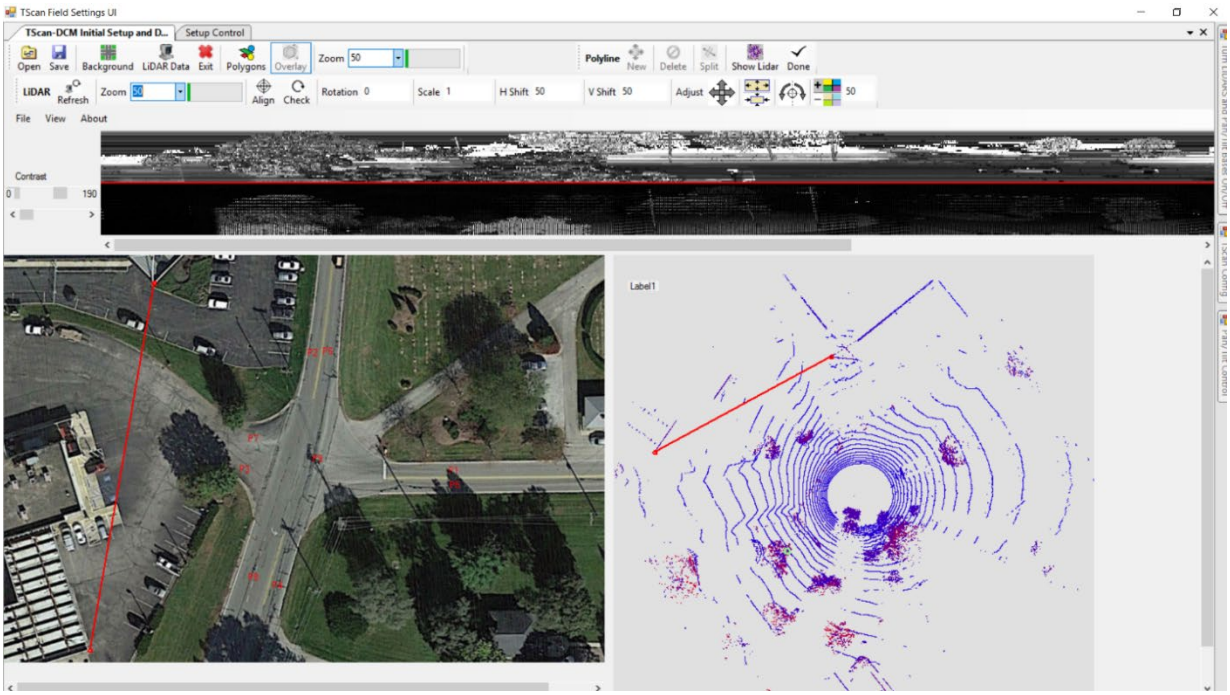


Figure D.30 Point correspondence to align TScan and orthographic-image.

Align

Once the reference points are created, the semi-automatic alignment is completed after clicking on Align button. As a result, the TScan data is overlayed on top of the orthographic-image as shown in Figure D.31. It also updates the alignment parameters for rotation, scale, and vertical and horizontal shift.

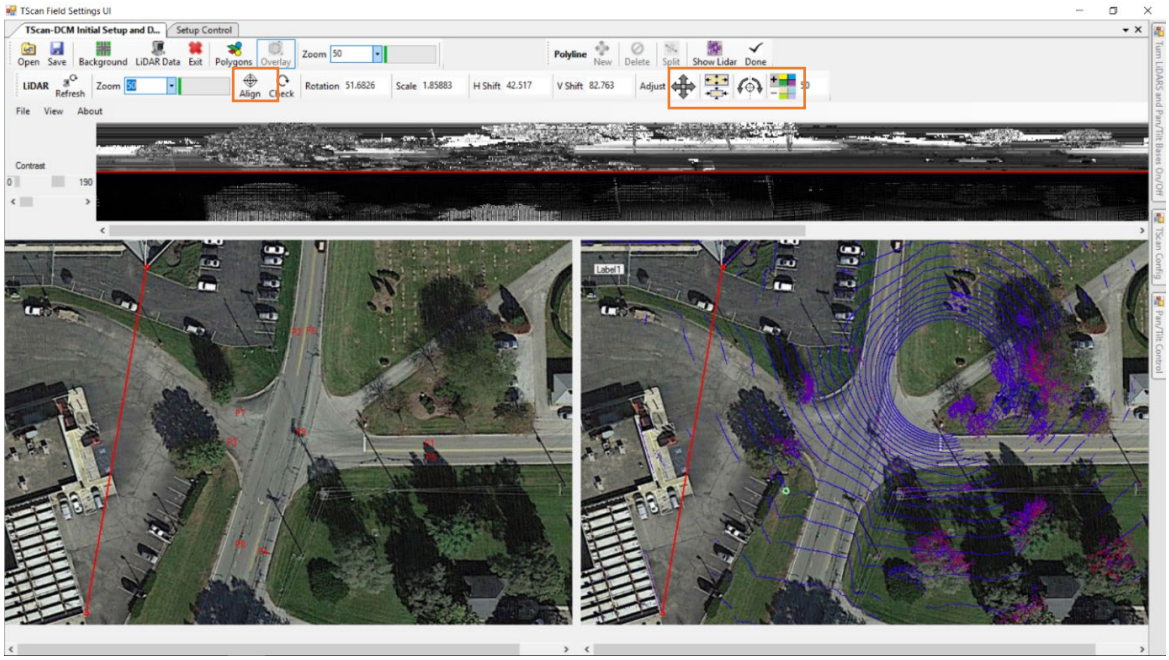


Figure D.31 TScan and orthographic-image alignment.

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 (Figure D.31). The overlay will be updated after clicking. 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 (Figure D.31). The new overlay will be updated after 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% 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 (Figure D.31). The overlay will be updated after clicking. Press and hold the left mouse button to repeat the operation five times per second and release it to stop.

D.4.5.3 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 (Figure D.31). The new overlay will be updated after clicking. 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.

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

D.4.5.4 exporting aligned polygons

Once the alignment is completed, return to the polygons view by clicking on *Polygons* (Figure D.32). Then click on the *Export* button to save the polygons in the TScan reference system. This process may take some time.

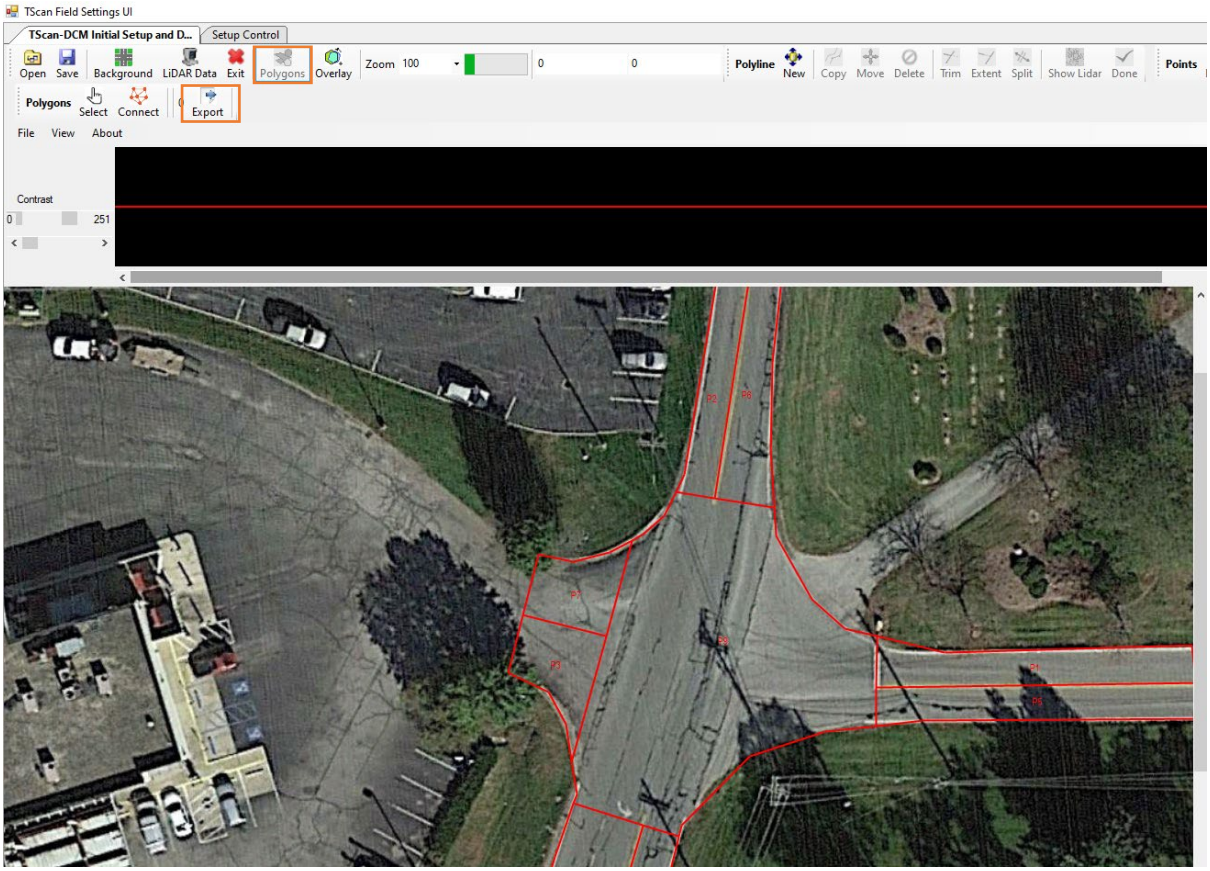


Figure D.32 Display polygons and export.

D.4.6 Background Identification

Select the control window tab to continue with the data collection process. The software indicates the correct transfer of the polygons by showing a green convection in the buttons. Click on Background Identification to remove the background and isolate only the points belonging to objects. This process takes approximately 20 minutes. The progress bar informs the user when the background is properly identified.

D.4.7 Data Collection

Click on *Collect Data* to launch the data collection process that 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.33.

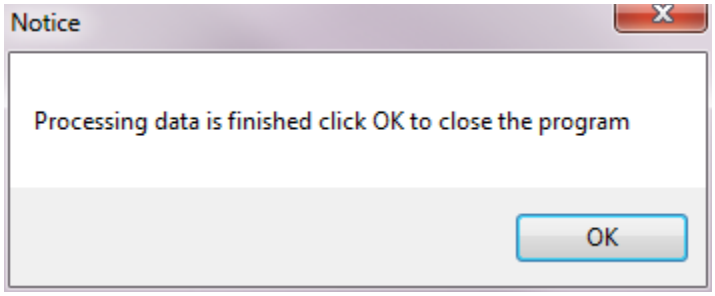


Figure D.33 End calculations.

D.4.8 Configuration Window

Each TScan head is preconfigured when the On-Site Setup program was installed. The configuration panel displays the hardware Ids and the communication port of the main components of the TScan head including the following.

- Main tilt.
- Pan minor.
- Tilt minor.
- Inertial measurement unit.
- Power relay controller.

APPENDIX E. TSCAN ENGINEERING APPLICATIONS

E.1 Preface

The TScan project aims to develop a data processing module for a novel LiDAR-based traffic scanner to collect highly accurate microscopic traffic data at road intersections. TScan uses Light Detection and Ranging (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 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 errors of estimation dependent only on the resolution and accuracy of the LiDAR sensor.

The Engineering Application Program include Trajectory Visualizer, Counting Vehicles, Display Pedestrian Occupancy, Evaluate Vehicle's Speed in the Intersection area, and Evaluate Traffic Conflicts. This toolset was developed by the Center for Road Safety. The User Manual includes the user interface to upload information from both the initial setup and the data collection, to enter the required information for the different types of analysis and for displaying trajectory files generated at a given intersection. The TScan Engineering Application toolset was developed as part of the Stationary LiDAR for Traffic and Safety Applications–Vehicles Interpretation and Tracking (TScan) project at the request of the Indiana Department of Transportation, and its development was supported through the Joint Transportation Research Program of Purdue University and the Indiana Department of Transportation.

E.2 Tscan Engineering Applications Overview

TScan Engineering applications are some applications developed to use TScan results. Figure E.1 shows TScan framework in which TScan Engineering Applications is included.

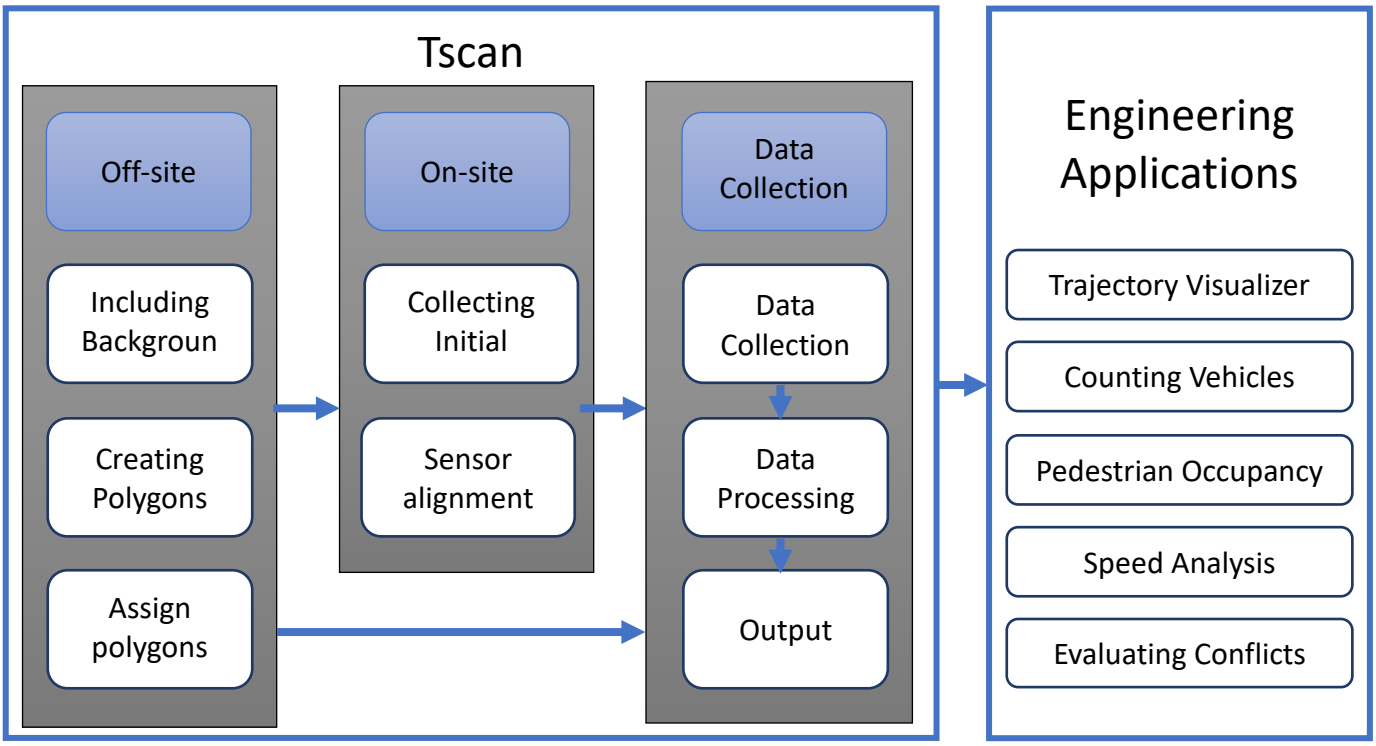


Figure E.1 TScan overview.

TScan Engineering Applications toolset is a computer application that allows the user to display vehicle's trajectories, counting turning movements, display pedestrian occupancy, evaluate vehicle's speeds in the intersection area, and evaluate traffic conflicts based on TScan data collection output.

E.3 Installation

TScan Engineering Applications is compatible with Windows 10. In order to run TScan Engineering Applications, the MS .NET 4.5.2 Framework or later component must be installed.

If the MS .NET 4.5.2 Framework is not present during the installation, TScan Engineering Applications will attempt to install this component if the PC is connected to the Internet.

To install the TScan Engineering Applications interface, follow the steps given below.

3. Extract the contents of the archived file to your local drive.
4. The installation process is initiated by clicking on the *setup.exe* file.
5. 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.4 Launching TScan Engineering Applications

The TScan Engineering Applications program can be launched using any of the following methods after installation.

Method 1:

1. Double click the shortcut on the desktop.

Method 2:

7. Press *Start* button.
8. TScan Engineering Applications should appear in the list of installed programs.
9. Single click on the shortcut.

Method 3:

7. Press *Start* button.
8. Start typing "*TScan Engineering Applications*," the program shortcut should appear in the search results.
9. Single click on the shortcut.

Method 4:

7. Open *My Computer.*
8. Browse to the location where the software was installed. Typically, *'C:\Program Files (x86)/TScan Engineering Applications /'.*
9. Click on *TScan Engineering Applications.exe.*

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

E.5 TScan Engineering Applications Interface

The TScan Engineering Applications interface includes a command bar in the first row and a tab control below (Figure E.2). The command bar facilitates operations on files, background, visualization, and window layout. The Tab control allows user to select among the different applications. 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.

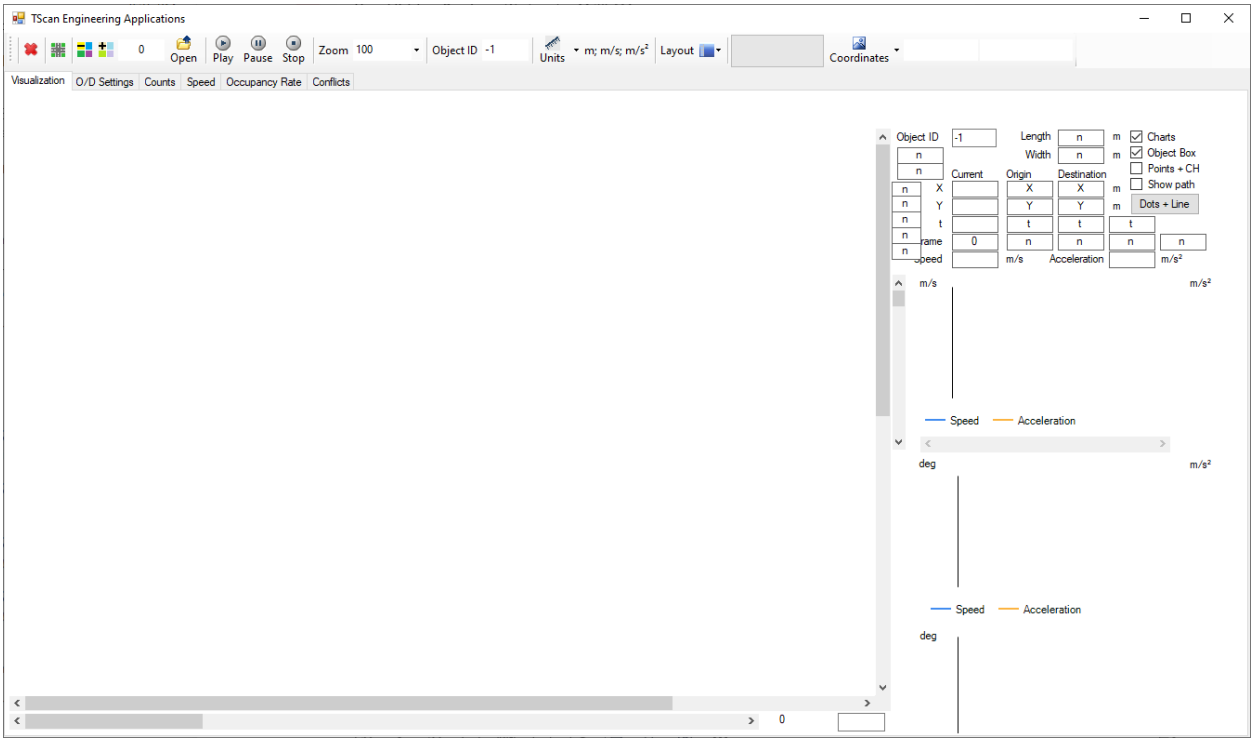


Figure E.2 TScan engineering applications main interface.

The *Visualization* tab allow user to display the vehicle's trajectories and inspect their parameters such as size, location, speed, acceleration. The *O–D Settings* tab allow user to create vehicle's movements groups, input the origin-destination matrix for directional counting and to set the counting interval. The *Counts* tab and the *Speed* tab allows users to display and save the results. The Occupancy Rate Tab is used to graphically represent the pedestrian occupancy rates. Finally, the Conflicts Tab allow user to enter the filter parameters and display the conflict results. The following chapters describe each activity in detail.

E.5.1 Upload Data

Even TScan generates a set of files that are required for the engineering applications, all files are managed by the collection settings file <*Location Name*>.*dsc*. All files should be kept in the same folder to maintain the data integrity.

To open the data collection settings file obtained after post processing the data, click on the *Open* button to open the Windows file selection window shown in Figure E.3, where the appropriate file can be selected. Once selected click on the *Open* button to upload the image and the corresponding data. The orthographic-image is uploaded automatically since its location is saved on the dcs file.

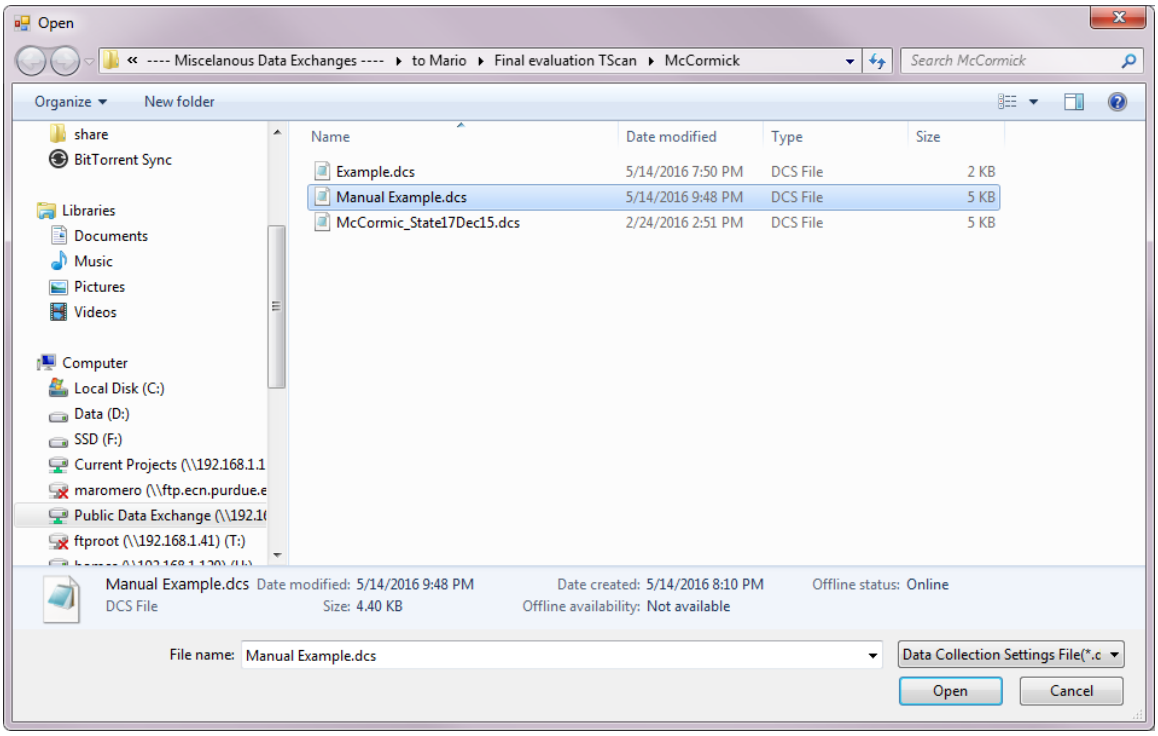


Figure E.3 Upload data collection settings file.

E.6 Trajectory Visualizer

The Trajectory Visualizer allows the user to graphically inspect the TScan tracking results frame by frame or by selecting a vehicle for detail inspection. The intersection layout is shown, along with the moving objects (vehicles, bicycles, or pedestrians) which are represented as a blue rectangle (Figure E.4).

The visualization is done frame by frame thus can be presented as a video using the play, pause or stop buttons (1). A particular frame can be selected either by using the scroll bar at the bottom or by typing the desired frame number (2). Also, the zoom level of the image can be adjusted by selecting it in the dropdown list (3), as well as the distance, speed, and acceleration units (4). Finally, a particular vehicle can be selected for a more detailed analysis by clicking on the vehicle's box or by typing in the vehicle ID (5) (Figure E.4).

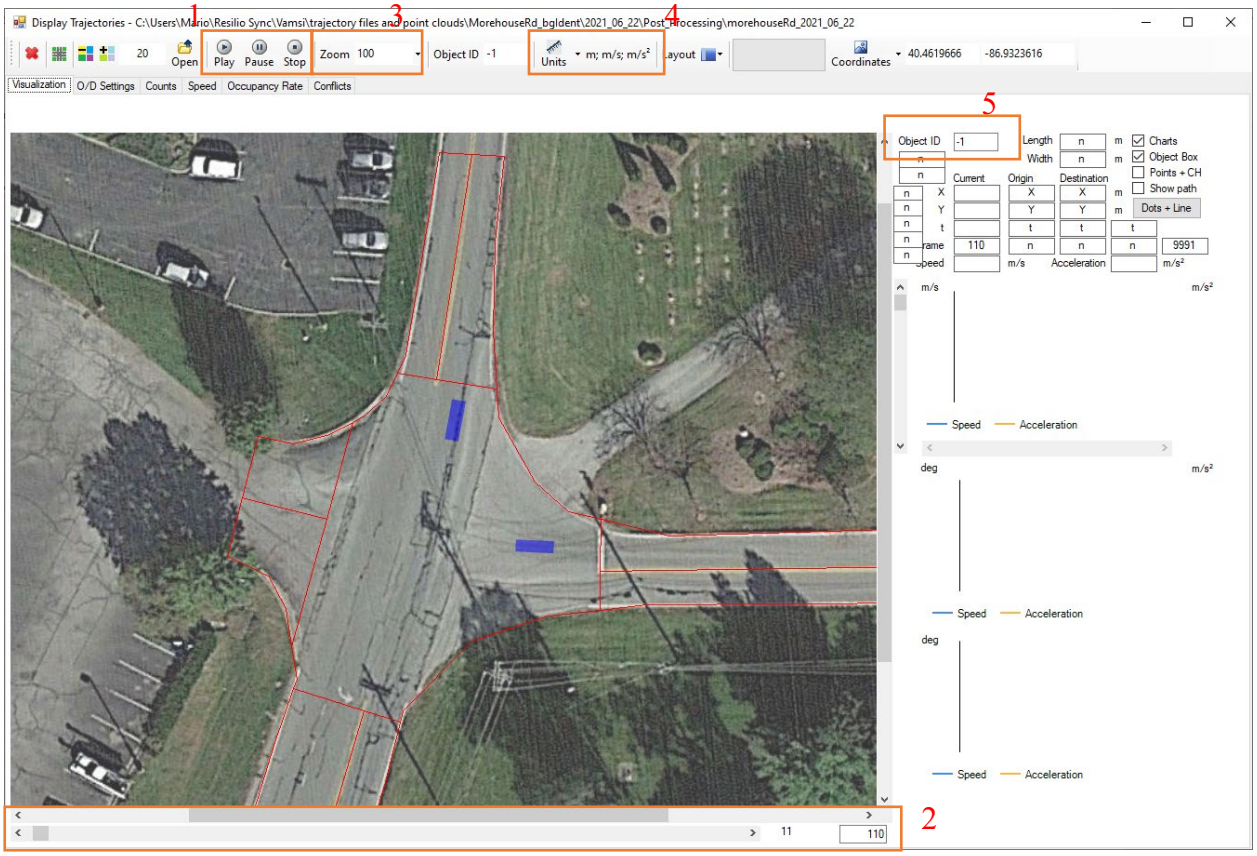


Figure E.4 Visualization tab.

Once a vehicle is selected, its location, speed, and acceleration in the current frame is displayed (1) as well as a chart that represents its entire trajectory (2). The representation of the objects can be changed to show the lidar points that compose it, its bounding box, and its path by selecting the corresponding checkboxes (3) (Figure E.5).

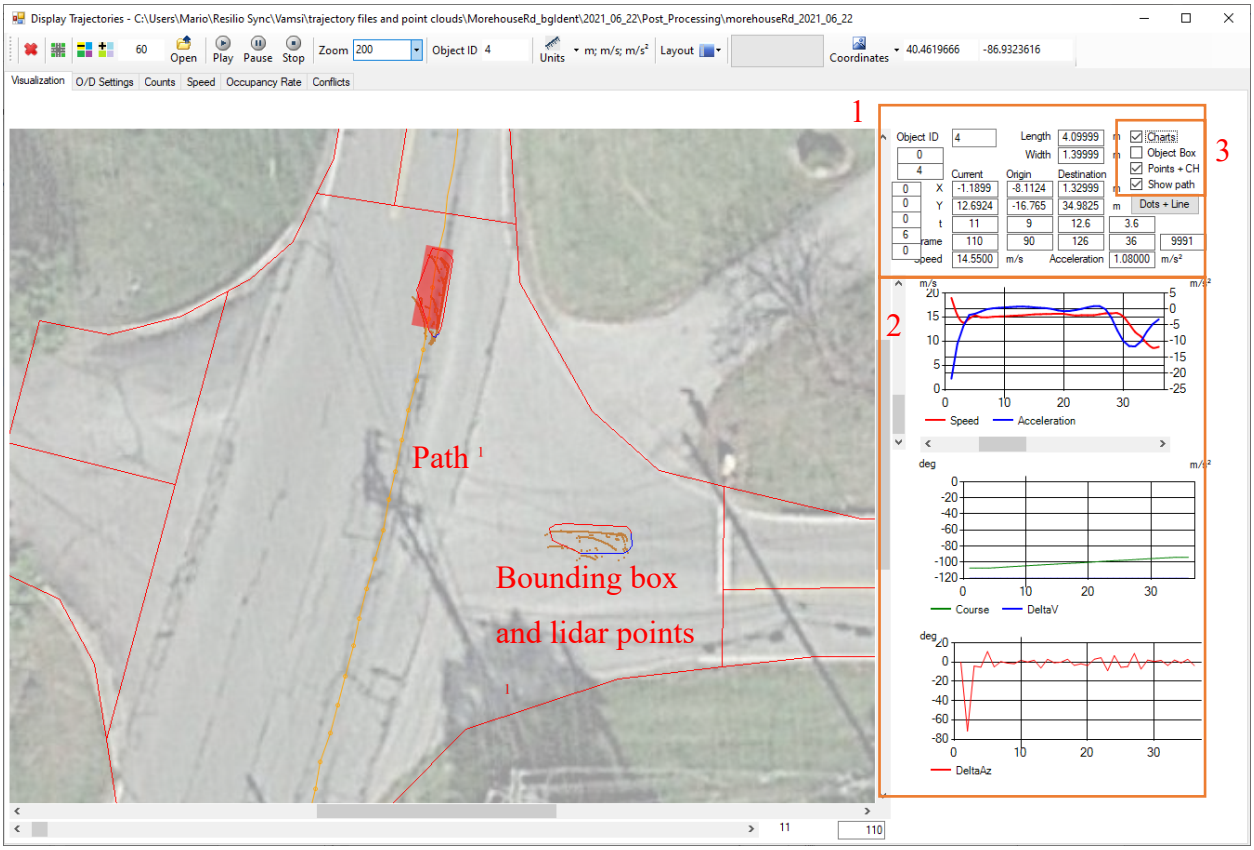


Figure E.5 Vehicle inspection.

E.7 Movements Setup

Both the Counting Vehicles and the Speed Analysis Module requires the user to review the uploaded data, select the time interval, set the starting time, and create the origin-destination matrix to define the movements, based on the created reference polygons of the intersection.

Each of the steps are explained in further detail below.

E.7.1 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.

E.7.2 Set the Starting Time

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

Vehicles entering the intersection polygon before the starting time are not considered in the counts or in the speed analysis.

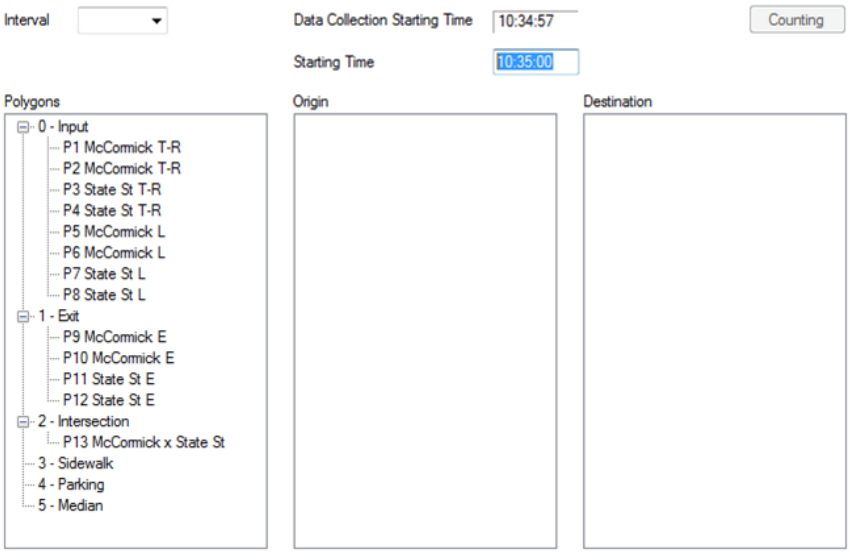


Figure E.6 Setting the starting time.

E.7.3 Select Time Interval

The selected time interval governs both the counting and the speed analysis aggregation level. Select the 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's time period is not a multiple of the selected interval, then the last portion is ignored.

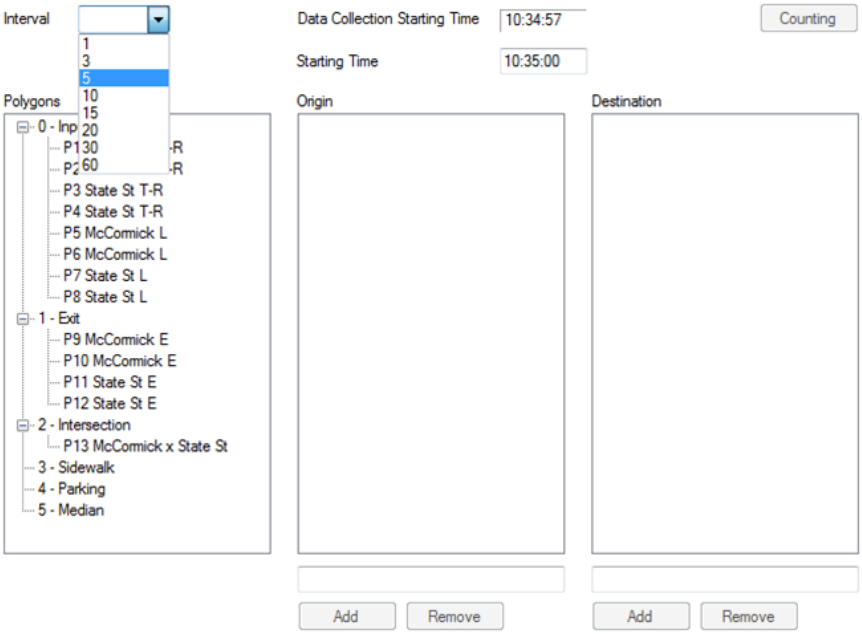


Figure E.7 Interval selection.

E.7.4 Creating Origin-Destination Matrix

To count turning movements it is necessary to set the origin 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.

E.7.4.1 Creating origin approaches

To create an origin approach. Select the current origin textbox. Enter the origin name and click on *Add* button. This action will create an origin place holder and selects it as shown in Figure E.8. Then add the corresponding polygons to the selected origin.

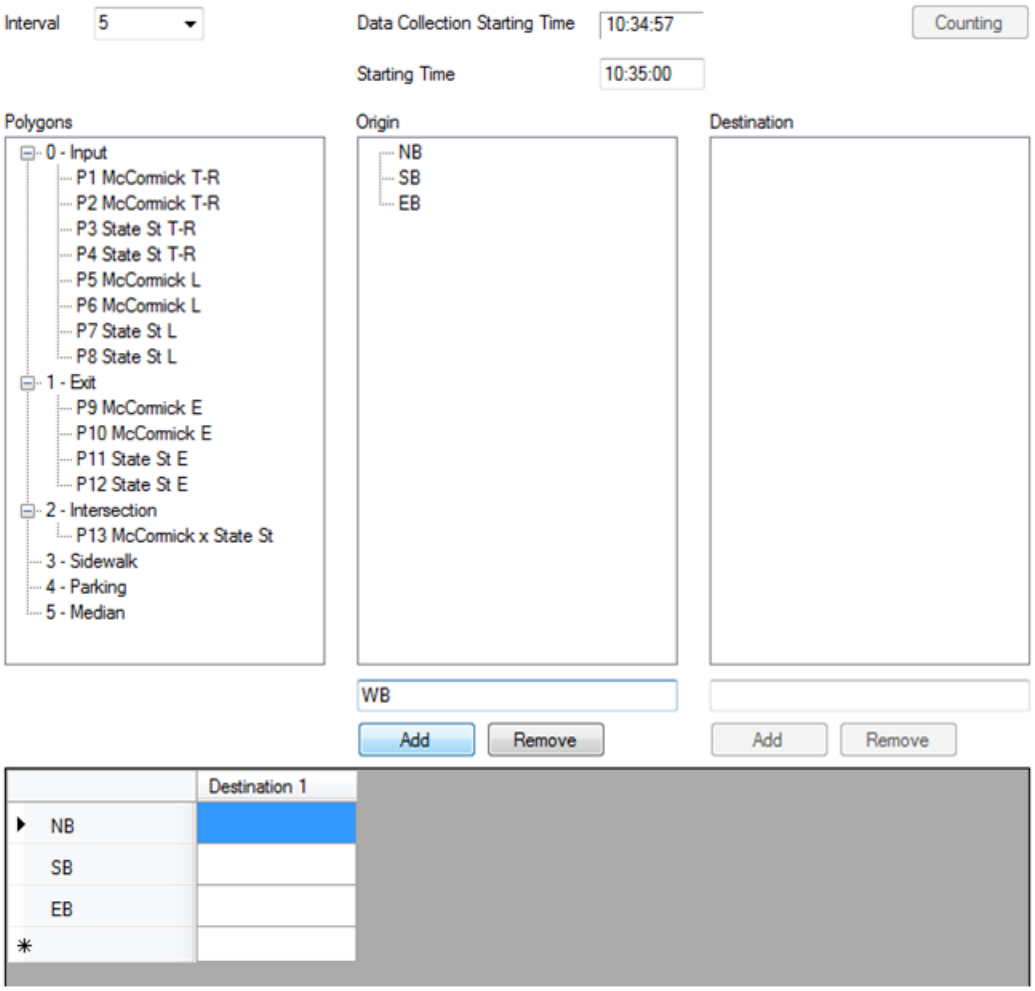


Figure E.8 Creating origin approaches.

E.7.4.2 Creating a destination area

To create a destination area. Select the current destination textbox. Enter the destination name and click on *Add* button. This action will create a destination area place holder and it becomes the selected destination as shown in. Figure E.9. Then add the corresponding polygons to the selected destination.

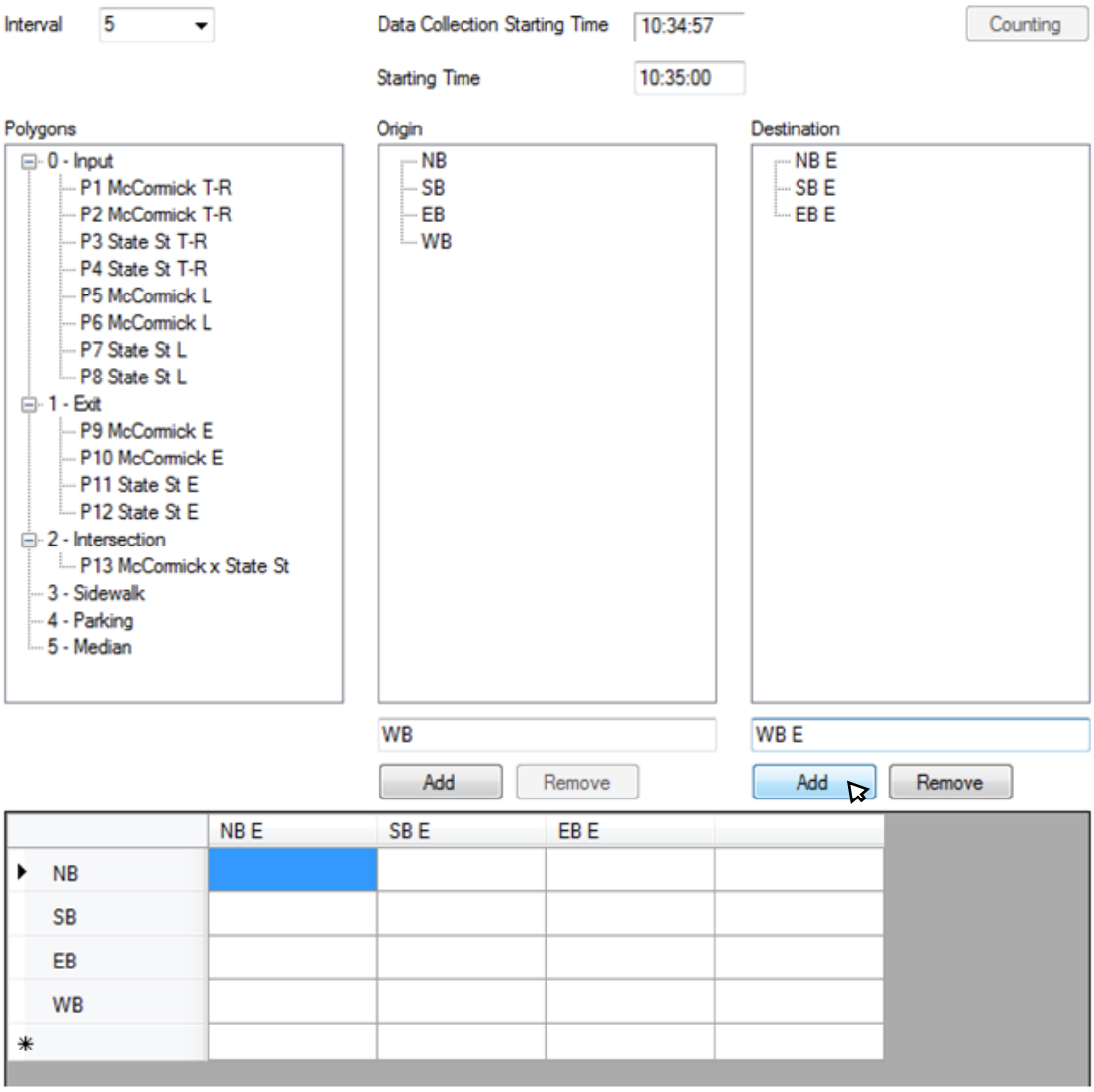


Figure E.9 Creating destination areas.

E.7.4.3 Assigning polygons to an origin approach or to a destination area

To assign 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.

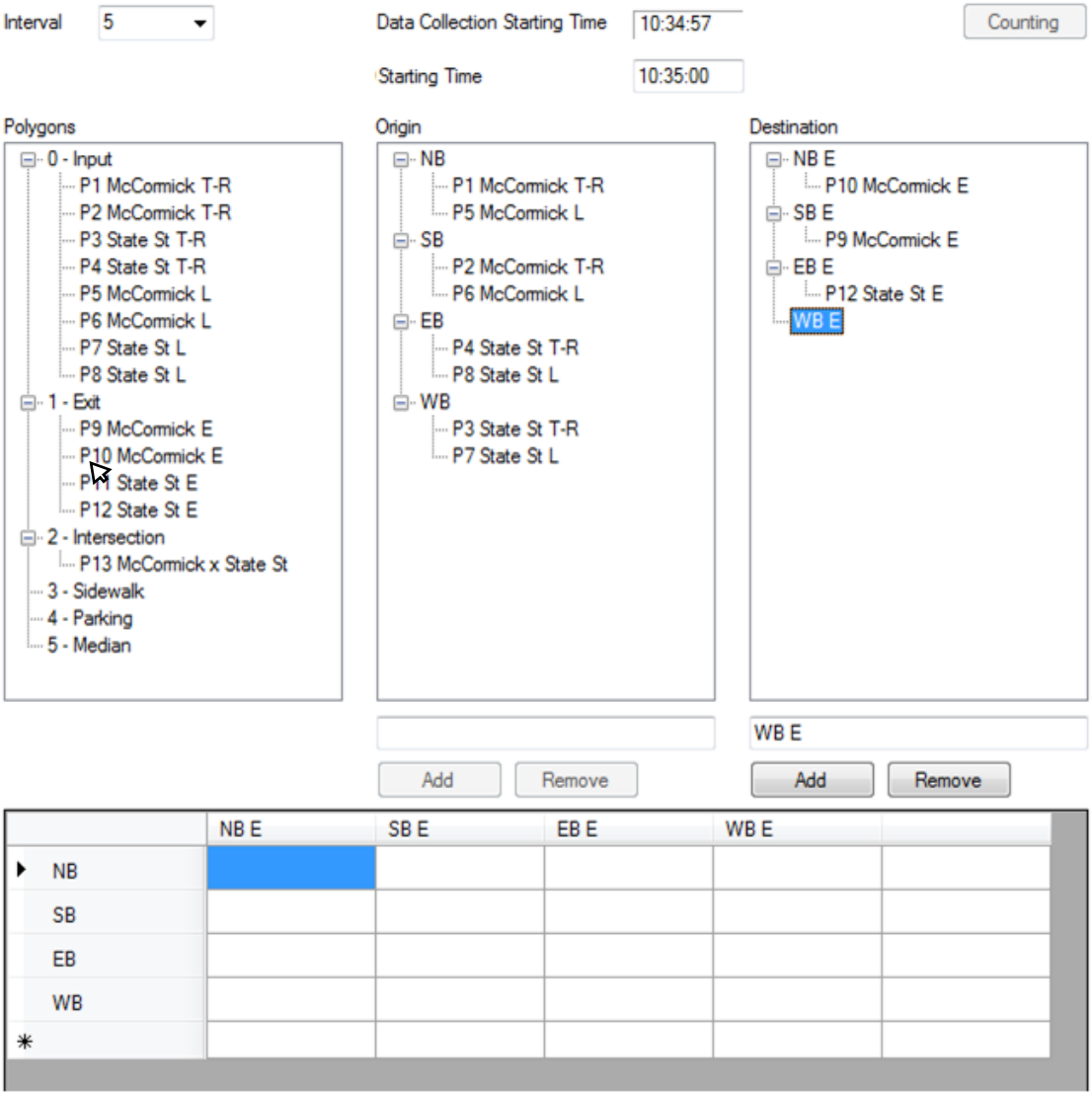


Figure E.10 Assigning polygons to an origin approach or to a destination area.

E.7.4.4 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.

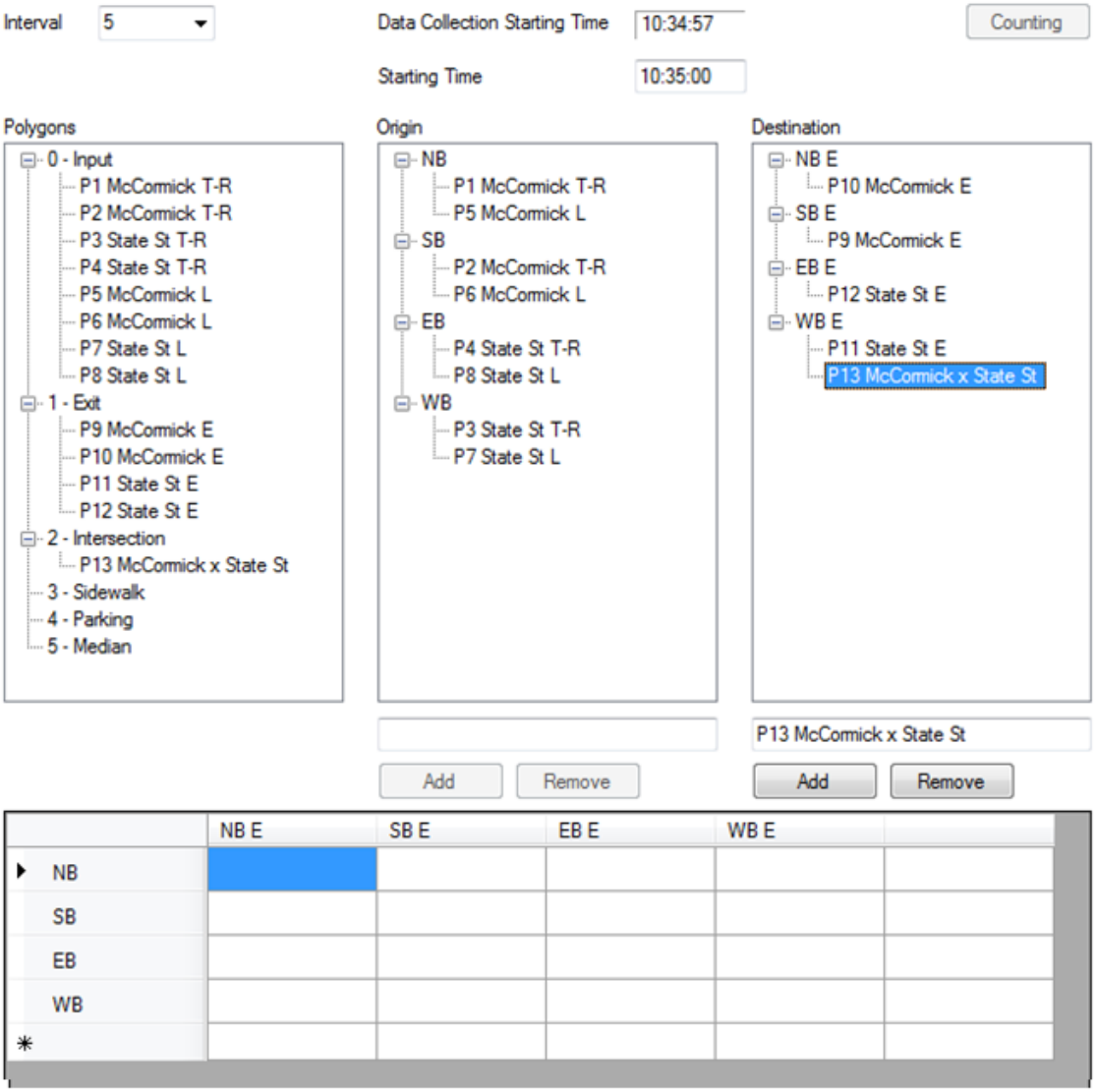


Figure E.11 Removing polygons from an origin approach or a destination area.

E.7.4.5 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.

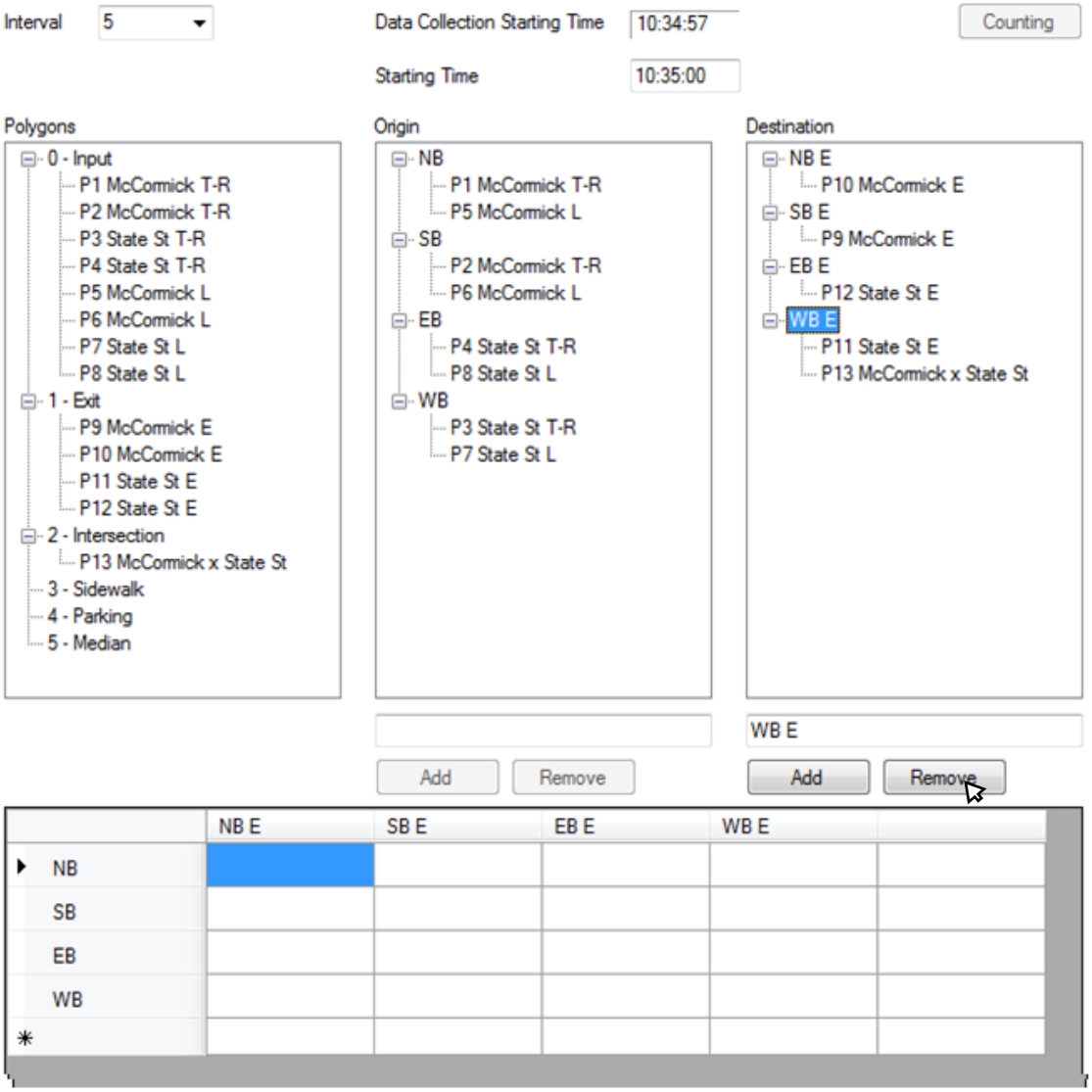


Figure E.12 Removing an origin approach or a destination area.

E.7.4.6 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.

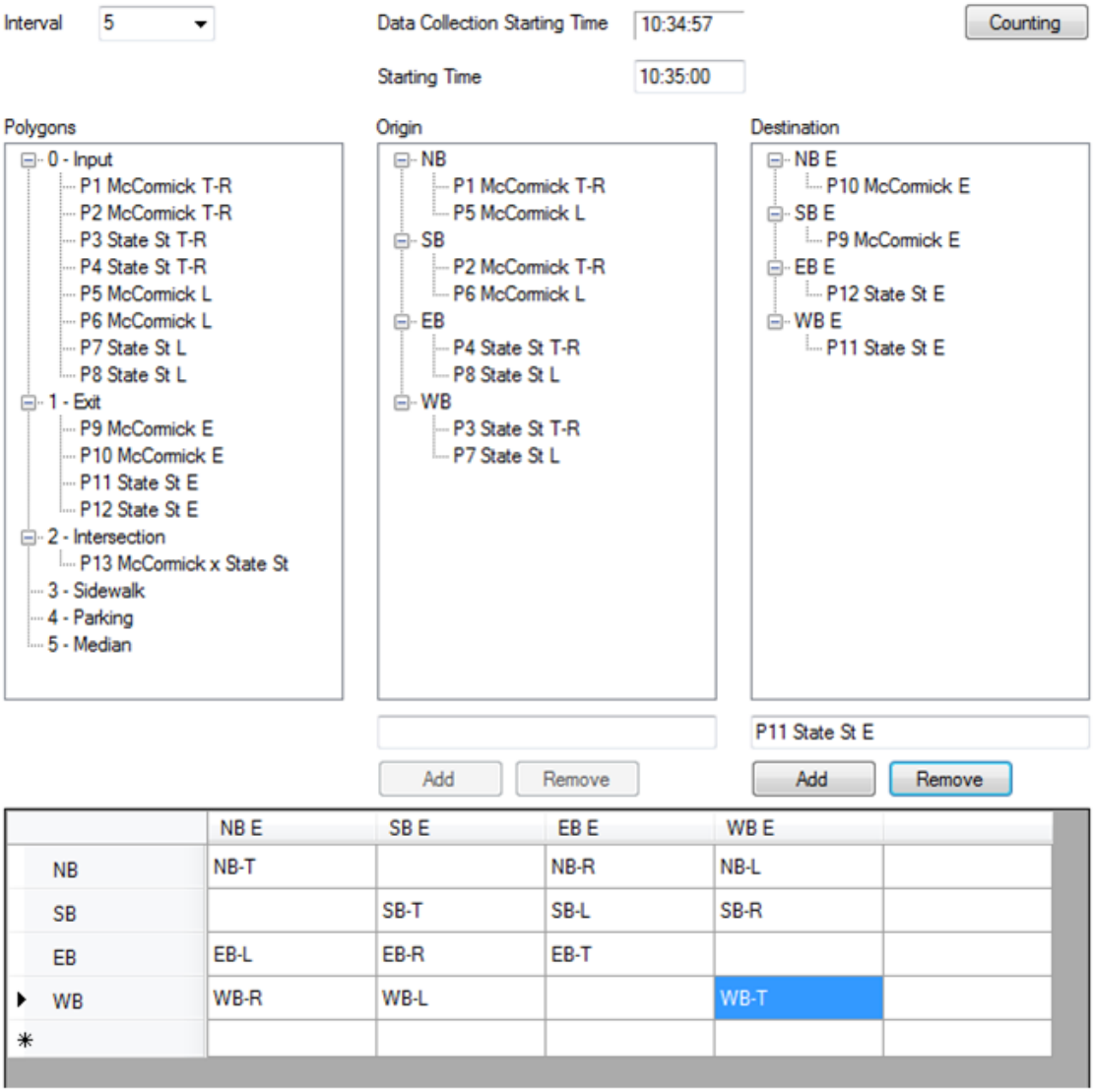


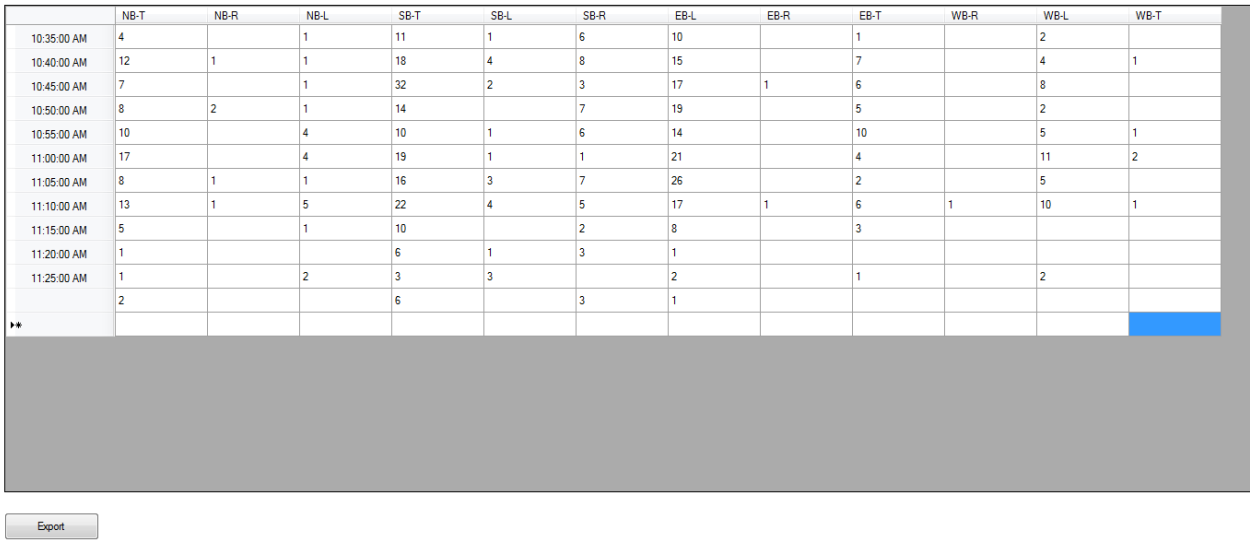
Figure E.13 Setting movement names.

E.8 Counting Vehicles

The Counting Vehicles Module allow the user to obtain directional counting statistics based on the TScan post processing output files using user defined counting periods. 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.

E.8.1 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.



	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					
▸*												

Export

Figure E.14 Counting results.

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

E.8.2 Export Results

To save the shown result table into a comma separated values file format, click on the *Export* button. 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.

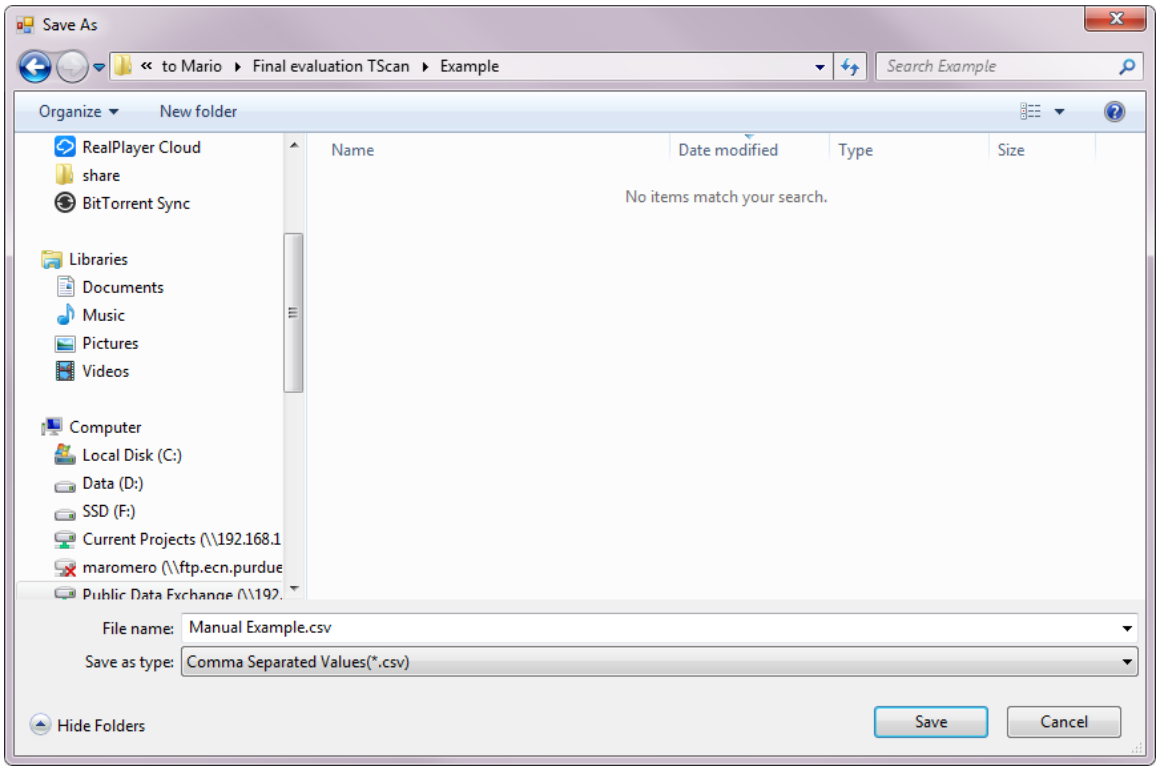


Figure E.15 Export results.

E.9 Pedestrian Occupancy Rate

The pedestrian occupancy rate graphically presents the time a "tile" is occupied by a pedestrian during a given period. The more intense the color, the higher the occupancy rate (Figure E.16).

To define the aggregation interval, select from a pulldown list for 1, 5, 15, 30 or 60 minutes (1).

With the sliders at the top (2), the color saturation level can be controlled depending on the occupancy time. The upper bar determines the point at which the color is fully saturated while the lower bar represents the point at which the saturation is set to zero.

Finally, it can be defined whether the color saturation variation is linear, logarithmic, or a combination of both (3).

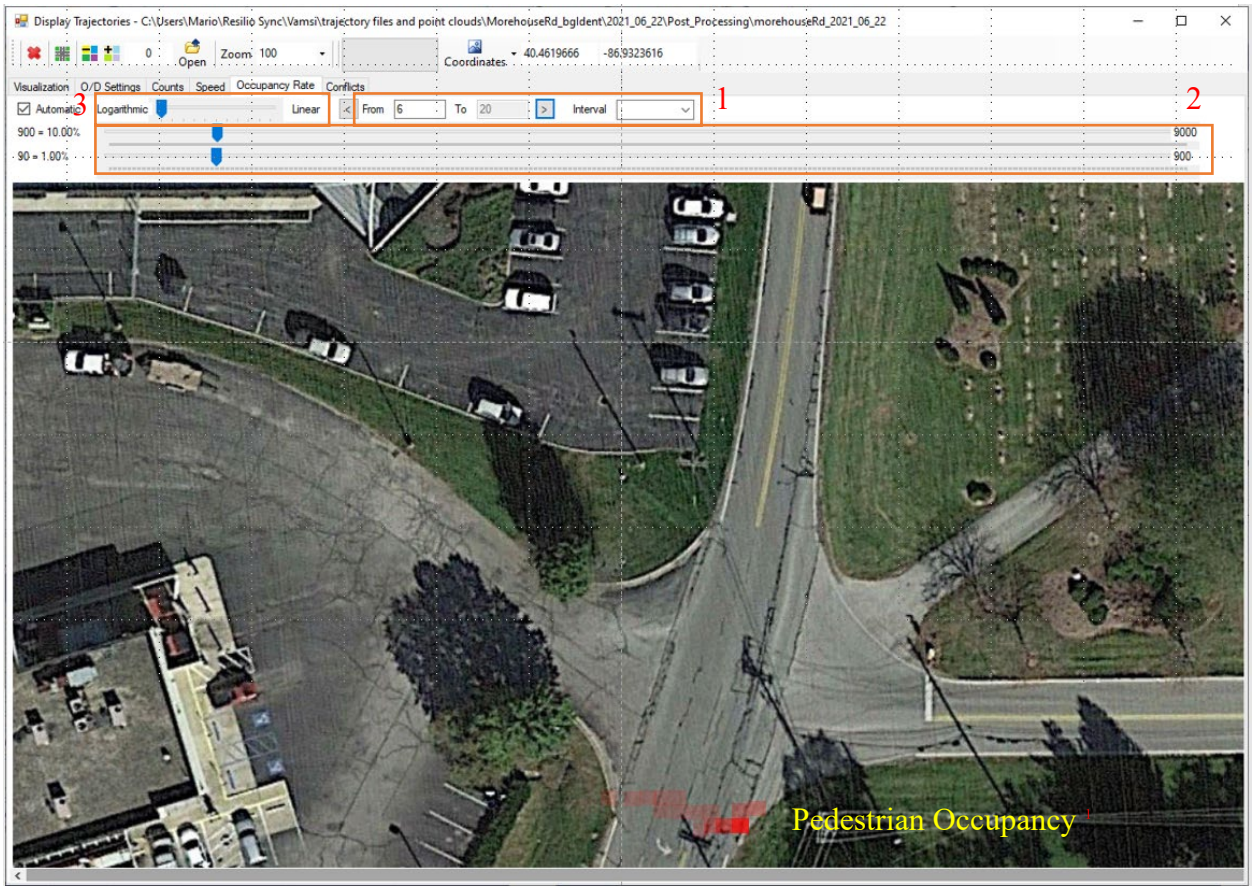


Figure E.16 Pedestrian occupancy rate.

E.10 Speed Analysis

The objective of the speed analysis is to calculate a user selected percentile of the average speed of the vehicles during their passage through the intersection zone for each of the movements defined in the origin-destination matrix. Information on how to create the origin-destination matrix can be found in Section E.7.4

First, Enter the speed threshold parameter (1) (Figure E.17). The units of this value are shown after the text box and depends on the selected units in the main command bar. The threshold is used to filter out those vehicles that at some point during their passage through the intersection have circulated at a speed lower than this threshold. It is important to note that this threshold does not apply to the average speed of the vehicle but to the spot speeds during its passage through the intersection zone.

In addition, the desired speed percentile must be selected in the pulldown list (2). Once these parameters have been selected, click on the *Calculate* button (3) to obtain the results. This operation may take several minutes.

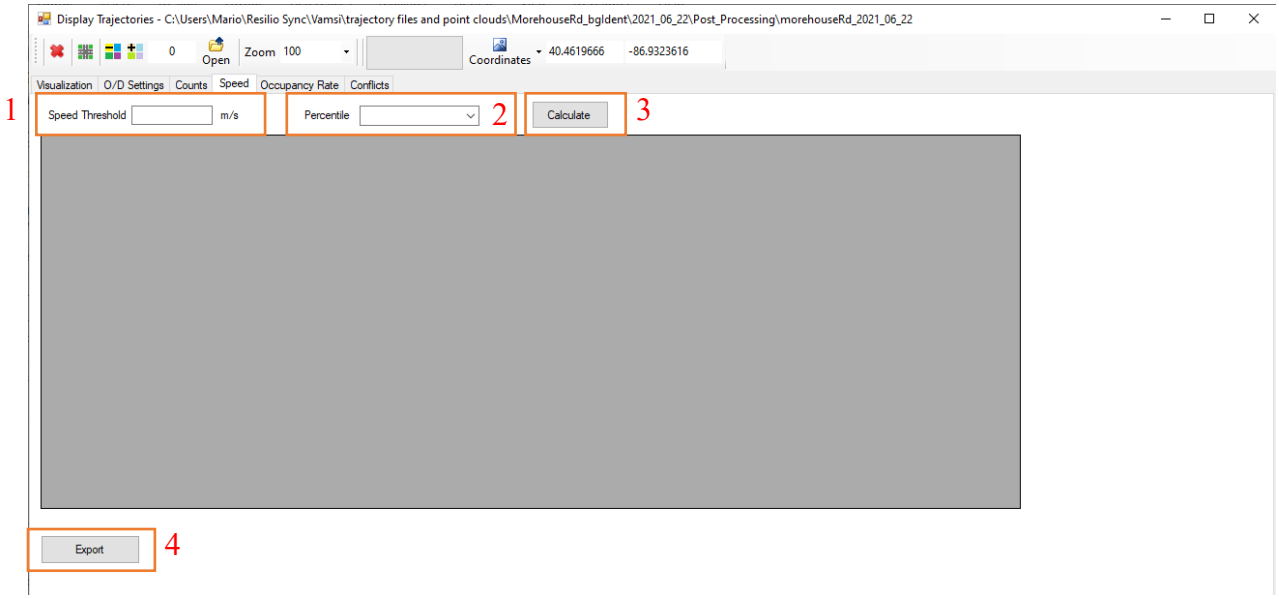


Figure E.17 Speed analysis interface.

Finally, click on the *Export* button (4) to save the results as a comma separated values .cvs file. 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.

E.11 Traffic Conflicts

TScan post-processing generates a list of interactions between vehicles that could eventually be considered as traffic conflicts. The user must filter this list using the parameters that determine whether an action taken by a vehicle is an evasion and therefore it is a potential conflict or not (1) (Figure E.18). Default values are provided but the user can modify them according to his experience.

Once the parameters are set, click on the *Extract* button (2) to obtain the list of conflicts. To determine if one of the interactions is a conflict or not, first a filter is made according to the parameters selected by the user, then the part of the vehicle's trajectory is replaced by a non-conflict path that resembles the one of the vehicles making the evasive maneuver. Then the program checks if a collision happens using the non-evasive maneuver. If so, the time to collision for that interaction is obtained.

The results are presented in a grid and can be inspected in the viewer by selecting the conflict in the grid. Click on the *Export* button (3) to save the results as a comma separated values .cvs file. 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. These files are compatible with the Conflict Diagram Builder program that can be used to analyze the conflicts patterns.

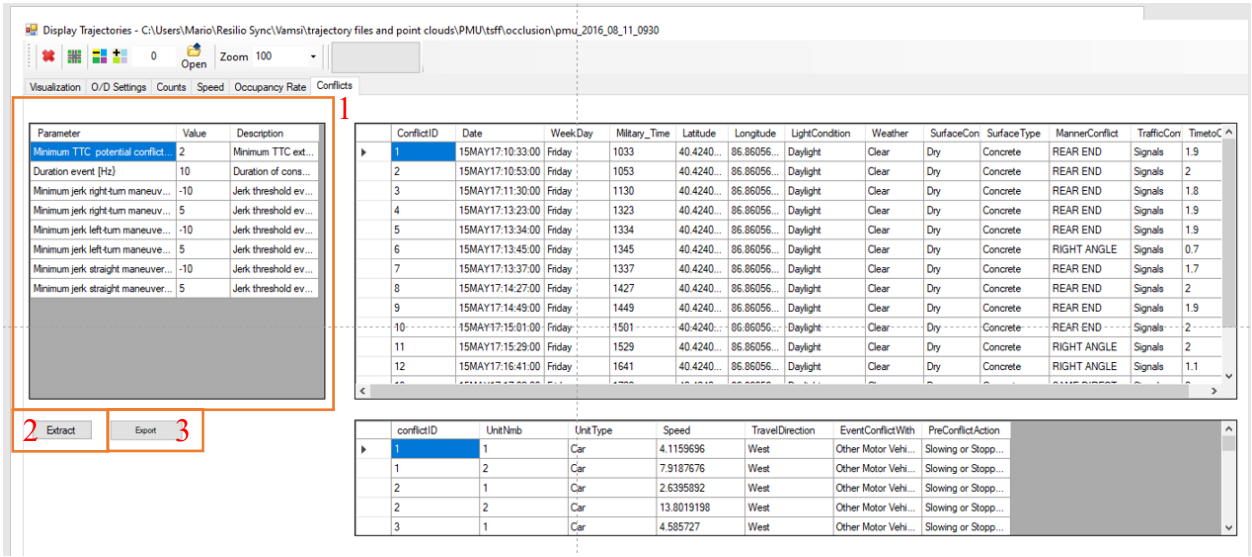


Figure E.18 Traffic conflicts interface.

Additional information about TScan Traffic Conflicts Calculations can be found in Section 7.5.

About the Joint Transportation Research Program (JTRP)

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

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

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

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

About This Report

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

Tarko, A. P., Romero, M. A., Bandaru, V. K., & Lizarazo, C. (2021). *TScan–Stationary LiDAR for traffic and safety applications: Vehicle interpretation and tracking* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2021/31). West Lafayette, IN: Purdue University. https://doi.org/10.5703/1288284317402