

**PERFORMANCE MONITORING FOR SAFE AND
LIVABLE COMMUNITIES: FUSING DATA, TO
IMPROVE ARTERIAL OPERATIONS FOR ALL USERS**

FINAL PROJECT REPORT

by

Michael Dixon, Michael Lowry,
and Randal Brunello
University of Idaho

Yinhai Wang
University of Washington

David Porter and David Kim
Oregon State University

for

Pacific Northwest Transportation Consortium (PacTrans)
USDOT University Transportation Center for Federal Region 10
University of Washington
More Hall 112, Box 352700
Seattle, WA 98195-2700



Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The Pacific Northwest Transportation Consortium and the U.S. Government assumes no liability for the contents or use thereof.

Technical Report Documentation Page

1. Report No. 2012-M-0005	2. Government Accession No. 01539974	3. Recipient's Catalog No.	
4. Title and Subtitle Performance Monitoring for Safe and Livable Communities: Fusing Data, to Improve Arterial Operations for All Users		5. Report Date October 8, 2014	
		6. Performing Organization Code KLLK843; 739436, N14-11	
7. Author(s) Michael Dixon, Michael Lowry, Randal Brunello, Yin Hai Wang, David Porter, David Kim		8. Performing Organization Report No. 5-739436	
9. Performing Organization Name and Address PacTrans Pacific Northwest Transportation Consortium, University Transportation Center for Region 10 University of Washington More Hall 112 Seattle, WA 98195-2700		10. Work Unit No. (TRAIS)	
		11. Contract or Grant No. DTRT12-UTC10	
12. Sponsoring Organization Name and Address United States of America Department of Transportation Research and Innovative Technology Administration		13. Type of Report and Period Covered Research 9/1/2012-9/30/2014	
		14. Sponsoring Agency Code	
15. Supplementary Notes Report uploaded at www.pacTrans.org			
16. Abstract Measuring or analyzing transportation system performance occupies a large transportation professional's time. So, improving performance measurement methods in terms of accuracy and cost are important contributions. This research documents development on five fronts for multi-modal transportation system performance measurement. Researchers developed a performance measurement development tool that leverages the advent of high resolution controller data. The research targets the use of high resolution controller data output from simulation to shorten the performance measurement development cycle. On another front, researchers developed and tested a GIS tool to process sparse bicycle counts and estimate network wide link bicycle counts, enabling transportation agencies to predict bike usage throughout the network. Additionally, researchers developed a portable wireless Bluetooth data collection system that is much more cost effective for short-term studies than existing products on the market. Pedestrian performance measurement is so elusive that obtaining counts is challenging. Researchers developed an application for an off-the-shelf product to count pedestrians using the Microsoft Kinect video game sensor. Finally, researchers developed a method to estimate turning movement counts for most signalized intersections and some unsignalized intersections from lane-by-lane counts. This last development leverages common matrix analysis techniques to assess data collection plans for solution feasibility and provides a solution if it is feasible.			
17. Key Words Turn movement estimation, pedestrian count, bicycle count, Bluetooth, travel time measurement, traffic signal system, performance measurement		18. Distribution Statement No restrictions.	
19. Security Classification (of this report) Unclassified.	20. Security Classification (of this page) Unclassified.	21. No. of Pages 165	22. Price NA

Table of Contents

Executive Summary.....	xi
CHAPTER 1.0 INTRODUCTION	2
CHAPTER 2.0 PERFORMANCE MEASURE CALCULATION USING HIGH-RESOLUTION DATA.....	4
2.1 Introduction.....	4
2.1.1 Overview.....	4
2.2 Literature Review.....	4
2.2.1 Introduction.....	4
2.2.2 Purdue Coordination Diagram	5
2.2.3 Purdue Split Analysis Chart.....	7
2.2.4 Green Time Utilization	8
2.2.5 Delay Measurement	10
2.3 Methodology	11
2.3.1 Introduction.....	11
2.3.2 High-Resolution Data Emulation.....	11
2.3.2.1 Interface	12
2.3.2.2 Flow Chart Narration.....	14
2.3.2.3 Coding Guidance for Anticipated Improvements	17
2.3.2.4 Output from Other Micro-Simulation Software	17
2.3.2.5 Incorporation of External Data	17
2.3.3 Purdue Coordination Diagram	18
2.3.3.1 Introduction.....	18
2.3.3.2 Interface	18
2.3.3.3 Flow Chart Narration.....	19
2.3.3.4 Coding Guidance for Anticipated Improvements	22
2.3.3.4.1 Incorporating Upstream Intersection Detection for Arrival Estimation	22
2.3.3.5 Automating the Generation of Graphs in the Excel Output.....	22
2.3.4 Green Time Utilization	23
2.3.4.1 Introduction.....	23
2.3.4.2 Interface	23
2.3.4.3 Flow Chart Narration.....	24
2.3.5 Phase Termination Analysis	27

2.3.5.1 Introduction.....	27
2.3.5.2 Interface	27
2.3.5.3 Flow Chart	28
2.3.6 Delay/Queue Length.....	29
2.3.6.1 Introduction.....	29
2.3.6.2 Interface	30
2.3.6.3 Flow Chart Narration.....	31
2.4 Testing.....	32
2.4.1 Overview.....	32
2.4.2 Purdue Coordination Diagram	33
2.4.3 Green Time Utilization	34
2.4.4 Queue Length and Delay Estimation	35
2.4.5 Split Failure Analysis.....	38
2.5 Conclusions.....	40
CHAPTER 3.0 USING ORIGIN-DESTINATION CENTRALITY TO ESTIMATE DIRECTIONAL BICYCLE VOLUMES.....	42
3.1 Introduction.....	42
3.2 Centrality.....	45
3.3 Method.....	47
3.3.1 OD Centrality.....	47
3.3.2 Preferred Bicycle Paths.....	48
3.3.3 Origin-Destination Pairs	51
3.3.4 OD Multipliers	53
3.3.5 GIS Toolbox.....	54
3.4 Analysis and Results	56
3.4.1 Case Study Data.....	56
3.4.2 Model Calibration and Validation	57
3.5 Conclusion	61
CHAPTER 4.0 PEDESTRIAN LIVABILITY AND MICROSOFT’S KINECT	64
4.1 Introduction.....	64
4.2 Literature Review.....	66
4.2.1 Depth Based Human Detection.....	66
4.2.2 RGB-D Image Based Pedestrian Detection	68

4.3 Data	69
4.4 Methodology	70
4.4.1 Microsoft Kinect	71
4.4.2 Pedestrian Contour Extraction from RGB Images.....	71
4.4.3 Conversion from Depth Space to RGB Space	73
4.4.4 Pedestrian Extraction	74
4.4.5 Pedestrian Tracking and Counting.....	77
4.5 Experiment.....	79
4.5.1 Experiment results	79
4.5.2 Counting Accuracy and Real- time Performance	83
4.6 Conclusion	84
CHAPTER 5.0 BLUETOOTH DATA COLLECTION SYSTEM FOR PLANNING AND ARTERIAL MANAGEMENT	86
5.1 Introduction.....	86
5.2 Literature Review and Background Information	87
5.2.1 Performance Data with Bluetooth Sensors	87
5.2.2 Data Description	88
5.3 Portable Collection System Design	90
5.3.1 System Description	90
5.3.2 Data Collection Unit Hardware and Software	92
5.3.3 Web-Based Software Application for Data Processing.....	97
5.3.4 System Cost and Packaging.....	100
5.4 Origin-Destination Study Data Collection Test.....	101
5.4.1 Estimation of Total Trip Counts	105
5.4.2 Conclusions.....	106
5.5 Intersection Performance Test	107
5.5.1 Intersection Test Setup.....	108
5.5.2 Test Results.....	109
5.5.3 Conclusions.....	112
5.6 Traffic Signal Timing Evaluation	112
5.6.1 Conclusions.....	116

5.7 Dedicated Short Range Communication.....	116
5.7.1 Technology Readiness	117
5.7.2 Equipment Availability and Economic Feasibility.....	119
CHAPTER 6.0 EFFECTIVE TURNING MOVEMENT VOLUME ESTIMATION FOR INTERSECTION ANALYSIS USING GAUSS-JORDAN ELIMINATION .	122
6.1 Introduction.....	122
6.2 Background.....	123
6.2.1 Analysis Scope.....	124
6.2.2 Detector Location.....	125
6.2.3 Use of detector and phase status event data.....	126
6.2.4 Input data	126
6.2.5 Estimate process.....	127
6.2.6 Summary.....	128
6.3 Methodology.....	128
6.3.1 Overall Method Description	130
6.3.2 Identifying Solvable Intersection Configurations and New Detector Placements	140
6.3.3 Method Validation	142
6.3.4 Conclusions and Recommendations	144
REFERENCES	146
APPENDIX	155

List of Figures

Figure 2.1 Example of a Purdue Coordination Diagram (Brennan, 2011)	6
Figure 2.2 Purdue Phase Termination Chart (UDOT, 2013)	8
Figure 2.3 Green Occupancy Ratio and Split Failures (Smaglik, 2011)	9
Figure 2.4 Simulation to High Resolution Data Emulation	10
Figure 2.5 Simulation to High Resolution Data Emulation	13
Figure 2.6 Purdue Coordination Diagram Interface	19
Figure 2.7 Purdue Coordination Diagram Flowchart	21
Figure 2.8 Phase Termination Chart Interface	24
Figure 2.9 Green Time Utilization Flowchart	26
Figure 2.10 Phase Termination Interface	27
Figure 2.11 Phase Termination Flowchart	29
Figure 2.12 Delay and Queue Estimation Interface	31
Figure 2.13 Delay/Queue length Flowchart	32
Figure 2.14 Hand Assembled PCD	32
Figure 2.15 Tool Generated PCD	33
Figure 2.16 Cyclic Delay of the different methods	35
Figure 2.17 The ground truth calculation of the cumulative arrivals and departures for the first 300 seconds of the simulation	36
Figure 2.18 The high-resolution data calculation of the cumulative arrivals and departures for the first 300 seconds of the simulation	37
Figure 2.19 Delay/Queue length Flowchart	38
Figure 3.1 Correlation between OD centrality and observed volumes for different reachable distance thresholds	53
Figure 3.2 Graphical user interface for the GIS tool to estimate bicycle volumes	55
Figure 3.3 Intersection count locations	57
Figure 3.4 Comparison of: (a) stress centrality and (b) OD centrality	59
Figure 3.5 Estimated one hour peak bicycle volumes for: (a) the entire community and (b) a selected intersection	61
Figure 4.1 Flow chart of the proposed method	70
Figure 4.2 Examples of successful detecting and tracking pedestrians for scenario 2 (Yellow solid circle locating one pedestrian), RGB image on top while corresponding depth image on bottom	81
Figure 4.3 Examples of successful detecting and tracking pedestrians for scenario 1 (Yellow solid circle locating one pedestrian), RGB image on top while corresponding depth image on bottom	81
Figure 4.4 Examples of successful detecting and tracking pedestrians for scenario 3 (Yellow solid circle locating one pedestrian), RGB image on top while corresponding depth image on bottom	82

Figure 4.5 Example of missing detection (Yellow solid circle locating one pedestrian), RGB image on the left while corresponding depth image on right.....	83
Figure 5.1 <i>Point detection</i> refers to estimating when a vehicle containing a discoverable Bluetooth device just passes the DCU marked by the vertical line across the road	88
Figure 5.2 A sample MAC address record stored on a DCU.....	89
Figure 5.3 System deployment for intersection performance estimation	91
Figure 5.4 Microcontroller unit portion of the DCU	93
Figure 5.5 External Bluetooth antenna and cable battery	94
Figure 5.6 Battery and GPS module	94
Figure 5.7 Web application screen shot showing origin-destination data entry	98
Figure 5.8 Web application screen shot showing origin-destination data analysis	99
Figure 5.9 DCU deployed on I-5 for origin-destination data collection.	100
Figure 5.10 Data collection locations in the Corvallis, Albany, Lebanon area	102
Figure 5.11 Data collection location used to estimate the vehicle fraction detected.	105
Figure 5.12 Intersection utilized for the data collection test.....	108
Figure 5.13 Test setup for intersection performance data collection test	109
Figure 5.14 DCU deployment locations (yellow pins) along Highway 99W in Sherwood, Oregon	114
Figure 6.1 Intersection Concurrency Groups, Counts, Lane Numbering, and Lane Specific Movement Definitions	131
Figure 6.2 Example Matrix “a0” (first concurrency group and second concurrency group).	133
Figure 6.3 Group 1 “a” matrices.	134
Figure 6.4 Matrix input to Step 8 from Step 7 and Matrix resulting from Step 8.	137

List of Tables

Table 2.1 Comparison of Hand Calculated GTU to Tool Calculated GTU.....	32
Table 2.2 Hand Calculated and Program Results for Creating the Purdue Phase Termination Charts.....	36
Table 3.1 Intersection Control and Cross Street Impedance Factor, F_c	51
Table 3.2 Estimation Coefficients and Model Statistics.....	58
Table 4.1 Pseudo-code of Pedestrian Extraction Algorithm.....	74
Table 4.2 Pseudo-codes of Pedestrian Tracking and Counting Algorithm.....	77
Table 4.3 Summary of counting test results.....	83
Table 5.1 DCU Component List.....	95
Table 5.2 DCU Components – Approximate Pricing.....	101
Table 5.3 Data Collection Locations for Origin-Destination Data.....	102
Table 5.4 DCU Deployment Periods for Origin-Destination Data Collection.....	103
Table 5.5 Maximum Travel Time used for Origin-Destination Data Processing.....	103
Table 5.6 Trip Counts.....	104
Table 5.7 Average Travel Times for the Trips in Table 5.4 (minutes).....	104
Table 5.8 Total MAC Address Counts.....	104
Table 5.9 Total Unique MAC Address Counts.....	104
Table 5.10 DCU to DCU Travel Time Accuracy Results.....	111
Table 5.11 Travel Time Data Collection Summary for Highway 99W Signal Timing Evaluation.....	115
Table 5.12 Travel Time Data Collection Summary for Highway 99W, 6AM - 8AM.....	115
Table 5.13 Travel Time Data Collection Summary for Highway 99W, 4PM – 7PM.....	116
Table 5.14 Most Relevant Specifications of DSRC and Other Wireless Technologies.....	118
Table 6.1 Turn movement estimates for Concurrency Groups 1 and 2.....	138
Table 6.2 Turn Movement Count Feasibility Examples – Minimum Detection.....	141
Table 6.3 Validation Test Results.....	144

Executive Summary

This project seeks to investigate the developments of and the potential for different methodologies for gathering arterial traffic performance data. In the effort to build more livable communities, this data is essential, but gathering, organizing, and applying it has, historically, been a difficult task. This report is organized around product areas, which are 1) an open-source tool to monitor dynamic performance measures from high resolution traffic controller data, 2) a practical and accurate tool for estimating bike volumes, 3) cost-effective pedestrian detection, 4) inexpensive and quickly applied tools to extract probe vehicle data, and 5) a pragmatic approach to accurately estimate signalized intersection turning movements. Each chapter of this report is an autonomous effort in studying one of these areas in particular. This project developed resources and performance measures to inform efforts to achieve the goals of safety and efficiency to promote community livability.

Chapter 2 proposes a tool that facilitates future performance measurement research related to traffic signal systems. The tool imports data from a simulation data source for experimentation in varied ideal settings or from field traffic signal systems for more rigorous application testing. Currently, imported traffic controller data are combined to produce dynamic performance measures, including the Purdue Coordination Diagram (PCD), Green Time Utilization (GTU), phase termination, and queue length/delay. These are integrated to be visualized with the PCD acting as the background. Additional measures and tasks can be supported by the tool's data import and various database functions.

Chapter 3 uses origin-destination centrality to estimate directional bicycle volumes. Limited input data, simple site specific calibration, trivial modeling requirements, and practical accuracy make this method very attractive relative to proposed alternatives. In addition, this

research provides the tools to import input data, process the data, estimate the bicycle volumes, and visualize the results with add-on applications created for industry-standard off-the-shelf software.

Chapter 4 proposes an efficient pedestrian detection method for crowded scenes by fusing RGB and depth images from Microsoft's ® Kinect. While traditional image-based pedestrian detectors provide very rich information, their performance degrades quickly with increased occlusion. The 3D sensing capabilities of Microsoft's Kinect present a potential cost-effective solution for occlusion-robust pedestrian detection. The results of the study demonstrate the feasibility of using the low-cost Kinect device and a proposed detection method for real-world pedestrian detection in crowded scenes.

Chapter 5 documents the research and development of an inexpensive portable wireless roadside data collection system using probe vehicles, whose movements are monitored using Bluetooth technology. The system addresses industry needs for low-cost portable traffic monitoring and supports travel time, origin-destination, and delay performance measures. The research also reviewed the potential of Dynamic Short Range Communications (DSRC) for to accomplish the same tasks and found DSRC advantages include 1) low-latency communications, 2) broadcast messages, 3) greater communication range (+200 m), and 4) greater bandwidth. Limited availability and expense of supporting hardware are major disadvantages of the system, which will subside as technology adoption spreads.

Chapter 6 presents a method that solves for turning movement volumes using Gauss-Jordan elimination row operations (e.g., row swapping, multiplying rows by non-zero constants, and adding a factor of one row to another row). The input data are phase status, lane-by-lane

detector counts, and limited exit detector counts. It evaluates existing intersection detector locations for their combined suitability to estimate turning movements and selects detection plans that minimize data requirements. The method accommodates varying lane configurations, varying detector locations, and includes or excludes phase status. Because the method is founded on direct implementation of basic matrix analysis row operations, the solution process is easy to implement. Three data sets validate the method's accuracy, with and without detection error, showing the method can be sufficiently accurate for professional applications in planning, design, and operations.

Arterial system performance measurement is important for assessing steps considered or taken to accomplish goals directed toward greater community livability. Several tools and methods were developed to collect data, import data, process data, and to estimate performance measures. These tools were tested to prove their feasibility and each chapter provides an insightful and detailed evaluation of these products.

CHAPTER 1.0 INTRODUCTION

The problem many transportation professionals face is measuring performance and correcting poor performance to meet community goals. Measuring performance using existing traffic data is natural. In fact, all performance measures either require traffic data or could greatly benefit from it. Fortunately, these traffic data exist in key locations found in signalized arterial networks. However, very little of these data are fully utilized to measure performance. This project developed methods and technologies to gather data from multiple sources to enable a more complete understanding of arterial traffic safety and arterial systems efficiency. This understanding will strengthen steps that professionals take to improve service.

Historically, performance measurement was largely an off-line, labor intensive endeavor. Gradually, this is changing to leverage widespread communication systems and technological resources for sensing, data processing, and systems management. Transportation systems benefit from these changes. For example, data from the large majority of signalized intersections are available to measure performance. However, measures need to strongly support local intersection and system level diagnostics to inform planning, design, and operational decisions that promote livable communities by way of multimodal safety and efficiency improvements.

This project developed resources and performance measures to guide efforts to achieve the goals of safety and efficiency to attain greater community livability. Specifically, this project's goals were to: 1) develop improved performance measurement for a diverse set of transportation modes, 2) develop processes/tools needed to estimate performance measures, and 3) develop a foundation for researching secure control feedback for mode, safety, weather and traffic condition sensitive response.

To achieve the above three objectives, the project team focused their efforts on product research and development to foster technology transfer. In light of this product focus, this report is organized around these product areas, which are 1) an accessible open-source tools to extract dynamic performance measures from high resolution traffic controller databases, 2) a practical and accurate tool for estimating bike volumes, 3) a cost-effective pedestrian detection technology, 4) inexpensive and quickly applied tools to collect probe vehicle data, and 5) a pragmatic method and tool for calculating intersection turning movement volumes. A chapter is dedicated to each of these areas. Each chapter is autonomous, containing a literature review, methodology, discussion of research, and findings and conclusions.

CHAPTER 2.0 PERFORMANCE MEASURE CALCULATION USING HIGH-RESOLUTION DATA

2.1 Introduction

2.1.1 Overview

The purpose of this research project was to create a tool that could be used to facilitate future research of traffic signal systems. The tool generates performance measures using high-resolution data collected by traffic signal controllers. High-resolution data is a record of every state change that occurred for a traffic controller. It usually consists of four fields for each state change: the intersection number, the phase number, the time, and an event identifier. There are all kinds of state changes that can be logged. This project however, is only interested with the signal timing and detector activations and deactivations. The tool can also be expanded to use multi-source data such as Bluetooth probe data or speed detection. Furthermore, this tool facilitates the generation of new performance measures because it can generate high-resolution data from traffic simulation software, such as VISSIM. The tool includes some performance measures which can be improved by adjusting open-source code for the data processing algorithms. Lastly, this tool can be expanded to experiment with the application of performance measures for strategic decision making.

2.2 Literature Review

2.2.1 Introduction

The purpose of this literature review is to describe the common performance measures that can be generated using high-resolution traffic controller data. The performance measures

discussed in this review were all included in the proposed tool. These measures and their application continue to be improved and are recommended for future research. In conjunction with one another, these measures can identify most operational problems at traffic signals as well as help find a solution. As such, they are prime candidates in a prototype performance monitoring tool.

2.2.2 Purdue Coordination Diagram

Darcy Bullock advanced the application of high-resolution performance measures by developing the Purdue Coordination Diagram (PCD) (Brennan, 2011). This diagram is a chart that displays arrivals in relation to the time in the cycle and the cycle in relation to the time of the day. Only one phase is analyzed on a single diagram. Figure 2.1 below is an example of a PCD. It is used to analyze the quality of coordination, among other system monitoring activities. Each point on the diagram is a vehicle arriving at the intersection, ideally provided by lane-by-lane advanced detectors. If a point lies above the green line, which is the beginning of green (BOG), on the graph then the arrival occurred during green. The red line is the end of green (EOG). As a result, a well-coordinated system shows vehicle arrivals as a dense cloud, predominantly above the green line, as indicated by the “ii” note in Figure 2.1.

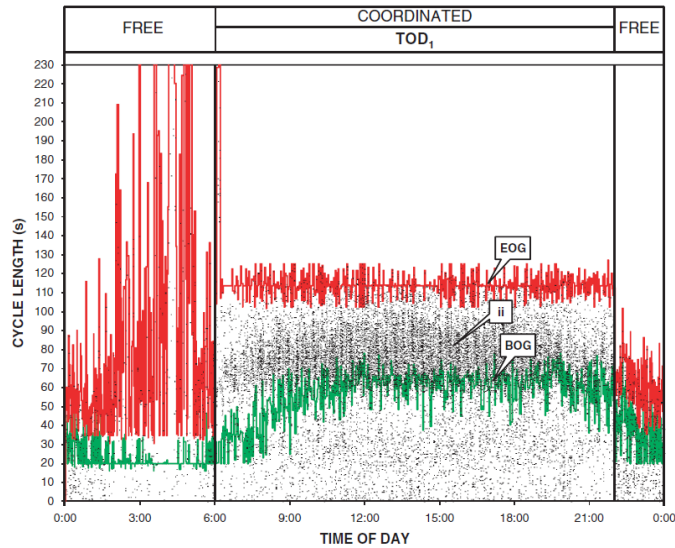


Figure 2.1 Example of a Purdue Coordination Diagram (Brennan, 2011)

The PCD does require significant attention to gather all of the important insights that may not be obvious. Issues like queues extending over the advance detectors and the lack of numerical quantification can make using the PCD challenging. For this reason Utah Department of Transportation (UDOT) includes average performance measures for each time of day plan. The percent arrival on green, percent green time, and platoon ratio give a numerical quantification that is a good representation of one of their diagrams. The percent arrival on green is calculated as the number of arrivals during the green time divided by the total arrivals during that study period. Similarly the percent green time is simply the total green time divided by the total time of the study period. The platoon ratio combines these two measures and is calculated as the percent arrivals on green divided by the percent green time.

The PCD sheds light on a broad range of traffic operations problems, like queue spillback or poorly timed offsets. At the same time, the PCD integrates well with more aggregate measures. As a result, the proposed tool should include the PCD to help advance performance

measurement by generating multiple performance measures, while leveraging the PCD to see individual vehicle operations.

2.2.3 Purdue Split Analysis Chart

The UDOT has implemented an extensive system that logs high resolution data for the majority of the traffic signals on the state system (UDOT, 2013). Their system allows users to view performance measures depending on the detection setup. For example if the intersection has advance detectors, PCDs can be generated. Most high-resolution performance measures depend on detectors. The Purdue Phase Termination Chart is the only measure that does not require detector data. This chart, see Figure 2.1, shows the conditions that lead to the green ending. It color codes max-outs, gap-outs, and force-offs. Then, comparing different phases, it can help identify problems with split times and in some cases find malfunctioning detectors. For instance, in Figure 2.1 Purdue Phase Termination Chart (UDOT, 2013) below, shows a plot created on UDOT's performance metrics website. The green dots are gap-outs, the blue dots are force-offs and the red dots are max-outs. Since the max-outs are similarly colored to the pedestrian walk markers, a group of max-outs has been circled in the figure. The orange dots that are just above the phase are pedestrian walk indications. There are two coordinated periods, one in the AM and the other in the PM. Because of the consistent force-off phase termination, it is clear that phases 2 and 6 are the major approach that is coordinated. During plans 1 and 13 where there is coordination they are always forced off.

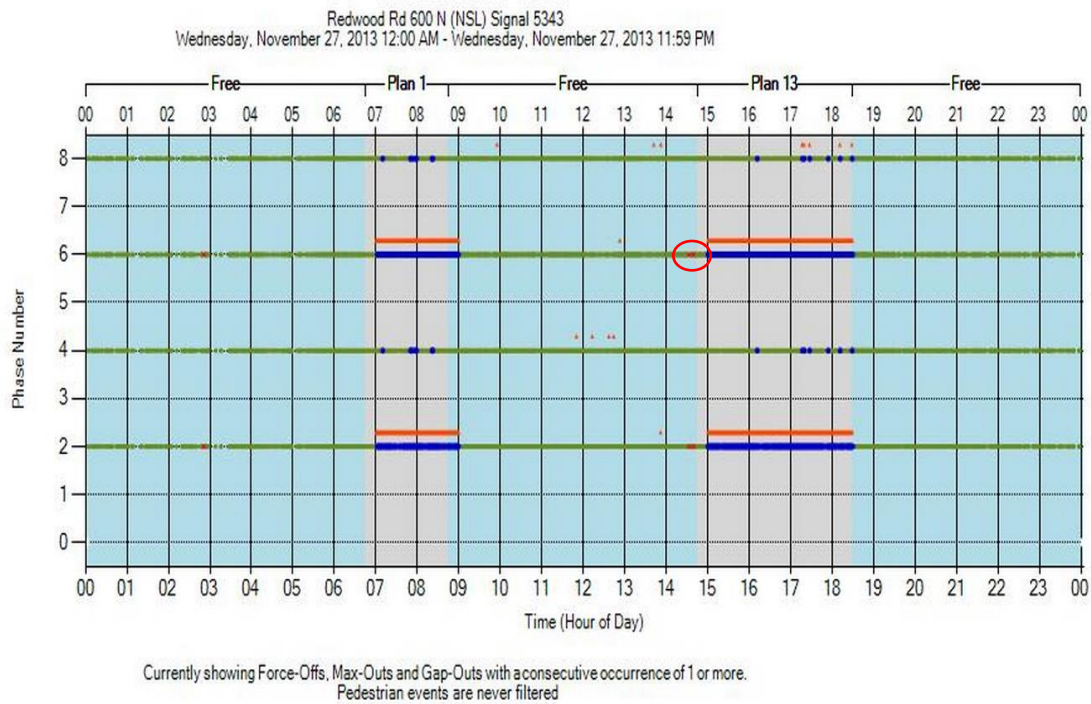


Figure 2.1 Purdue Phase Termination Chart (UDOT, 2013)

This measure is very simplistic but can still be very useful. For instance, in the case shown above, if another measure identified a problem on the major corridor this diagram could be consulted to compare the different phases. Due to the consistency of the minor phases gapping out, this intersection may benefit from having its split times adjusted to better meet its demands. Unfortunately, nothing concrete can be determined from this measure alone; it is simply a companion to the other measures. Nevertheless, when this measure is considered in parallel with other measures, it helps diagnose problems with signal timing plans and should be included in the proposed tool.

2.2.4 Green Time Utilization

Green Time Utilization (GTU) is a simple performance measure which tells you the percent of the green time that the stop bar detectors were active. This serves as a surrogate

measure for capacity utilization. However, this measure requires some calibration for proper use. Calibration is necessary because the variety of detection types, accuracy, and sensitivity affect the GTU significantly. However, even after being calibrated it still does not have a significant correlation with delay. A study found that GTU had an r^2 of .513 when correlated to delay, (Smaglik, 2011). It can be a good indicator of the performance even though it is not very precise. In their study, when the GTU was greater than 95%, half of the cycles were split failures, meaning the queues failed to clear during the green time (see Figure 2.1). Note that, in Figure 2.1, Smaglik used the term green occupancy ratio (GOR) instead of GTU.

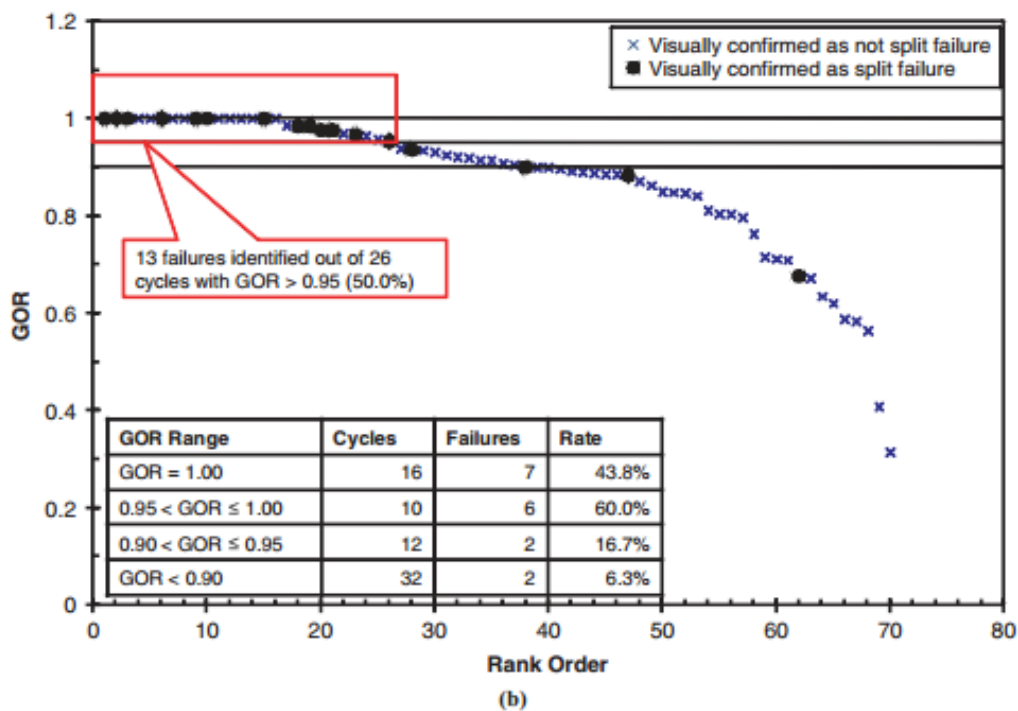


Figure 2.1 Green Occupancy Ratio and Split Failures (Smaglik, 2011)

This performance measure is germane to traffic signal systems operation and, as illustrated above, is a simple indicator of impending problems and imbalances in phase services. As a result, GTU is a strong candidate for inclusion in the proposed tool.

2.2.5 Delay Measurement

UDOT also did a study on estimating delay (Saito, 2011). It used loop detector data to estimate delay and travel times using stop bar data from the intersection in question and the upstream intersection. The estimated travel time would determine matches between upstream and downstream detector hits. The dataset was trimmed to not include vehicles that were already in the system at the start of the analysis or still in the system at the end of the analysis. They also had to account for vehicles exiting and entering mid segment. This problem led to the creation of two methods. The first method had detectors on the mid segment driveways so that the vehicles that exited and entered could be accounted for. The other method did not have these mid segment detectors. Currently, the second method is the only one that might be viable for use in the proposed tool, until a feature for midblock detection is added. Once the vehicles entering and exiting mid-stream have been removed from the analysis, the process checks to make sure that all the data points from the upstream detectors have matches with data points from the downstream detectors. Then the average travel time is calculated. First, the sum of the upstream detector hit times is subtracted from the sum of the downstream detector hit times. That value is then divided by the total number of vehicles. The average delay time is then defined as the difference between the average travel time and the free flow travel time (driving the speed limit). The results of this study seemed mixed with errors in delay less than 5 seconds per vehicle for most cases and the percent error varied from 30 to 40 percent. The test conditions were not sufficiently varied to generally conclude the method's value. However, longer distances will likely require more complex considerations of driver behavior to successfully match a vehicle's upstream and downstream detections. These complications arise from more midblock traffic and larger travel time variations.

The proposed tool should include delay as a performance measure. However, to minimize complications, the distance between entrance and exit detectors should be much shorter than what was used in UDOT's test, at approximately 400 feet.

2.3 Methodology

2.3.1 Introduction

The system for generating performance measures requires two parts. The first part is comprised of three different elements: a database that contains the high resolution data, a database table that contains the maximum green times, and a tool for converting VISSIM 5 outputs to a high resolution database (VISSIM is a traffic microsimulation software developed by PTV group). The database used in this project was Microsoft SQL Server. The other part is the suite of python programs, called the Performance Measurement Research Suite. There are a total of seven files that make up the suite, but only two of the files need to be run or edited to use the program: the file that produces the graphical user interface (GUI) and the file that controls the default information. In order to run the different performance measure programs all that must be done is open the GUI program. In order to change the default information the python file named "CustomDefaults.py" must be edited.

2.3.2 High-Resolution Data Emulation

This high-resolution data emulation program, or emulator, converts VISSIM 5 output files to basic eventcode style high resolution data. High resolution data are a record of every change of state. It includes the time stamp of the state change, the eventcode to identify the state that is changing, the intersection identifier, and the phase identifier. These eventcodes are extracted from the detector output files (.ldp) and the signal output file (.lsa). The (.ldp) files

must be formatted in a specific format so that the time in seconds of the simulation is the first column followed by an empty column and then followed by the corresponding detectors which are also separated by an empty column. Any traffic controller in VISSIM 5, including hardware in the loop, is acceptable, so long as the (.ldp) and (.lsa) files exist in the VISSIM simulation configuration.

2.3.2.1 Interface

The input fields for running the emulator can all be customized using the CustomDefaults.py. After customizing, the emulator GUI starts by requesting a unique table name, so that a table will be created holding the output data in the same format that high resolution data would be collected in Centracs. The next item to select is a valid VISSIM 5 (.inp) filename for a simulation by browsing or entering in the path and filename. All (.ldp) and the (.lsa) files will need to reside in the same directory in order to be found at this time as this is where they are output by default from VISSIM 5. Additional (.kfg) files are also needed to determine the correct detector name that corresponds with the data in the (.ldp) output file. The program will allow for any number of signal intersections and detectors to be used as long as there is an appropriate (.ldp) and (.kfg) file created based on the (.inp) file selected.

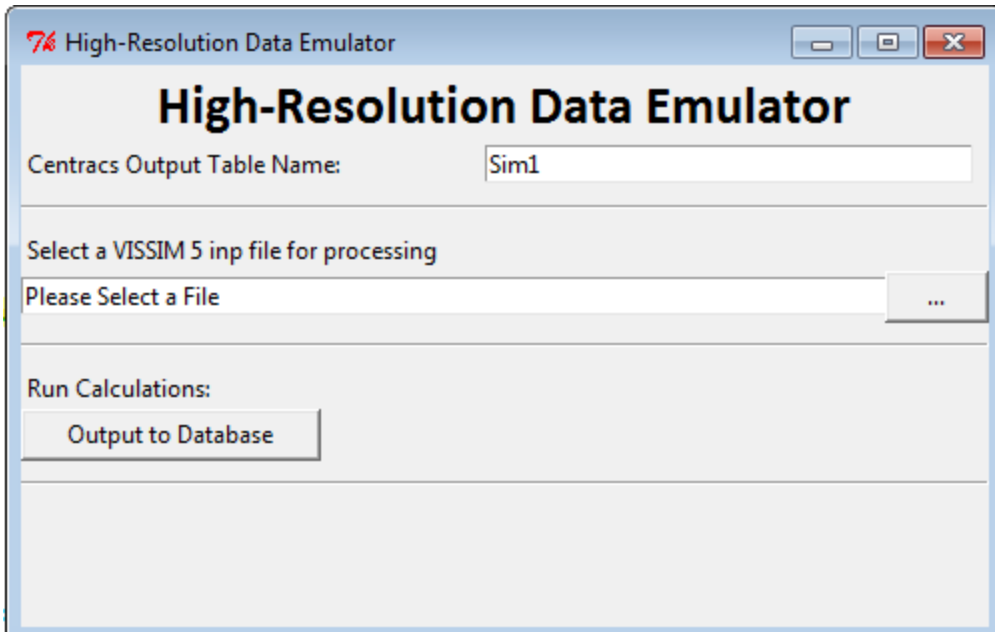


Figure 2.4 Simulation to High-Resolution Data Emulator

There are two buttons that follow the entry fields. The first button is a “Browse” button for selecting and collecting the simulation output data. First, the detector data are collected in the order of the intersections entered into the input field. Following the detector data’s collection, the phase changes are collected. These files are then each entered into a table in the database. The “Output to Database” button runs the analysis and conversion of the output files to high resolution data. The detector data are output by VISSIM every time step. Therefore, it must be simplified down to detector activation and deactivation times. The phase changes are already output by VISSIM in the form of state changes. Only the format must be changed to fit the event code style. Also, the green times are checked with the maximum green to determine the reason for phase termination. The maximum green times are retrieved from a table in the database. The table is formatted to have 3 fields, the intersection, phase, and maximum green. These fields were integers in the example but could be any numerical datatype that is needed. Each file is analyzed and entered into a string. This string is then entered into the database. Since the order of

the data produced by this program is not chronological, all queries to this table need to include an “ORDER BY TimeStmp”.

2.3.2.2 Flow Chart Narration

There are two levels of flow charts (see Figure 2.5). The first level introduces each function in the program. The second level explains each step through the program’s functions. Each of the functions has its own smaller flow chart since they do not need to be run in order, although it is recommended.

This program has three sections of code. The first section of code generates the graphical user interface. This consists of two entry fields and two buttons as well as an area at the bottom for messages for the user if input needs to be corrected or an error occurs. The first entry is the name of the database table that will have all data from the VISSIM 5 simulation entered into it. If the table already exists the program will ask for a new table name. The second entry field is for the VISSIM 5 (.inp) configuration file that will tell the program what VISSIM (.ldp) and (.kfg) files exist so that the program can insert all available data into the table that will be created. The entry must be a full path to the (.inp) file, a button with three periods is next to this entry field so a user can browse to the file if they do not know the full path. If the file does not exist the program will not run. The last button labeled “Output to Database” will start the processing based off of the (.inp) file selected. If there are any errors, this will be reported in the message area at the bottom.

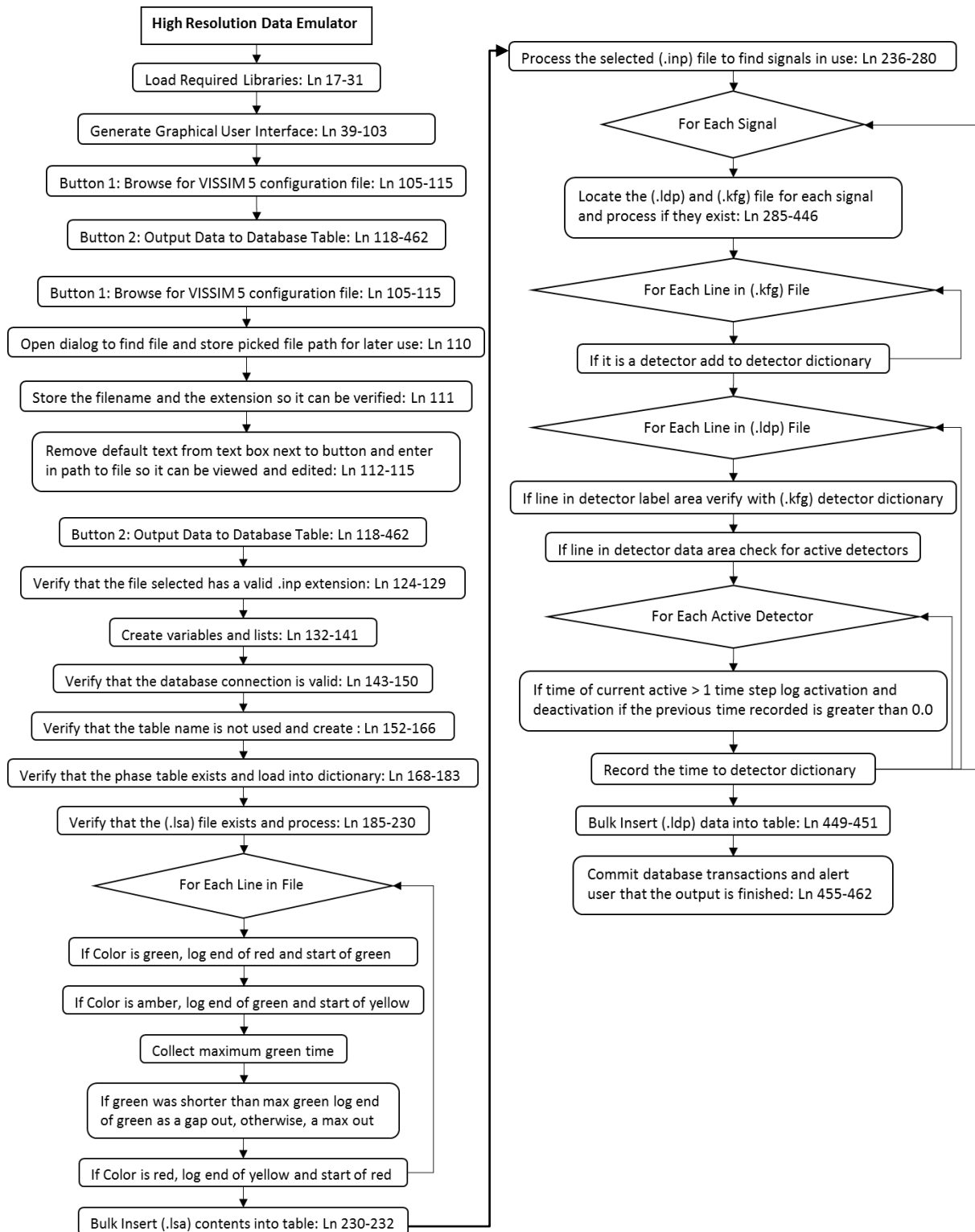


Figure 2.5 Simulation to High-Resolution Data Emulator

The second section of code is activated when the button with the three periods is pressed. This will open a window to browse to an (.inp) file. This section ties in with the third, since the third section will validate that this file is the correct file type.

The third section is where the processing of the (.inp) file occurs. The first process that occurs is that the (.lsa) file will be looked for, and once found it will be processed so that each line is examined for changes in the signal values. The (.lsa) data will give information of when the light status changes and at what time so this can be entered into the table in the appropriate formatted state-change events. In addition to the file being processed for the data that is there, any phase state change to amber will result in determining if the change was due to a max-out/force-off or a gap-out. The data is saved into a python list until the end of the file. Once end of file is reached, a bulk insert query is initiated so the data will be uploaded into the database table.

After processing the (.lsa) file, the (.inp) file will allow the program to find all relevant (.ldp) output files as well as their corresponding (.kfg) file. The (.kfg) file holds all the relevant information about the detectors and how the (.ldp) file is to be formatted. Once the (.kfg) file for the signal is processed, then the (.ldp) file is processed and it is made of 4 sections; the program is only interested in the third and fourth section of the (.ldp) files. The third section will allow the data in the (.kfg) file to be validated, so we know the data is formatted correctly. The fourth section holds all the relevant detector state data. The program will process all lines in the fourth section and based on the detector status being activated and will record this activated time in a python dictionary for later lookup. If the time in the dictionary of the last activation is more than one time step, then the program will append to a python list that the detector has changed status

indicating a deactivation at the previous recorded time plus one time step as well as an activation at the current time. At the end of each (.ldp) file a bulk insert statement is made to insert all this data into the database table.

2.3.2.3 Coding Guidance for Anticipated Improvements

Only VISSIM 5 files can be used at this time. Additional work is needed to incorporate VISSIM 6 files.

2.3.2.4 Output from Other Micro-Simulation Software

Changes in output file format will be difficult to address without significant code changes. If there are only minor changes, such as detector status changed to “true”/“false” or “1”/“0” instead of “.”/“\” then the change will be simple. If the output files are in the form of status changes already then it would likely be easiest to recode all the logic, only maintaining the writing of events to the string. The “if” statements in the loop that analyzes the signal timing output file will require adjustment. The string they check for is very specific, requiring the exact number of advance and trailing spaces. One of three alternatives may be easiest: 1) write a separate program that changes the output files to match what is required, 2) provide separate logic for inputting data, depending on the source simulation, or 3) allow the user to define the data delineation.

2.3.2.5 Incorporation of External Data

Including other forms of data, like Bluetooth probe detection or manually collected data, is fairly simple. As long as you can produce an event code entry from external data, it is a simple matter to add it to the database. If it will occur in every study it may be included in the current functions. More likely however, this process would be added as a separate function that runs upon button press. This would require the external data to be read into the program, either by

reading a text file, independently from any existing function in this program, or by inputting a file into the database and reading it from there. Then the program would process the data and append events to the same string that is produced by the emulator. This can be done entirely with processes already developed within the program.

2.3.3 Purdue Coordination Diagram

2.3.3.1 Introduction

This program analyzes high resolution data and outputs an Excel file that contains data organized into columns. This Excel file can easily be used to create a Purdue Coordination Diagram, or PCD, using an Excel scatter plot. It also produces summary measures including: percent time green, percent arrivals on green, and platoon ratio for each cycle and overall. These cycle-by-cycle measures are plotted with the PCD.

2.3.3.2 Interface

This interface, seen below in Figure 2.6, is the base interface for most of the following programs. It has 4 fields that collect information. As with all of the programs, the default values for the graphical user interfaces (GUI) can be changed using the CustomDefaults.py file. The first field is the intersection numbers separated by commas. The second field is the phase numbers separated by commas. Only one phase number can be selected for each intersection selected. However, the same intersection may be listed multiple times to include multiple phases from one intersection and any number of intersection/phase pairs are acceptable. The detector input field is slightly more complicated. The detectors for a given phase are separated by commas. Then each phase's group of detectors is separated from the other detectors by a semicolon. The final entry is the database table name, where the data will be retrieved. At the very bottom of the GUI, there is a button which will run the program once the four fields are

completed. This button will run the entire program from retrieving inputs to creating the excel output file.

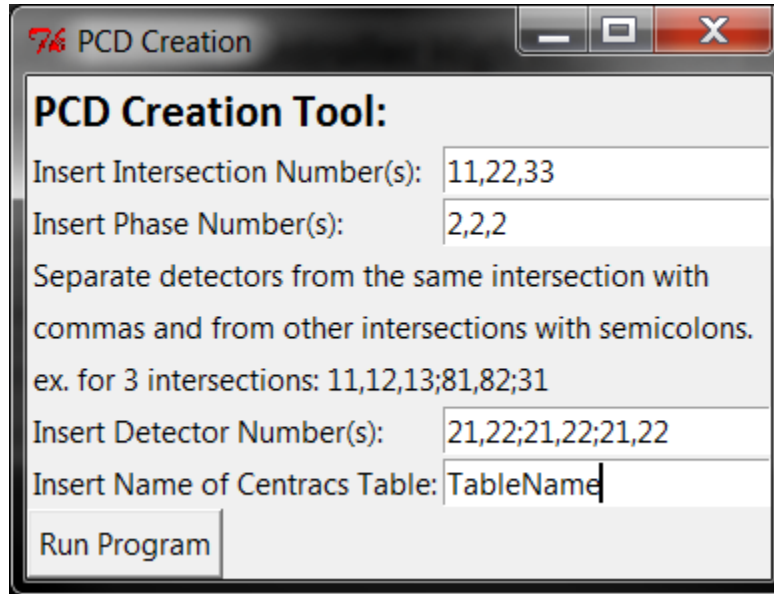


Figure 2.6 Purdue Coordination Diagram Interface

2.3.3.3 Flow Chart Narration

The Purdue Coordination Diagram program has two parts and these are illustrated in the flow chart below, Figure 2.7. In the first part, it constructs the graphical user interface, seen in steps 2.1 to 2.4 of the flow chart. The second part runs the analysis based on the information entered into the graphical user interface, seen in steps 3.1 to 3.12 of the flow chart.

The user interface is described in detail above, so this discussion emphasizes the calculation process, which begins by retrieving the information from the graphical user interface. Then the process connects to a database and creates an excel workbook, steps 3.3 and 3.4. Once this has been done, it loops through the intersection/phase combinations that were entered into the graphical user interface. For each loop, the program uses the remaining input information to construct three SQL queries, step 3.6. The queries retrieve all of the intersection green times, red

times, and detector activations for the given phase. These are stored in three separate lists. The process organizes the data cycle-by-cycle by looping through the beginning-of-red time list. The beginning-of-red time indicates the ending time of the current cycle for which detection data are being organized. For each cycle's red time, the program loops through the detector times, searching for detections that occurred during the cycle. The loop through the detection times stops when a detector time occurs after the beginning-of-red time. Each detector time is logged as an arrival in the workbook. Next, after completing looping through the detector data the program checks the green time list to see if the green time is the correct one. It does this check by making sure that the green time occurred between the previous beginning-of-red time and the current beginning-of-red time. It then logs the green time in the workbook for the current cycle for which data are being organized. Once it is done looking at the detector times and green times, it calculates the cycle's performance measures. Percent arrivals on green, percent green time, and the platoon ratio are the measures calculated. The program then moves to the next cycle's red time and repeats this process until there are no more red times. Once all the red times have been completed, the program steps to the next intersection and repeats the process again. Last, the program saves the workbook and it is ready to be used to create a PCD. All output excel files are saved in the same directory as the python programs. The excel files are named as the performance measure and datetime separated by an underscore. For example, the format, PM_Month_Day_Hour_Year, results in a PCD_Jan_15_2_30.xlsx.

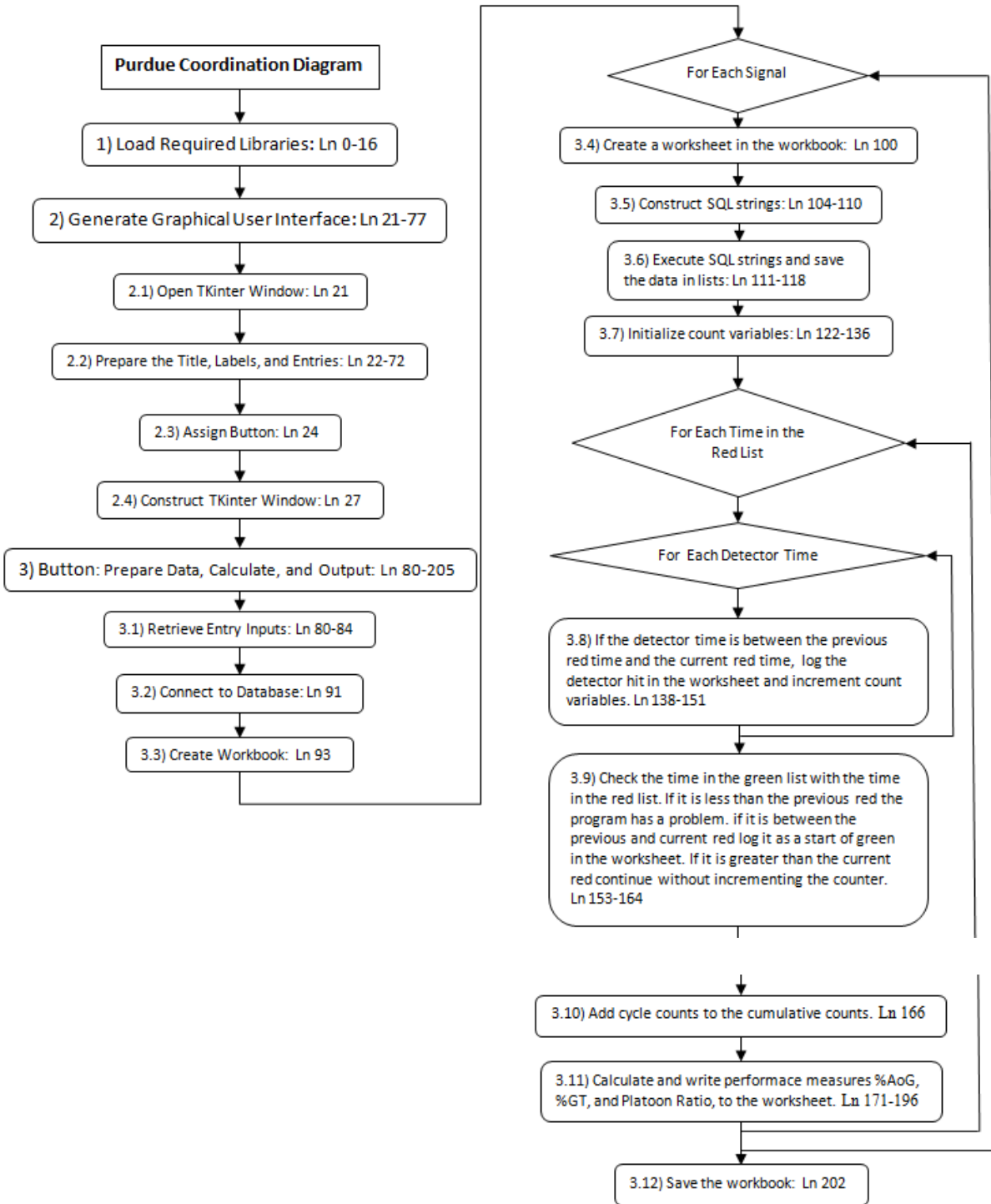


Figure 2.7 Purdue Coordination Diagram Flowchart

2.3.3.4 Coding Guidance for Anticipated Improvements

2.3.3.4.1 Incorporating Upstream Intersection Detection for Arrival Estimation

PCD-oriented information is most useful when detection occurs upstream of the subject intersection's queue. In many cases, there are no detectors at this strategic location. Using arrival estimations from an upstream intersection is an easily implemented addition to the program. There will need to be an addition to the GUI's current entry fields, allowing the user to enter the upstream intersection information, which is the intersection number and the phase numbers that contribute to the subject intersection/phase arrivals. This program would then need to collect departure data which will be processed to create the list of arrival times. For example, the program would retrieve the departure times and add the travel time between intersections. The travel time could be based on probe data or simply defined as the distance divided by the estimated speed. The resulting values would be estimated arrival times. The program would then need to refer to these estimated arrival time values instead of retrieving arrival times directly from queried detector results.

2.3.3.5 Automating the Generation of Graphs in the Excel Output

Currently, the user needs to manually create the PCD diagram from the output data that this program creates. The program can be modified to automate the generation of graphs in excel using Visual Basic for Applications (VBA). The VBA program can either be run in excel after the output file has been created or VBA can be executed by the python program after the workbook has been saved.

2.3.4 Green Time Utilization

2.3.4.1 Introduction

This program analyzes high resolution data and outputs to an excel file that contains both cyclic and overall performance measures. The primary performance measure calculated is the Green Time Utilization. However, detector hits are also counted, which can be used to measure the most basic performance measure, flow rates.

2.3.4.2 Interface

This interface, shown below in Figure 2.8, is very similar to the Purdue Coordination Diagram's interface. There are four entry fields: Signals, Phases, Detectors, and Table Name. They work the same as the Purdue Coordination Diagram. The intersection field accepts comma-delimited values. The Phases field requires the same number of entries as the intersections field. The intersection and phase values are considered as pairs. For instance, the first intersection-phase pair would be 1-2 and the second pair would be 1-6. The detectors field follows the same rules as they have in the previous interface but the user should use a stop bar detectors if they are available. For the best results, the Purdue Coordination Diagram program uses advance detectors, while stop bar detectors would be best for the GTU measurement. The final entry is the name of the high resolution data table. The "Run Calculations" button will run the calculations and produce an excel file with the calculated GTU measures.

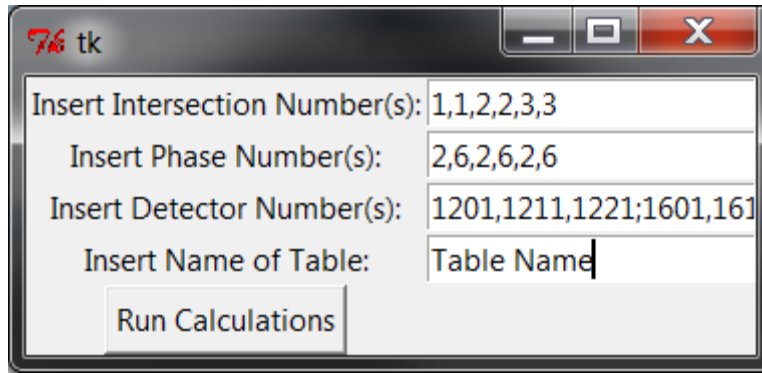


Figure 2.8 Green Time Utilization Chart Interface

2.3.4.3 Flow Chart Narration

This discussion provides an overview for the computer program that extracts GTU information from high resolution controller data and uses the flowchart in Figure 2.9 for illustration. The program starts by importing all the libraries required to function. These are all included with the scripts. The graphical user interface is then constructed. There are 4 entry fields and a button. The button “Run Calculations” executes this program’s only function. As shown in steps 3.1 to 3.13, this function collects the data from the database, processes it, and outputs the results into an Excel file.

The first step is to collect the information entered into the entry fields, step 3.1. Next it enters a loop where it analyzes each intersection-phase pair. In the loop, it initializes the temporary count and summation variables that will be used, step 3.5. Then it produces a query that selects all detector activation and deactivation times along with the start of red and start of green times. Each of these times also has a corresponding event. These values are arranged in one list which is analyzed in the next steps, 3.8-3.10. Each loop processes one event. If the event is a green light a few things are done. First the cycle counter is incremented up one. Then the number of hits in the previous cycle is appended to a list where they are stored until the reporting

step. This is followed by the green light Boolean being set to true and the previous green time being logged. Last the temporary variable that holds the previous detector activation is overwritten with the start of green. This has to be done because the detector events are only analyzed if the light is green. Therefore, if the detector activates during the red phase, which is very likely, the program will recognize the detector as on immediately, at the same time as the beginning of green. If the detector is not active when the light turns green then this temporary variable will be overwritten with the actual activation time before the time the detector is on is calculated. There are also a few steps that are taken if the event is a red light. First the green light Boolean is set to false. Then, the green time is calculated and recorded, and if the detectors are active the time it has been active is calculated and added to the sum. Similar to storing the start of green in the temporary detector, the end of the green must turn off any detectors and log their time towards the sum. When the event is not either of these two then the program checks to see if the light is currently green. If it is green then it will process the events accordingly. This is done because the program is not concerned with detector events while the light is red. The program then loops through the detectors to see which detector had a state change. Each of the detectors' states is stored in a list. If any detector is active, then the green is still being used and will be counted towards the utilized part of the phase's green time. When the phase's detection status changes states from off to on, the size of the gap is calculated and the activation is logged. When the phase's detection status changes from on to off the duration of the activation is calculated and reported. Once all the events have been analyzed, the performance measures are calculated and output to Excel.

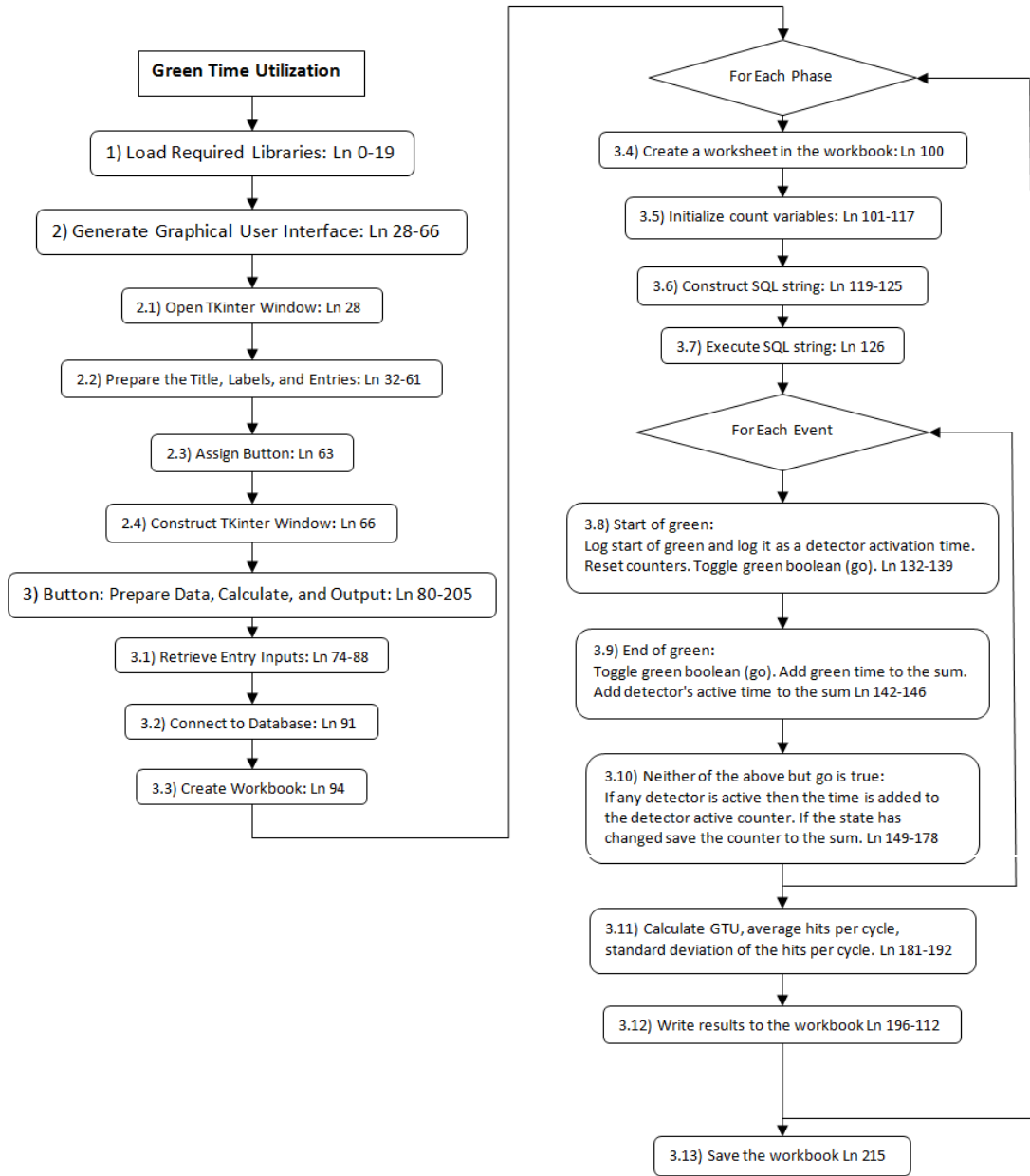


Figure 2.9 Green Time Utilization Flowchart

2.3.5 Phase Termination Analysis

2.3.5.1 Introduction

This program looks at the end-of-green events and uses the event code to prepare an excel file. The gap-outs are separated from the max-outs and force-offs and both can be easily added to a graph for analysis.

2.3.5.2 Interface

The interface, shown below in Figure 2.10, is very similar to the GTU interface. It does not, however, require detector inputs of any kind, because it only gathers reasons for phase termination. Also the phase entry data are formatted similar to the detector input of the previous program. There can be multiple phase entries for each intersection entry. Phase entries for the same intersection are comma delimited and phase groups for each intersection are separated by a semicolon, see Figure 2.11. For instance, to specify phase termination information for all eight phases of Intersection 22, the following settings would be the entry for the phases:

2;1,2,3,4,5,6,7,8;2.

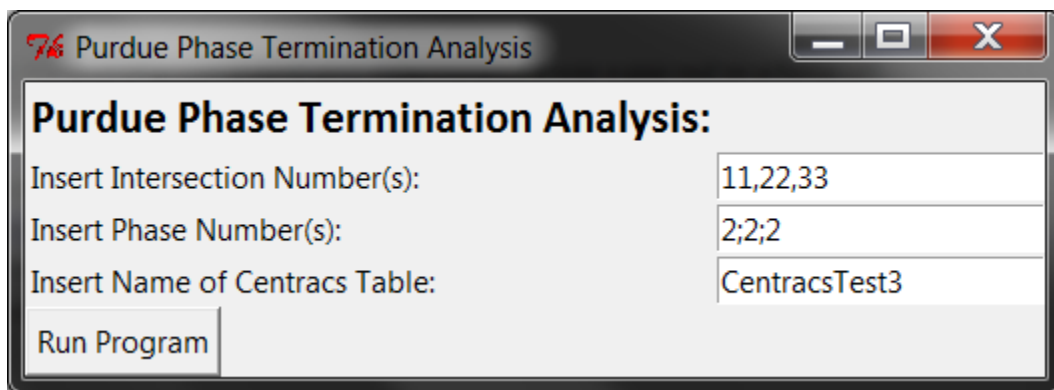


Figure 2.10 Phase Termination Interface

2.3.5.3 *Flow Chart*

This program has very simple calculations and, as shown in the flowchart used for illustration, it is therefore significantly shorter than the others (see Figure 2.11). As with each of the programs, the first task is to load the required libraries. Then the program constructs the graphical user interface. When the “Run Program” button is pressed, the values of the entry fields are retrieved and used to create two queries. The queries collect the end of green times and gap-outs in one list and force-offs/max-outs in the other. For isolated intersection operations, force-offs should be considered max-outs. The lists retrieved from the queries are written to an Excel workbook storing the output data.

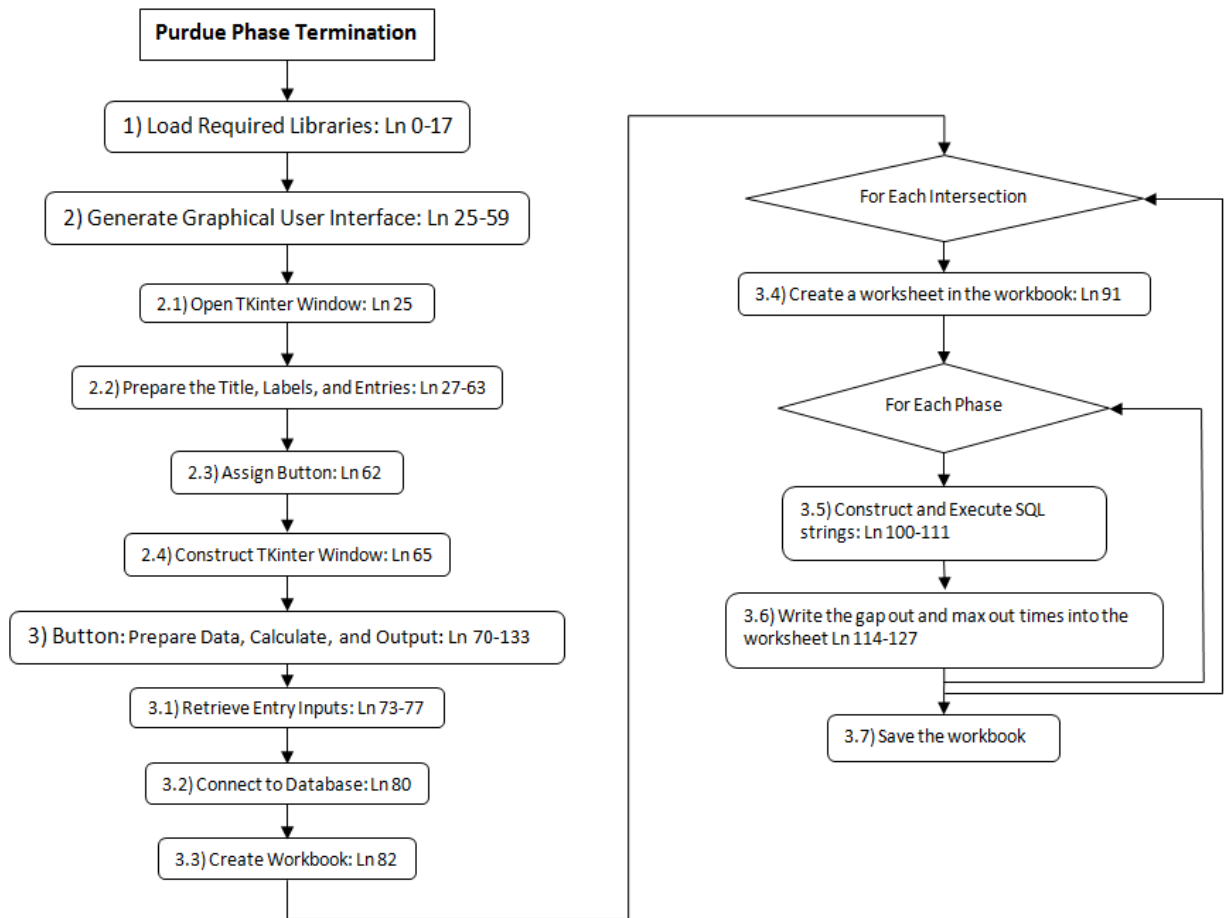


Figure 2.11 Phase Termination Flowchart

2.3.6 Delay/Queue Length

2.3.6.1 Introduction

This program estimates delay and queue length using high resolution data. The estimates are calculated using arrival and departure counts based on detector activations. This program requires lane-by-lane detection for both the arrival detection and the departure detection. Ideally, arrival detection would be accomplished by advanced detectors located upstream of the queues and departure detection would occur by stop bar detectors.

2.3.6.2 *Interface*

Similar to the previously discussed programs, this interface has entry fields for the intersections, phases, arrival detectors, departure detectors, and the table name of the dataset (see Figure 2.12 below). The first notable difference is that there are two sets of detector inputs. This is because it is important to differentiate between the arrival detection and departure detection in the analysis. The intersection field requires that the intersection values be separated by commas. Similarly, the phase numbers must be separated by commas and must have the same number of entries as intersections. The detectors are broken into groups by semicolons which each refer to a phase. The detectors for a given phase are separated by commas and are separated from detectors associated with other phases by a semi-colon. All of the information needed for delay at phase 2 of intersection 11 is indicated by dashed boxes in Figure 2.13. Queue length and delay could be extracted for other phases of the same intersection by repeating the intersection number, adding a different phase number, specifying the corresponding arrival and departure detector numbers.

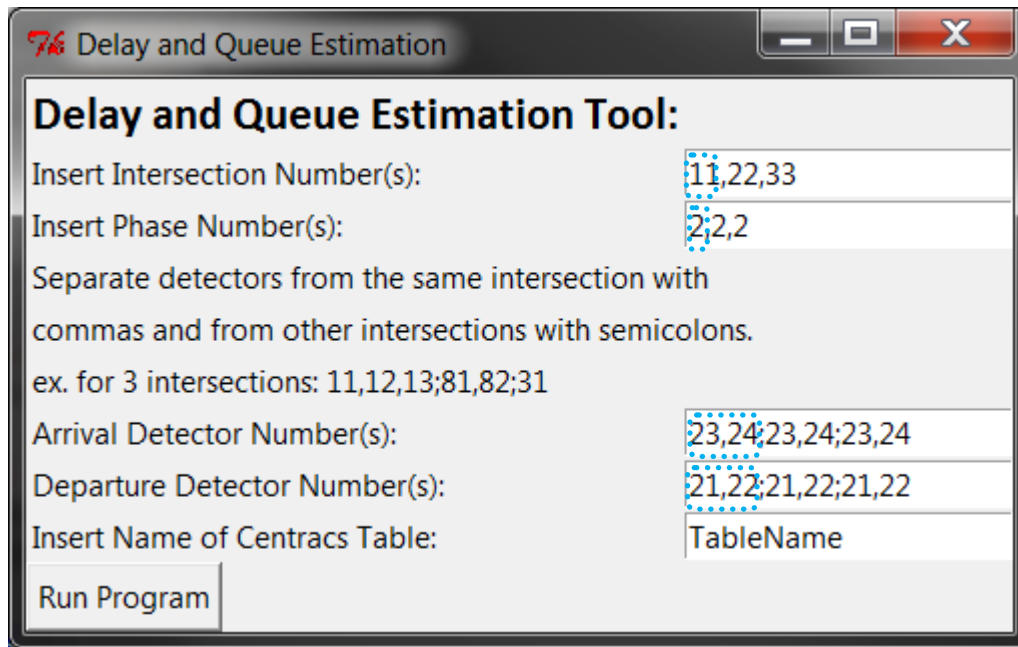


Figure 2.12 Delay and Queue Estimation Interface

2.3.6.3 Flow Chart Narration

The flow chart for this program is shown below in Figure 2.13. This program begins by loading the libraries it uses. Then it produces the graphical user interface. When the “Run Program” button is pressed it executes the function. This function starts by retrieving the information in the entry fields. Then it opens a connection to the database, opens a workbook object, and begins looping through each of the intersection/phase combinations. For each combination, it creates a worksheet, retrieves the red times, retrieves the arrival times, and retrieves the departure times. Then it loops through each time step and checks for when different events occur (beginning of red, arrivals, and departures occur). If the event is the beginning of red then the queue length is recorded. If an arrival occurs then the queue is incremented up one. If a departure occurs, the queue is decremented down one. For each time step, the time, queue, and delay are recorded. Then the delay for the next time step is calculated. The loops are complete after they have finished cycling through the lists.

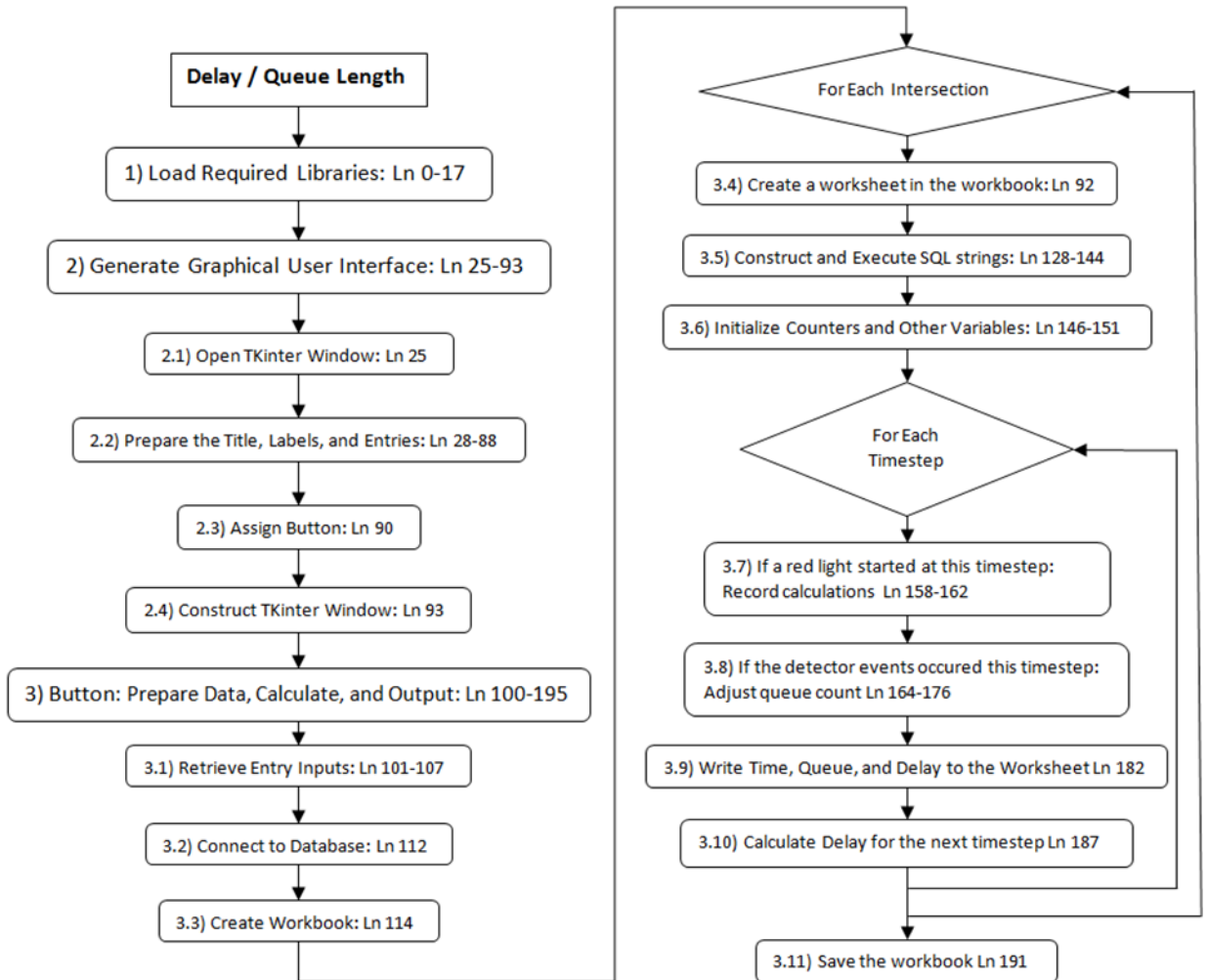


Figure 2.13 Delay/Queue Length Flowchart

2.4 Testing

2.4.1 Overview

The output from the performance measure calculation tool was tested by directly comparing the measures produced by the tool with the hand calculated measures. All of the measures relied on the same VISSIM output files. The hand calculated measures, GTU, PCD,

and PPTC, were produced using the detector output files (.ldp) and signal changes output file (.lsa) from VISSIM. The delay was compared to the calculated delay from VISSIM. Each measure was tested for accuracy over five cycles. Each of the tested measures was an exact match with the exception of the delay and queue length measures.

2.4.2 Purdue Coordination Diagram

The part of the tool that produced the PCD was tested by comparing the list of arrivals that were output to the detector output files. A PCD assembled in Excel using the detector output files was created to compare as well. These PCD's are shown below as Figure 2.14 and Figure 2.15.

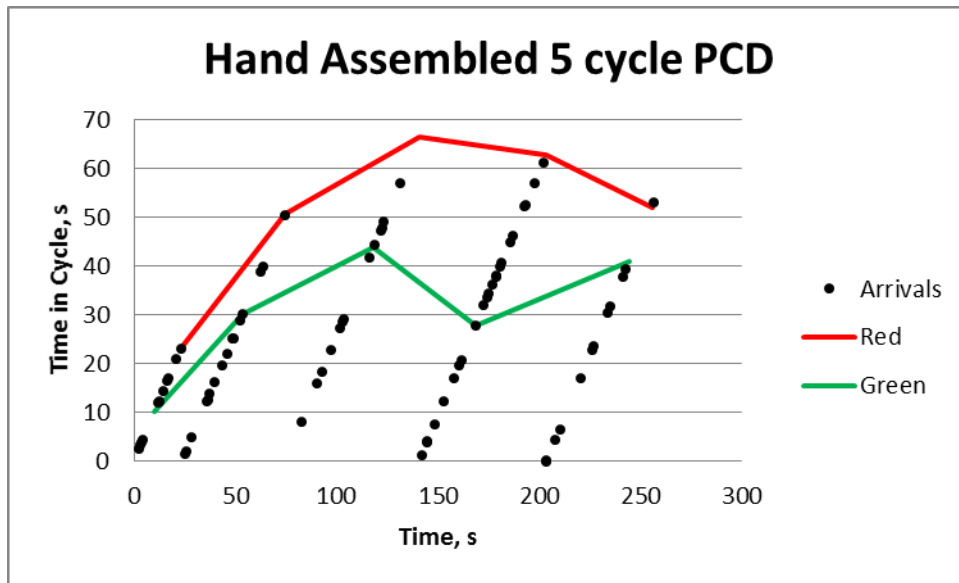


Figure 2.14 Hand Assembled PCD

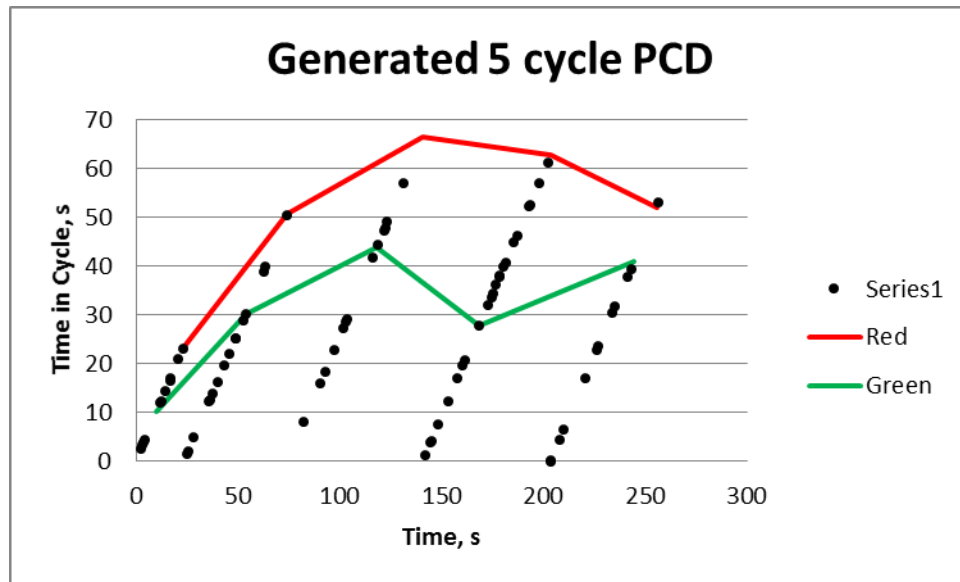


Figure 2.15 Tool Generated PCD

The arrivals are identical between the two methods.

2.4.3 Green Time Utilization

The GTU performance measurement creation was the most difficult to calculate. The complications during calculation were primarily caused by detectors being active when the phase began. When a detector activated and deactivated during the phase the time that it was active is added to the total active time and is easily accounted for. When a detector is already active, at the start of the phase, the program wouldn't know it was active until the detector deactivated.

The table below shows the GTU that was produced by the program for two phases. The column titled "GTU 1" is the calculated GTU for the first five cycles of phase 1 in intersection 1. The values in the column to its right, labeled "Calculated" contain the hand calculated values. These values were identical to the ones produced by the program. The column "GTU 6" and its

corresponding “Calculated” column were another test which included a phase with multiple lanes. The second test also produced identical results.

Table 2.1 Comparison of Hand Calculated GTU to Tool-Calculated GTU

Cycle	GTU 1	Calculated	GTU 6	Calculated
1	0.27	0.27	0.21	0.21
2	0.33	0.33	0.32	0.32
3	0.14	0.14	0.22	0.22
4	0.34	0.34	0.24	0.24
5	0.36	0.36	0.38	0.38

2.4.4 Queue Length and Delay Estimation

Along with the other measures, queue length can be estimated using high-resolution data. This measure requires both advanced and stop bar detectors that are sensitive enough to detect individual vehicles. The queue length estimation is based on an input output model which assumes that every vehicle is detected, both entering and exiting the system.

To test the queue length estimation, VISSIM Data Collection Points (VDCP) were added to the system to act as the ground truth to check the controller detectors and the resulting queue length and delay measures. Unlike the controller detectors, these VDCPs are for collecting simulation data and not for signal actuation.

During an hour long simulation, manual inspection found 15 detection errors in the controller detector data. The errors were caused by vehicles changing lanes over the advance detectors. These errors lead to a vehicle being counted twice as it enters the system. This creates a bias in the delay estimate, because vehicles are not double counted at the stop bar.

Accuracy was determined by calculating the delay with both the VDCP and the controller detectors, with the VDCP delay acting as the ground truth. The results of the delay were assessed

through visual inspection of the delay curves (see Figure 2.16), which show that the delay was accurate during the beginning of the simulation. However, delay became increasingly inaccurate through accumulated errors from double counting at the entry detectors. Delay errors became apparent after five cycles. Figure 2.16, below, shows the increasing errors in estimation as simulation progressed. The initial high resolution data estimation is the line that continues to climb upward. In comparison, the ground truth calculation and the adjusted calculation, which is explained in the next paragraph, are steady near the 20 seconds/vehicle mark. Adjustments needed to be made to improve the calculations.

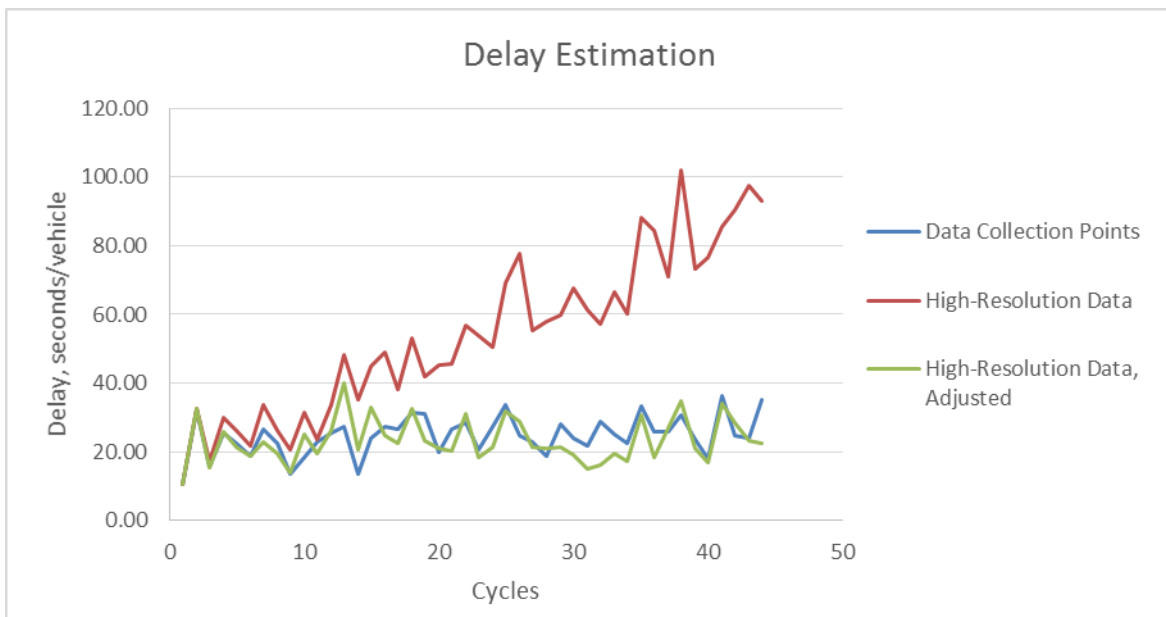


Figure 2.16 Cyclic delay of the different methods

Since the cause for the errors was known, steps were taken to remove these imperfections from the dataset and retest the method with filtered controller detector data. Although this filtering was manually implemented, it does illustrate the potential accuracy this approach has.

A t-test determined the delay estimation method to be accurate when comparing the entire hour of data, resulting in a t-statistic of .096. Even with the statistical confirmation it

should be noted that the queue length estimation works most accurately with smaller datasets, especially when detectors are found to systematically count high or low. Figure 2.17 and Figure 2.18 below show the resulting cumulative vehicle diagrams for the ground truth data and the fixed high-resolution data. It is clear that the calculation starts out working well but as the slight inaccuracies of high-resolution data continue the accuracy decreases. Even in a plot that only shows the first five minutes of the simulation, the high resolution estimations do not match the ground truth cumulative vehicles diagram.

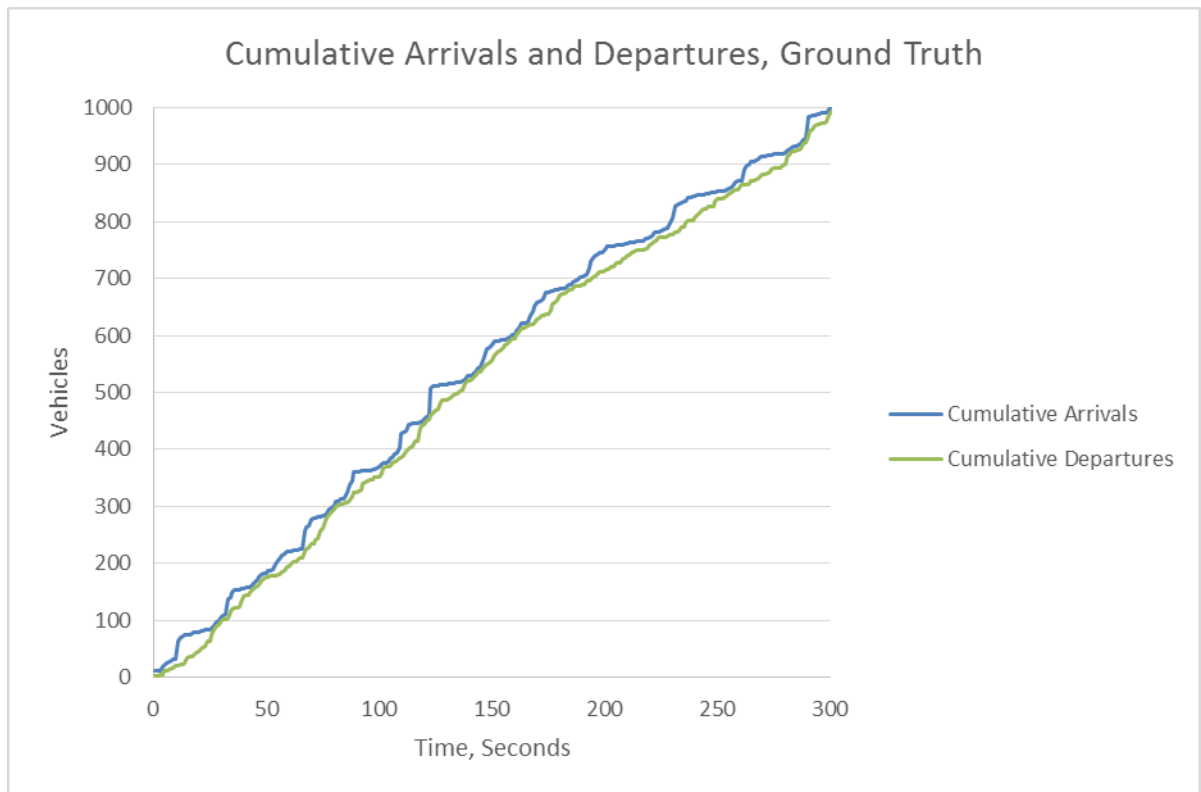


Figure 2.17 The ground truth calculation of the cumulative arrivals and departures for the first 300 seconds of the simulation

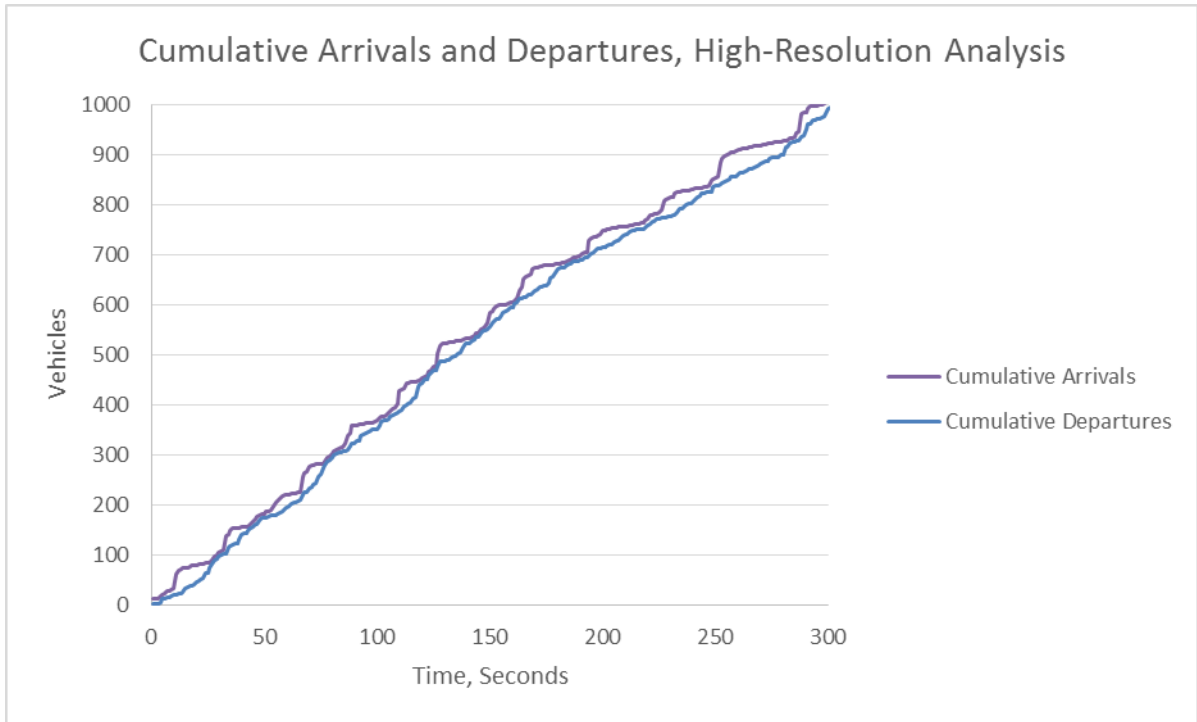


Figure 2.18 The high-resolution data calculation of the cumulative arrivals and departures for the first 300 seconds of the simulation

While the delay performance measure has potential, it needs improvement. Employing a double counting filtering algorithm would reduce delay calculation bias. In addition, instituting a detector bias measurement processes that assesses whether or not a bias exists would help as well. Bringing in other measures whose relation with delay is well known would make this last step possible. For example, knowing whether or not a queue existed at the end of a phase by way of variations in detector occupancy would eliminate additive count errors from one cycle to the next.

2.4.5 Split Failure Analysis

This is the most basic of the performance measures generated by the tool. The table below validates the results generated by the tool with hand checked phase terminations. Figure

2.19 below, shows the resulting chart from the validated tool results. The calculated measures were determined using the signal changes output file.

Table 2.2 Hand Calculated and Program Results for Creating the Purdue Phase Termination Charts

Calculated Results		Program Results	
Time Stamp	Phase	Time Stamp	Phase
Gap-out		Gap-out	
5.2	1	5.2	1
109	1	109	1
235.7	1	235.7	1
Max-out		Max-out	
74.4	1	74.4	1
172	1	172	1
Gap-out		Gap-out	
23.7	6	23.7	6
74.4	6	74.4	6
140.8	6	140.8	6
255.7	6	255.7	6
Max-out		Max-out	
203.6	6	203.6	6

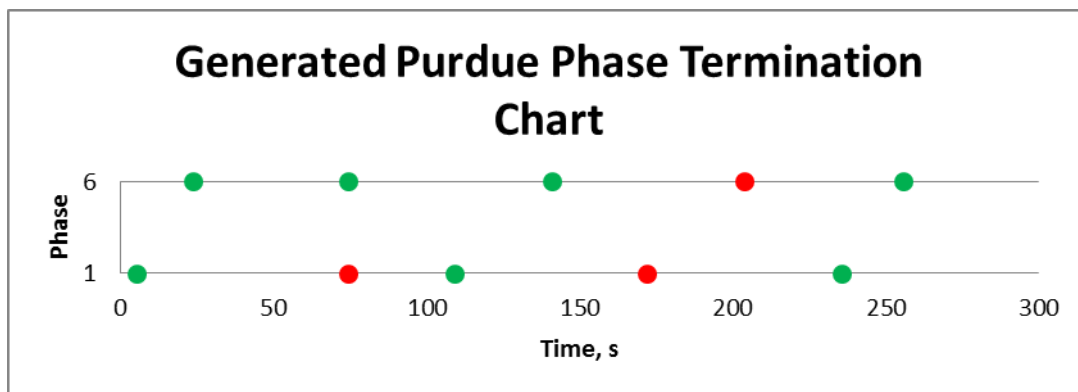


Figure 2.19 Purdue Phase Termination Chart

2.5 Conclusions

This project has demonstrated and utilized the strengths of high resolution data. It has also shown that the data can be very limiting. The Purdue Coordination Diagram is very informative, because the arrivals were accurately counted using the advance detectors and the Green Time Utilization. Several other measures were shown with the Purdue Coordination Diagram, the percent green time, percent arrivals on green, and platoon ratio. The Phase Termination Chart was included in the project as well. Delay and Queue Length estimation were also tested, showing delay's potential to be measured with high resolution data, but more detector error filtering is needed.

A tool that encompassed all these performance measures into a single analysis package was created using Python 2.7. In addition to calculating these measures it could create high resolution tables in a database using VISSIM output files. This requires a little setup but is a valuable alternative to using field data or hardware in the loop simulation with controllers that are linked to a system that produces a high resolution database.

The processes of emulating the high-resolution data and calculating the performance measures were described in detail with flow charts. Recommendations and instructions were included for future development of the tool. Finally the tool was tested by comparing the results it calculated with hand calculated measures and VISSIM's calculated measures.

By accomplishing the research objectives, researchers will be able to produce high-resolution data easily and efficiently. This will help accelerate the development of improved performance measures, a valuable development given that this high resolution data is beginning to be collected by some state agencies.

CHAPTER 3.0 USING ORIGIN-DESTINATION CENTRALITY TO ESTIMATE DIRECTIONAL BICYCLE VOLUMES

3.1 Introduction

Traditional methods to estimate bicycle volumes can be categorized as multi-step travel demand models or as direct demand models (Suhrbier and Schwartz 1999). Models in the first group attempt to forecast an elaborate combination of travel choices across large transportation networks. For example, the ubiquitous “four step model” is a sophisticated attempt to estimate four complex aspects of travel behavior: *trip generation*, *trip distribution*, *mode choice*, and *route choice*. Trip generation tries to predict the number of trips originating from an entire analysis zone for a particular purpose and time of day, such as the morning work commute. Trip distribution is a prediction of the destination for each trip. Mode choice is an attempt to predict the mode of travel that will be used and route choice tries to predict the specific facilities (i.e. street segments, intersections, shared-use trails, etc.) that will be used to arrive at the destination.

Liu et al. (2012) reviewed many of the shortcomings of using multi-step demand models to estimate bicycle volumes and offer various suggestions for future research. One major criticism is that in practice most multi-step demand models aggregate travel over large analysis zones, rather than modeling individual trips and trip-chains; for estimating car volumes, aggregate modeling is often acceptable, but for estimating bicycle travel this approach is not sufficiently fine grained. Consequently, considerable effort has sought to improve multi-step demand modeling with disaggregate, activity based techniques (Wang et al. 2011) Other efforts have sought to customize the 4-step model with additional steps related to bicycle travel, feedback loops, or more bicycle-specific calibration (Replogle et al. 1995; Schwartz et al. 1999;

Eash et al. 1999; Hudson et al. 2010). Despite the many advances, multi-step demand models continue to be data intensive, expensive, and complex.

Direct demand models bypass the behavioral context of travel by simply estimating the expected volume on a particular bicycle facility as a function of the facility's attributes. Linear regression is the most common form of direct demand modeling, such that

$$\text{Facility Bicycle Volume} = \beta_0 + \beta_1 x_1 + \dots + \beta_m x_m \quad (3.1)$$

where the explanatory variables, x_1, \dots, x_m , represent characteristics of the bicycle facility and the regression coefficients $\beta_0, \beta_1, \dots, \beta_m$ are derived from observed data. For example, Griswold et al. (2011) counted the total number of bicyclists passing through 81 intersections in Alameda County, CA and developed a linear regression model with 5 explanatory variables: (1) number of commercial properties within 0.1 miles, (2) the presence of bicycle markings on approach, (3) the natural log of network distance to UC Berkeley Campus, (4) the average slope of terrain within 0.5 miles, and (5) the connected-node ratio within 0.5 miles of the intersection. Their model predicted total bicycle counts passing through the intersections with an R^2 of 0.60. Jones et al. looked at using more than 30 explanatory variables to predict total bicycle counts on off-street bike paths. Their best model ($R^2 = 0.47$) included 3 explanatory variables: (1) total footage of off-street paths within 0.5 mile, (2) employment density within 0.25 mile, and (3) population density within 0.25 mile.

Direct demand models, including the two examples just cited, are usually designed to estimate total *non-directional* counts. In other words, they can estimate the number of bicyclists

entering an intersection, but not the actual turn movements or even the direction of travel.

Another drawback of direct demand models is that often some or all the explanatory variables are locality-specific, such as distance to the local university or distance to a particular subway station, so the models are not transferable to other communities. Likewise, for some communities certain explanatory variables might exhibit very little variation across the whole community and therefore result in poor estimation power. For example, “population density within 0.25 mile” does not significantly vary from facility to facility for many communities, so it is a useless explanatory variable.

On one hand, direct demand models are advantageous because they simplify the complexities of travel behavior, but on the other hand this makes it more difficult to attain a rich understanding of travel patterns. For example, multi-step demand models can predict not just total counts at an intersection, but also every expected turn movement through the intersection. This is because multi-step demand models rely on information about the entire street network and the interaction between origins and destinations.

This paper introduces a new method to estimate bicycle volumes that combines the strengths of multi-step demand modeling and direct demand modeling. The method uses network analysis to quantify travel patterns between origins and destinations through a new metric that we call *OD centrality*. The metric is then used as an explanatory variable in a direct demand model which we programed as a tool for geographic information systems (GIS) software. The method is presented through a case study using data collected as part of the National Bicycle and Pedestrian Documentation Project (NBPDP); however, data collected through any manner, including automated counters, could be used (NBPDP 2008).

This paper is useful for practitioners looking for simple and ready to use tools to estimate bicycle volumes and beneficial for researchers studying how urban form influences bicycle travel. Researchers may find it useful to integrate aspects of the new method into traditional methods.

3.2 Centrality

In graph theory, the importance of a link or node in a network can be quantified through various measures of centrality. The most common forms of centrality are: closeness centrality, which measures how close a link is to all other links; degree centrality, which measures how many nodes are connected to a link; and betweenness centrality, which measures the proportion of shortest paths that pass through a link. There are many other forms of centrality, including alpha centrality, load centrality, stress centrality, straightness centrality, Katz centrality, and eigenvector centrality (Brandes 2008).

Centrality measures can be formulated for links or for nodes. For example, stress centrality, which we modify in the next section, is formulated for link e in a network as

$$\text{Stress Centrality}_e = \sum_{i,j \in V} \sigma_{ij}(e) \quad (3.2)$$

where

V = set of all nodes in the network,

σ_{ij} = shortest path from node i to node j , and

$$\sigma_{ij}(e) = \begin{cases} 1, & \text{if link } e \text{ is used in } \sigma_{ij} \\ 0, & \text{otherwise} \end{cases}.$$

Shimbel (1953) introduced stress centrality in 1953 to analyze communication networks, but in the context of a transportation network, stress centrality represents the number of times a street would be used if someone were to travel from every node to every other node.

Researchers have demonstrated the usefulness of the many different centrality metrics in a variety of transportation planning applications; for example, to improve airline networks (Liu et al. 2011), design public transportation systems (Derrible 2012), and characterize traffic analysis zones (Zhang et al. 2011). In the mid-1980s a group of architects and urban theorists developed a specialized approach to analyzing centrality called “space syntax” with the intent to quantify the connectivity of hallways and rooms in a building (Hillier & Hanson 1984; Jiang & Claramunt 2004). Space syntax was eventually extended to larger, outdoor urban environments. One space syntax measure called integration has shown high correlation with vehicle and pedestrian volumes (Hillier et al. 1993; Penn et al. 1998; Raford & Ragland 2004). Raford et al. (2007) used space syntax to explain how bicyclists choose different routes through London. McCahill and Garrick (2008) explored using space syntax in a direct demand model to estimate bicycle volumes. They calibrated the model with aggregate bicycle counts at 16 intersections in Cambridge, Massachusetts, but unfortunately concluded that, on its own, space syntax was not effective for explaining the observed volumes ($R^2 = 0.16$).

3.3 Method

3.3.1 OD Centrality

We modify stress centrality in three ways to create a new metric, which we call *origin-destination (OD) centrality*. First, we define σ_{ij}^* as the preferred bicycle path between locations i and j using current research on bicycle route choice. Second, we only consider a specific subset of origin-destination pairs that can be reasonably reached by bicycle. Third, we augment the calculation with origin and destination multipliers that represent a magnitude of “trip potential” between OD pairs. The new formulation for a link e in a network is

$$OD\ centrality_e = \sum_{i \in I, j \in J \mid d_{ij} \leq \delta} \sigma_{ij}^*(e) M_i M_j \quad (3.3)$$

where

I = subset of origins in the network,

J = subset of destinations in the network,

σ_{ij}^* = preferred bicycle path from node i to node j ,

M_i = multiplier for origin i ,

M_j = multiplier for destination j ,

d_{ij} = distance between origin i and destination j , and

δ = reachable distance threshold for bicycles.

In the following sections, we describe these innovations and introduce the GIS tool we created to calculate OD centrality. We also explain how OD centrality can be used in a direct demand model to estimate and spatially interpolate bicycle volumes throughout a community.

3.3.2 Preferred Bicycle Paths

The preferred path between two points on a network can be determined in various ways, such as the fewest number of links, shortest geographic distance, or shortest travel time. A common approach is to define an impedance (or generalized cost) for every link and node, such that the “shortest” path is the path that minimizes total impedance (cost). Route choice algorithms, such as those used by Google and MapQuest, analyze the impedance of potential paths to identify the preferred path. If a bicyclist has complete information (i.e. is aware of the impedance for all route options, even subconsciously), then according to rationale choice theory, the bicyclist will choose the route with the lowest impedance (Stinson & Bhat 2003; Lee & El-Geneidy 2011).

In recent years, the ability to place GPS trackers on bicyclists has allowed researchers to observe the relative attractiveness of different facility types, or in other words, to quantify the impedance associated with different facility characteristics (Broach et al. 2012, Harvey et al. 2008, Larsen & El-Geneidy 2011; Menghini et al. 2010) The main impedance factor for bicyclists is distance; for this reason it is common to quantify other characteristics in terms of added distance. For example, Broach et al. (2012) found that, for a commute trip, bicyclists are willing to travel about 4 tenths of a mile extra to avoid a one mile slope that is between 2% and 4%, 1.2 miles extra to avoid a one mile slope that is between 4% and 6%, and 2.2 miles extra to avoid a one mile slope that is greater than 6%.

For the case study, we defined two types of impedance: *link impedance* for traversing a street segment and *node impedance* for passing through an intersection. Link impedance is based on the street segment's length, friction, and slope. Friction represents the deterrence/attraction for bicyclists to use a particular street segment and can be determined according to various attributes, such as vehicle traffic volumes, presence of a bike lane, or beautiful vistas. In the case study, we used the Highway Capacity Manual's bicycle level of service (BLOS) to define friction, f , (HCP 2010; Lowry et al. 2012; Callister & Lowry 2013) The BLOS calculation involves ten attributes, which include vehicle volumes, vehicle speeds, and shoulder width, to produce a letter grade A through F and a corresponding numeric score between 1.00 and 5.75. For slope impedance, we used the findings from Broach et al. (2012). We combined length, friction factor, and slope factor as follows:

$$C_{link} = L * F_f * F_s \quad (3.4)$$

with

$$F_f = f^{0.5} \quad (3.5)$$

$$F_s = \begin{cases} 1.0, & \text{if slope} < 2\% \\ 1.4, & \text{if } 2\% \leq \text{slope} < 4\% \\ 2.2, & \text{if } 4\% \leq \text{slope} < 6\% \\ 4.2, & \text{if } 6\% \geq \text{slope} \end{cases} \quad (3.6)$$

where

C_{link} = link impedance (cost),

L = length for the street segment,

F_f = friction factor for the street segment,

F_s = slope factor for the street segment, and

f = friction for the street segment.

Node impedance is calculated based on turn angle (relative bearing, 0-180 degrees), the type of intersection control and the functional class of the cross street. Thus, for every turn movement k the calculation is:

$$C_{node,k} = F_t + F_c \quad (3.7)$$

with

$$F_t = \begin{cases} 0.5 * \alpha, & \text{if right-hand turn} \\ 1.2 * \alpha, & \text{if left-hand turn} \\ 1.0 * \alpha, & \text{if through} \end{cases} \quad (3.8)$$

where

$C_{node,k}$ = node impedance (cost) for turn movement k ,

F_t = turn impedance factor for turn movement k ,

α = turn angle (relative bearing) for turn movement k , and

F_c = intersection control and cross street impedance factor (see

Table 3.1).

Table 3.1 Intersection Control and Cross Street Impedance Factor, F_c

Intersection Control	Cross-Street			
	Local	Collector	Minor Arterial	Primary Arterial
No Control	20	40	90	120
Stop Sign	25	20	50	80
Signal	30	20	20	20

Note: Factors are unitless.

The node impedance factors are adapted from Broach et al. (2012). Their research did not report functional class of the cross-street, but rather the cross-street’s Annual Average Daily Traffic (AADT). Note that the factors for link impedance are multiplicative (Equation 3.4) and for node impedance the factors are additive (Equation 3.7). This is because the characteristics of a street segment act over a distance while the characteristics at an intersection act at a single point.

3.3.3 Origin-Destination Pairs

Two steps determine the origin-destination pairs for the analysis. First, the analyst must specify certain locations as “origins” and/or “destinations.” For the case study, since bicycle counts were observed in the morning and in the evening, we simply specified residential parcels as origins and non-residential parcels as destinations to analyze the morning data and vice-versa to analyze the evening data. Other, more sophisticated specifications could be explored. For example, some cursory investigation showed improved model fit for the evening analysis when

non-residential parcels were specified as origins *and also* as destinations. This might be explained by the fact that the morning commute is often more uniform—people travelling from home to work—while the evening commute is more diffuse—people not just travelling straight home, but to a myriad of destinations. An investigation of different OD specifications could be part of the calibration process, wherein the model is run and re-run to find the best model fit. For the case study, a straightforward specification was used for the purpose of simply introducing this new method.

The second step to determine origin-destination pairs is achieved automatically by the GIS tool. The tool limits origin-destination pairs to those that can be reasonably reached by bicycle based on a “reachable distance threshold,” which is designated as δ in Equation 3.3. This parameter is included to recognize the reality that most people do not use their bicycle for long-distance utilitarian travel (Landis 1996). The δ parameter could also be investigated by the analyst as a calibration parameter, but we chose to hardcode it into the GIS tool based on results from our sensitivity analysis and previous research about bicycle travel. Figure 3.1 shows the results from the sensitivity analysis. The horizontal axis represents different values of δ . High values of δ allow more origin-destination pairs, while low values of δ signify a tighter restriction on the pairs to be included in the calculation of OD centrality. Figure 3.1 shows that the best correlation occurs at about $\delta = 1.5$ miles for the bicycle data and $\delta = 0.8$ miles for the pedestrian data. This corresponds well with previous research suggesting most trips are no more than 1.5 miles for bicyclists and 0.5 miles for pedestrians (Turner et al. 1997).

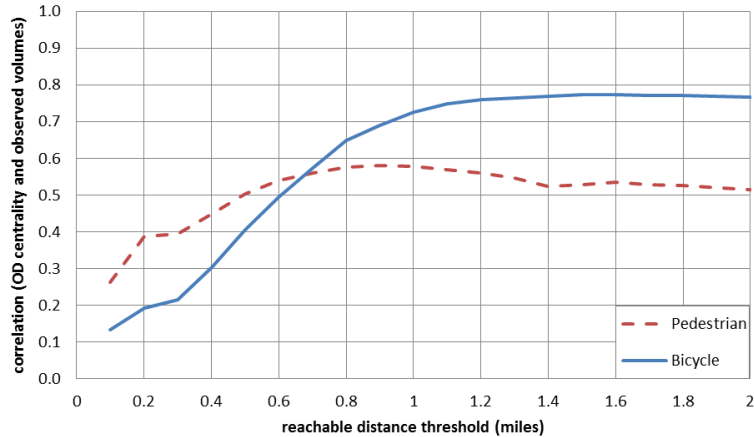


Figure 3.1 Correlation between OD centrality and observed volumes for different reachable distance thresholds

3.3.4 OD Multipliers

The other innovation we introduced into OD centrality is the inclusion of origin and destination multipliers, M_i and M_j , respectively. The multipliers represent relative “trip potential” leaving or going to each node. For example, a large shopping mall is more likely to attract trips than a small coffee shop; likewise, an apartment building is going to generate more trips than a single family home.

The analyst who is calculating OD centrality needs to specify the multiplier associated with the origins and destinations. However, it is important to note that unlike *trip generation* in traditional multi-step demand models, the intent here is *not* to predict the number trips produced and attracted, but rather to merely specify the relative magnitude of trip potential across different locations. For the case study, the number of dwelling units was used as the multiplier for residential parcels and for all other land use types the multiplier was specified according to

square footage. Another approach might be to define the multipliers based on the *ITE Trip Generation Manual* (TGP 2012).

3.3.5 GIS Toolbox

A GIS toolbox was created to calculate OD centrality and execute a direct demand model. The tools were written in modifiable open-source python code for ArcGIS® 10.1. The centrality calculation is based on Brandes's (2001) algorithm. Figure 3.2 shows the graphical user interface for the tool to estimate and spatially interpolate bicycle volumes. Other tools in the toolbox are for data preparation purposes. Each tool prompts the user for input data and the desired directory for output. The user can access help documentation that explains the required input by clicking the "Tool Help" button. The tool shown in Figure 3.1 requires four input files:

Observed Count Data – This is a folder of spreadsheet files for each intersection. The format is similar to the guidelines in the NBPD.

Origins – This is a point or polygon shapefile representing origins, such as a land use parcel file. The user is also prompted to specify the origin multiplier attribute field.

Destinations – This is a point or polygon shapefile representing destinations. This could be the same parcel file used for the origins, perhaps with different destination multipliers.

Bikeway Network – This is a polyline shapefile representing streets and shared-use paths. Most communities have a "street center line file" that can be used, but if not, ESRI provides a free street file for all of North America. The file must exhibit correct topology. The user is also prompted to specify the impedance field associated with the network (which can be calculated using a different tool in the toolbox).

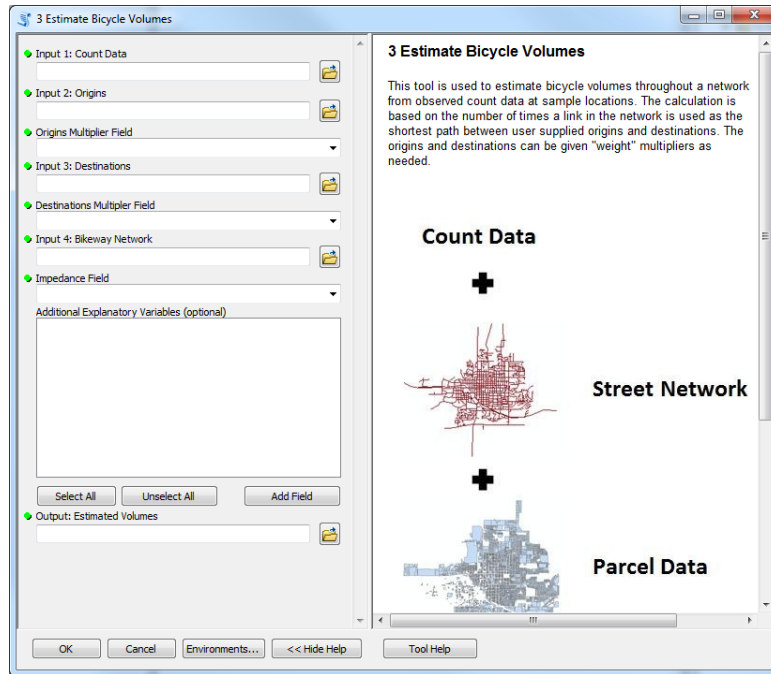


Figure 3.1 Graphical user interface for the GIS tool to estimate bicycle volumes

Once the required input is provided, the user runs the tool by clicking “Ok”. The tool calculates OD centrality, calibrates a regression model, and uses this regression model to estimate bicycle volumes throughout the network. The user can choose to include additional explanatory variables in the regression model (see Figure 3.1). The output includes a text file with a summary of the regression statistics and a polyline shapefile with predicted bicycle volumes for the entire network. For the case study community the execution time on a standard, workstation-class Lenovo laptop is about 7 minutes.

3.4 Analysis and Results

3.4.1 Case Study Data

The National Bicycle and Pedestrian Documentation Project is an increasingly popular grass-roots effort to collect data in communities around the United States (NBPDP 2013) Often the work is accomplished by citizen-volunteers who stand on an assigned street corner counting bicyclists and pedestrians for a two-hour period in the morning and evening. The City of Moscow, Idaho (population 25,000) has conducted counts each fall for the past three years. The city has high bicycle ridership because it is home to the University of Idaho and is located nine miles away from the City of Pullman, Washington (population 32,000) the home of Washington State University. Moscow's fall 2012 data were used for the case study.

The 14 intersections that were observed are located throughout the city as shown in Figure 3.1. The volunteers observed every thru and turn movement for the intersection, tallying the counts for fifteen minute intervals from 7:00 to 9:00 AM and 4:00 to 6:00 PM. With twelve movements at a standard four legged intersection and six movements at a three legged T-intersection, a total of 162 movements were observed.

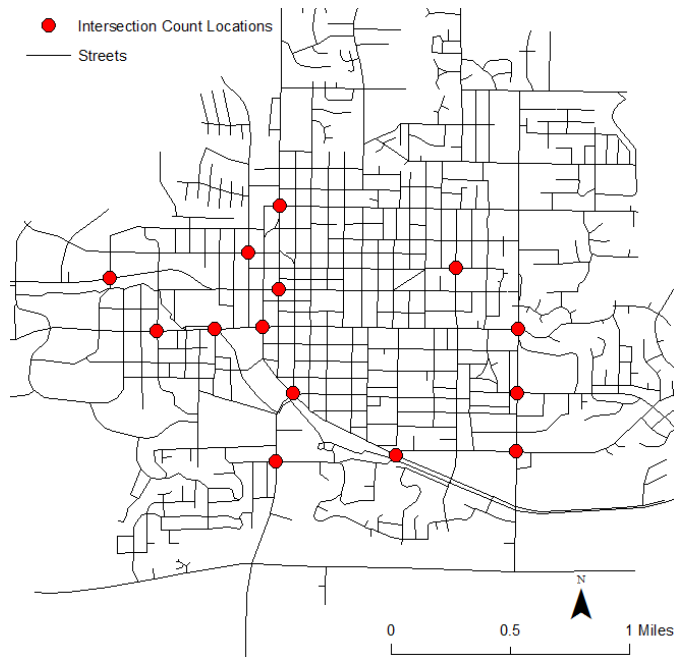


Figure 3.1 Intersection count locations

3.4.2 Model Calibration and Validation

The data were randomly split into two sub-samples; 90% for calibration, and 10% for validation. Various regression techniques that are common for direct demand modeling were investigated, including ordinary least squares regression, log-linear regression, Poisson regression, negative binomial regression, and zero-inflated negative binomial regression. Ordinary least squares regression produced the best model fit based on within-sample (calibration) and out-of-sample (validation) R^2 .

Error! Reference source not found. shows the coefficients and statistics for the morning and evening models using conventional stress centrality and OD centrality as the sole explanatory variable (in practice, additional explanatory variables could be used, but were ignored here to highlight the predictive power of OD centrality). Conventional stress centrality is not statistically significant and does not produce a useful estimation model. OD centrality, on the

other hand, is statistically significant for the AM and PM models and exhibits good R^2 values, suggesting OD centrality provides strong explanatory and predictive power. The best model was for the PM counts which explained 61% of the variability in the calibration dataset and 73% of the variation in the validation set. Presumably, these models could be significantly improved with additional explanatory variables.

Table 3.1 Estimation Coefficients and Model Statistics

Model Variable	Stress Centrality Model		OD Centrality Model	
	AM	PM	AM	PM
Constant	1.97 **	2.54**	1.54**	2.13**
Stress Centrality	0.11	0.17*	-	-
OD Centrality	-	-	0.69**	0.63**
Calibration R-squared ^a	0.04	0.10	0.45	0.61
Validation R-squared ^b	0.08	0.07	0.58	0.73

Note: Dependent variable is one hour peak bicycle count observed in the morning (AM) and evening (PM).

*Significant at 95% ($p < 0.05$); **significant at 99% ($p < 0.01$).

^a Within-sample goodness-of-fit for a random 90% of observed data, $n = 147$.

^b Out-of-sample goodness-of-fit for a random 10% of observed data, $n = 16$.

Figure 3.1 shows the difference between stress centrality and OD centrality across the case study community. Dark street segments signify higher centrality, or in other words, more incidence of being on a shortest path. This figure illustrates how the modifications of stress centrality produced better values. First, the figure shows improvement from using bicycle-specific impedances. This can be seen by the straightness of the paths in Figure 3.4b compared to the indirect, stair-step paths in Figure 3.1a. Furthermore, for stress centrality many of the shortest paths include busy streets and steep hills. Second, the figure illustrates improvement from incorporating origin and destination multipliers. Note in Figure 3.1b, high values of OD

centrality are generally located toward the southwest, which is expected because this is where the university is located.

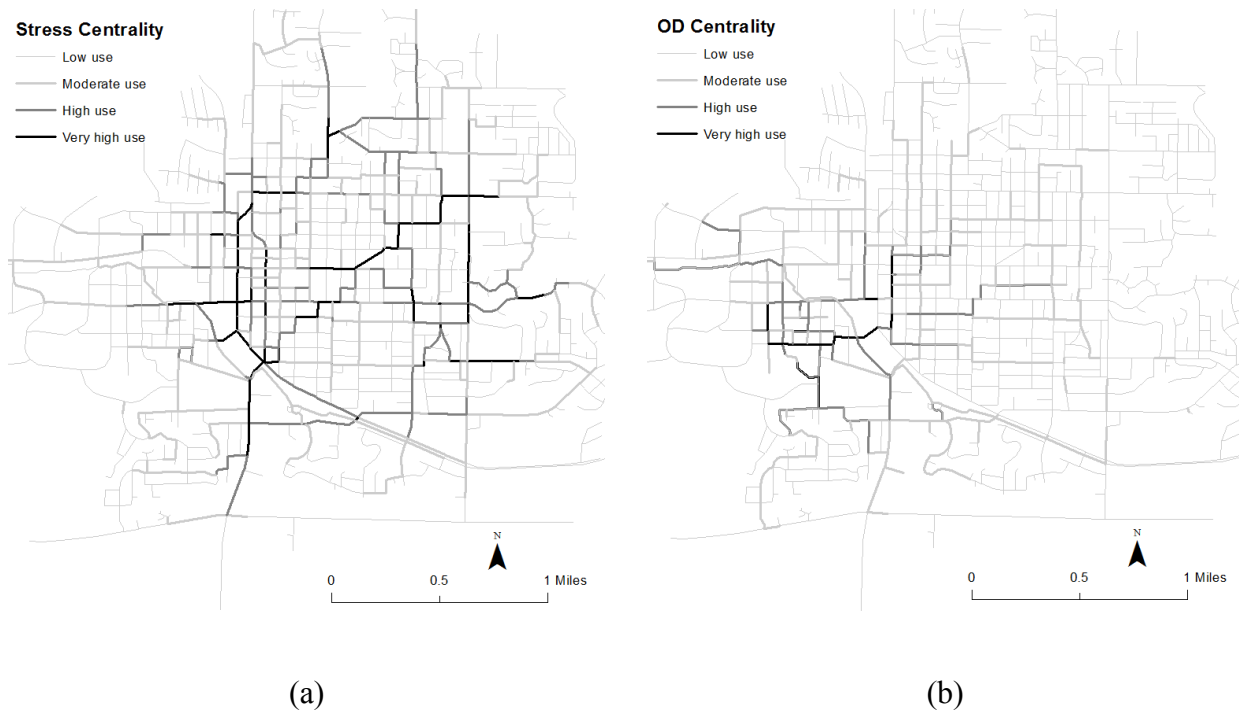
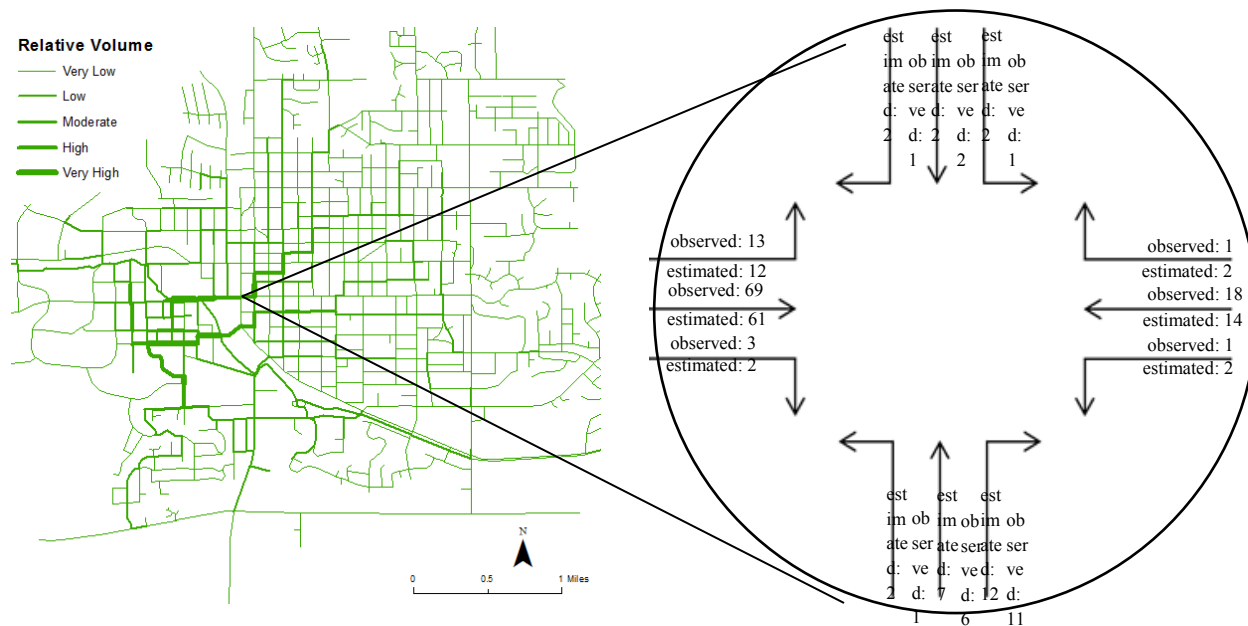


Figure 3.1 Comparison of: (a) stress centrality and (b) OD centrality

Figure 3.2 shows bicycle volume estimates for the whole city. This is the output of the GIS tool. Figure 3.2b provides detail for a selected intersection where counts were actually observed. In general, the difference between observed and estimated volumes is fairly small and magnitudes correspond very well. Engineers and planners could use maps like Figure 3.2 in a variety of ways, including prioritizing capital investments and analyzing risk exposure. Furthermore, practitioners could use the tool for scenario planning by first calibrating the model under current conditions and then running it again under different conditions. For example, if a community is proposing to build a new residential subdivision, then the tool could be used to

predict how this would change bicycle volumes throughout the community. Likewise, the tool could be used to investigate the change in travel patterns from improving network connectivity by constructing new shared-use pathways or painting new bike lanes. These types of applications would not be easily accomplished through a traditional direct demand model.



(a)

(b)

Figure 3.2 Estimated one hour peak bicycle volumes for: (a) the entire community and (b) a selected intersection

3.5 Conclusion

This paper presents a new method to estimate bicycle volumes that is advantageous over traditional multi-step demand models and direct demand models. Multi-step demand models are data intensive and require specialized skills and software. The new approach is easy to use and only requires a street network, digital elevation map, residential and non-residential parcels, and counts at select locations, all of which are readily available for most communities. This new approach integrates these data, thereby facilitating analysis of urban form and bicycle travel.

Direct demand models are also easy to use and straight forward, but do not adequately consider

network characteristics and the important spatial relationship between origins and destinations. Consequently, traditional direct demand models cannot easily model detailed information, such as directional volumes for all turn movements at an intersection. The new approach's advantages provide the opportunity for engineers and planners to use the new method for many important applications.

The method is based on a new metric called OD centrality. Future work could investigate how to improve the metric. For example, different origin and destination multipliers could be examined, perhaps from the ITE *Trip Generation Manual* (Turner et al. 1997) Another possible improvement could be to use a “distance decay function” rather than a fixed reachable distance threshold for δ (see Equation 3.3) (Iacono et al. 2010). Future research could develop guidelines for the calibration process and other best practices, including investigating additional explanatory variables that would work well alongside OD centrality.

CHAPTER 4.0 PEDESTRIAN LIVABILITY AND MICROSOFT'S KINECT

4.1 Introduction

In recent years, there has been increasing interest in active forms of transportation from a public health perspective. With a growing body of evidence supporting the health benefits of walking and biking for transportation (Pucher et al. 2010), urban planners and public officials are increasingly interested in the relationship between rates of participation in active transportation and the design of neighborhoods and other public spaces. In addition, due in part to the observed correlation between participation and access to walking and bicycling infrastructure (Dill 2009; Buehler & Pucher 2012), there is a trend toward increased public investment in new and existing facilities. As in all forms of transportation infrastructure, a great deal of pedestrian and bicycle traffic data are needed to inform the planning, design, and management of such facilities. In addition, from an academic research perspective, pedestrian data are indispensable for studies including analysis and modeling of pedestrian behavior and movement patterns, traffic safety, and signal timing.

Despite the importance of accurate pedestrian data, most data collection has relied on labor-intensive manual counting. In the case of large and widespread pedestrian data collection efforts, manual methods are expensive and often impractical. Therefore, automatic detection and tracking of pedestrians is of significance to both engineers and researchers.

Over the past decade, a number of technologies have been developed and applied to various pedestrian detection applications. Currently, the most commonly employed sensors for pedestrian detection are imaging sensors in various configurations using visible light and infrared

(IR) radiation, as well as the time-of-flight based sensors such as RADAR and LASER scanners (Gadhi & Trivedi 2007). Imaging sensors can capture a high-resolution view of the scene with very rich information, but extracting information involves a substantial amount of processing. In addition, one of the most challenging issues for imaging-based approaches is occlusion, where a pedestrian is partially obscured by another pedestrian or an environmental feature. Although a number of image processing techniques have been developed to solve this problem, the performance of most existing detectors degrades quickly as the number of pedestrians and corresponding rate of occlusion increases. Occlusion is less of an issue for time-of-flight scanners because they are based on depth information, but resolution is often limited. In this sense, these two types of sensors are complementary, and their fusion is expected to result in more robust detection.

Microsoft's Kinect device with 3D sensing capabilities provides us a new way to easily fuse the 2D image information and depth information for pedestrian detection. In this paper, we aim to utilize Kinect to develop a low cost solution for occlusion robust pedestrian detection in crowded scenes. The paper contains the following contributions: 1) Utilizing the RGB-D image from Microsoft Kinect, we present an efficient pedestrian detection approach for crowded scenes where occlusion occurs frequently. 2) More specifically, by fusing the pedestrian contour regions extracted from the RGB image with the depth information, we develop a pedestrian extraction algorithm. As our results demonstrate, this fusion presents a novel way to make pedestrian detection robust to occlusion. 3) We design a pedestrian tracking and counting algorithm based on template matching.

The remainder of this paper is structured as follows: Section 4.2 summarizes a literature review of Kinect based pedestrian detection approaches, Section 4.3 describes our proposed approach in detail, Experimental results are introduced in Section 4.4, and Section 4.5 concludes the study.

4.2 Literature Review

Since it was first introduced in November of 2010, Kinect has generated a great deal of research interest particularly in the Computer Graphics and Computer Vision communities as a low cost 3D sensor. It has been the subject of considerable research in a number of fields, including robotics, medical imaging, and natural human-computer interaction (Stowers et al. 2011; Alnowami et al. 2012; Boulos et al. 2011). Among these recent research efforts, many have focused on human detection for gaming or other purposes. Previous work on Kinect-based pedestrian detection can be generally classified into two categories: depth based and RGB-D based.

4.2.1 Depth Based Human Detection

Xia et al. (2011) proposed a model based human detection approach using depth information. It detects humans using a 2-D head contour model and a 3-D head surface model, and segment a human from his/her surroundings to extract the whole contours of the figure. It was reported that a detection accuracy of 98% was achieved in their test cases. However the frame achieved in this study was only 0.4s/frame which is hard for real-time applications. Furthermore, it failed to detect a human when his/her head is occluded, which is often the case in crowded scenes.

Hsieh et al. (2012) presented a people counting system with Kinect. They first apply morphological processing to the depth image to find the regions of interest (ROI), then determine the targets from the ROI and count the pedestrian number. It was reported that a counting accuracy of almost 100% was achieved in their tested case, but no details were released on how to deal with occlusion in their paper. In addition, the algorithm did not perform well when background still objects such as tables, shelves and pillars also appeared in depth images which occur in most real-world cases.

Charreyron et al. (2013) proposed a pedestrian data collecting system using Kinect. They used the Kinect SDK skeleton tracking information to track pedestrians and get volumes and walking speeds. The system has a pedestrian volume accuracy of 92% in low to moderate traffic conditions. However, the system cannot deal with occlusion due to the limitation of the Kinect SDK middle-ware.

Wu et al. (2011) presented a feature descriptor, Histogram of Depth Difference (HDD), for detecting pedestrian in depth images. The HDD feature descriptor can describe the depth variance in a local region in depth image. Though it was reported that the detection accuracy was about 96% for single pedestrian detection, it cannot effectively deal with occlusion.

Won et al. (2013) proposed a pedestrian detection algorithm on labeled depth data. It computes feature responses for head and legs of human body using depth and label data, and detects pedestrians by removing edges and partitioning a bipartite graph of head and leg response blobs using prior knowledge about human body. The reported detection rate was only 68.31%, and the detector fails when severe occlusion occurs.

4.2.2 RGB-D Image Based Pedestrian Detection

Luber et al. (2011) combined a multi-cue person detector for RGB-D data with an on-line detector to detect and track pedestrians. The two detectors were integrated into a decision framework with a multi-hypothesis tracker that controls on-line learning through a track interpretation feedback. For on-line learning, they took a boosting approach using three types of RGB-D features and a confidence maximization search in 3D space.

Spinello et al. (Spinello & Arras 2011) presented a people detection approach using RGB-D data. They designed a Histogram of Oriented Depths (HOD) method to detect people in dense depth data, and then proposed Combo-HOD, a RGB-D detector that probabilistically combines HOD and HOG to further detect pedestrians. While the approach is effective in scene with slight occlusion, its performance degrades increasingly in scenarios with severe occlusion.

Seera et al. (2012) used three Kinects mounted on the ceiling of a hallway to record pedestrian trajectories. They mapped the depth information from each Kinect into a common world coordinate system using a rigid transformation, and then grouped depth information from a single Kinect in the world coordinate system into individual pedestrian based on hierarchical clustering. There was no need to deal with occlusions because the data was collected from a top-down perspective. Although this approach works well, it is not feasible for many real world pedestrian detection scenarios such as outdoors or in areas with particularly low ceilings.

Bosso et al. (2013) proposed a multi-person tracking algorithm for mobile platforms equipped with a RGB-D sensor. Their approach features a point cloud depth-based clustering, an HOG-like classification to initialize a pedestrian tracker and classifier with online learning to manage the person ID matching. It was reported that the algorithm proved to correctly track 96%

of pedestrians even in case of temporary occlusion. However, it fails when occlusions exist from pedestrians entering the scene to leaving it, which occur frequently in a crowded scene.

Salas et al. (2011) presented a strategy that combines color and depth images to detect people in indoor environments. The similarity of image appearance and closeness in 3D position over time yield weights on the edges of a directed graph they partitioned into “*tracklets*”. Each “*tracklet*” was assigned the highest score that a Histograms-of-Oriented Gradients (HOG) person detector yielded. High-score tracklets were deemed to correspond to people. Like other HOG based methods, this approach is only feasible in non-occlusion or slight occlusion cases.

It is clear from the discussions above that while much work has gone into Kinect-based pedestrian detection in recent years, most has only focused on detecting pedestrian in simple scene with slight occlusion or no occlusion. Most current work has some limitations for crowded scenes, especially in resolving the occlusion issue.

4.3 Data

We evaluate our method using sequences of RGB and depth images produced by Kinect in three scenarios. In the first scenario, we recorded a sequence in our lab with at least five pedestrians present simultaneously in the scene with tables, chairs, shelves, computers, pillars, etc. The second scenario is outdoors on a sunny day, where there are at most five pedestrians simultaneously. The third is also an outdoor scene but on a cloudy day, where at most nine pedestrians are present simultaneously in the detection range.

All the pedestrians in the three scenarios have a variety of poses, and they have frequent interactions with others or the surrounding objects. There are approximately 9000 frames in each

test database and the resolutions of both RGB and depths image are 640×480 at a frame rate of 30fps. This amounts to about 5 minutes of video for each scenario.

4.4 Methodology

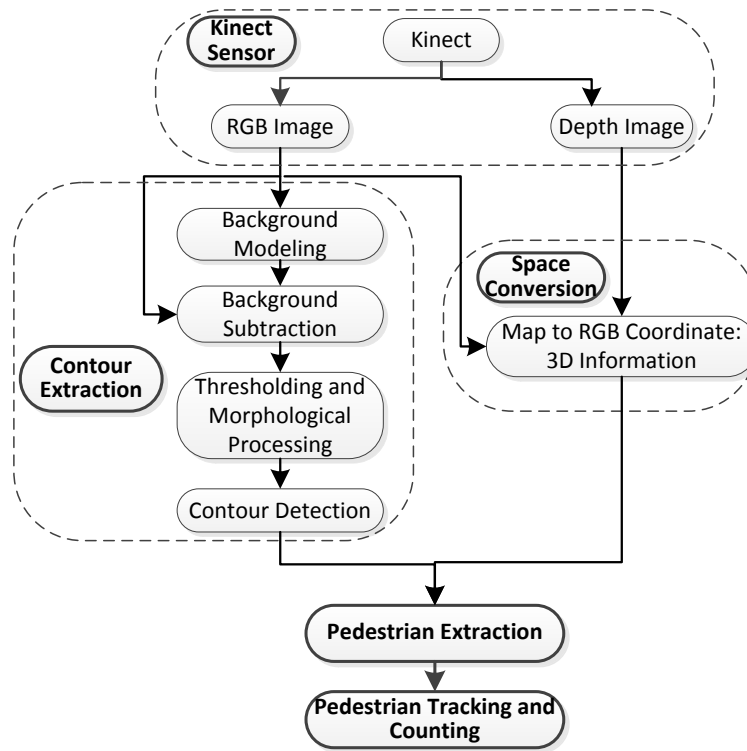


Figure 4.1 Flow chart of the proposed method

The pedestrian detection method proposed in this paper contains four principle steps. The first step is to acquire both the RGB image and Depth image through Microsoft Kinect device. The second step is to extract the pedestrian contours from the RGB image by using traditional image processing techniques and map depth image coordinates to RGB image coordinates utilizing the API function provided by Kinect for Windows SDK v1.6. The third step is to detect pedestrians based on the extracted contours and the 3D information. This is done by developing a

region-cluttering algorithm which can easily and efficiently deal with the occlusion issue. The last step is to track and count pedestrians. Figure 4.1 illustrates a simplified flow chart of the proposed method. Its details are described step-by-step as follows.

4.4.1 Microsoft Kinect

Essentially, Kinect is a motion sensing input device marketed by Microsoft for the Xbox 360 video game console . This device has an RGB camera, an infrared (IR) emitter and an IR Depth camera, which make it capable of capturing a color image and depth of each pixel in the scene. These data contain visual and geometric information of the scene. The two images are complementary and provide us with a way to perform tasks that would be difficult when 2D imagery is used alone. Both the depth and RGB image have a resolution of 640x480 at 30fps, which is ideal for real-time application.

4.4.2 Pedestrian Contour Extraction from RGB Images

As illustrated in Figure 4.1, it is a fundamental step to extract the pedestrian contours from RGB video sequences streamed from Kinect. To achieve this, real-time segmentation of moving pedestrian regions in the RGB image must be accomplished. Background subtraction is a simple matter of identifying the background image and subtracting it from the current image to obtain the moving regions. This approach is by far the most popular, as it is natural and provides a strong cue for moving objects, thus effectively reducing the area of interest.

In background subtraction, the first step is to choose a suitable background model which can effectively deal with the dynamic background. After comparing it with other commonly used background models such as running average and frame difference, we selected the model developed by Zivkovic et al. (2004) to acquire the background in our approach.

Once the background is obtained, it is stored and subtracted from each incoming frame to get the foreground. Then thresholding with a threshold \mathbf{Th}_b is applied to the foreground to create a binary image. That is, for a given frame \mathbf{i} , the binary image's pixel value at location $\delta(\mathbf{x}, \mathbf{y}, \mathbf{i})$ is calculated as follows:

$$\delta(\mathbf{x}, \mathbf{y}, \mathbf{i}) = \begin{cases} \mathbf{0}, & \text{for } |I(\mathbf{x}, \mathbf{y}, \mathbf{i}) - \mathbf{B}(\mathbf{x}, \mathbf{y}, \mathbf{i})| < \mathbf{Th}_b \\ \mathbf{1}, & \text{for } |I(\mathbf{x}, \mathbf{y}, \mathbf{i}) - \mathbf{B}(\mathbf{x}, \mathbf{y}, \mathbf{i})| \geq \mathbf{Th}_b \end{cases} \quad (4.1)$$

where $\delta(\mathbf{x}, \mathbf{y}, \mathbf{i})$, $I(\mathbf{x}, \mathbf{y}, \mathbf{i})$ and $\mathbf{B}(\mathbf{x}, \mathbf{y}, \mathbf{i})$ are the pixel values at coordinate (\mathbf{x}, \mathbf{y}) on the foreground binary image, current frame \mathbf{i} , and background image, respectively; \mathbf{Th}_b is the threshold.

After the foreground binary image $\delta(i)$ is obtained, there still exist some speckles caused by noise, or vacancy and fractional false regions while the foreground and background are of similar color. To improve the foreground quality, we adopt a morphological reconstruction filter (19) as the post-processing procedure, defined as:

$$\delta_g(i) = \delta_m(i) \cap (\tilde{\delta}(i) \oplus SE) \quad (4.2)$$

where $\delta_g(i)$ is the final refined binary image, $\delta_m(i)$ is the 'mask', SE is a structure element with size of 7×7 pixels, and $\tilde{\delta}(i)$ is defined as:

$$\tilde{\delta}(i) = \delta(i) \cap (\delta(i) \oplus N) \quad (4.3)$$

where N denotes the 3×3 structuring element with its origin at the center.

Based on the above steps, an improved binary image $\delta_g(i)$ is obtained, which needs further processing to extract the pedestrian contours. In our approach, pedestrian contours were obtained from a binary image $\delta_g(i)$ using a simple approximation method which has been implemented as a standard method of the image class (*Image.FindContours()*) in OpenCV 2.4. Meanwhile, the small contours whose perimeters are less than a threshold $Th_{contour}$ are filtered out. The remaining ones are the extracted pedestrian contours to be used for extracting pedestrian with depth information.

4.4.3 Conversion from Depth Space to RGB Space

Even though we obtain the pedestrian contours from RGB image and the depth information, we cannot directly use them to extract pedestrian. Because RGB and depth images come from two different cameras with different fields of view, pixels in the two images do not always line up exactly.

For this reason, lining up data is an essential prerequisite to acquiring the 3D information of the pedestrian contours. Fortunately, Kinect for Windows SDK v1.6 provides an API function that enables us to achieve this complex task easily. It implements a mapping of depth coordinates to RGB coordinates. The API function we used is *CoordinateMapper.MapDepthFrameToColorFrame*. By calling this function, we can obtain the 3D information of the pedestrian contours.

4.4.4 Pedestrian Extraction

Table 4.1 Pseudo-code of Pedestrian Extraction Algorithm

```
Times-out-of-region=0; /* a variable to record the times a pixel not belonging to any existed seed region */
For each (pedestrians' contour in current frame)
  Get current contour's bounding-rectangle;
  For each (line y in the bounding-rectangle from top to bottom)
    For each (column x of the bounding-rectangle from left to right)
      If (pixel p(x,y) is out of the contour)
        Continue; /*continue to judge next pixel */
      End if
      If (pixel p(x,y).depth is not available)
        Continue; /*continue to judge next pixel */
      End if
      If (seed-list.counts==0) /* the first time to encounter a pedestrian in the contour, treated as a seed */
        Add pixel p(x,y).depth to the seed- list;
        Continue; /*continue to judge next pixel */
      Else
        For each (seed in current seed- list)
          If (Rj(p(x,y))==1) /*p(x,y) belongs to the current seed's region */
            Calculate Rj.depth ;
            Break; /*continue to judge next pixel */
          Else
            If (seed is the last one of the current seed-list)
              If (++times-out-of-region<5)
                Break; /*continue to judge next pixel */
              End if
            Add pixel p(x,y).depth to the seed- list; /* a new seed is found, and add it to the seed-list */
            Times-out-of-region=0;
            Break;
          End if
        End for
      End if
    End for
  End if
End for
End for
End for
```

From steps discussed above, we extract the pedestrian contours from the RGB image and acquire the 3D information of each pixel in the contours. However, each pedestrian contour may contain one pedestrian or more due to the possible occlusions which often exist in real word scenes. For traditional image processing approaches, a complex occlusion reasoning algorithm is required to deal with the issues caused by occlusion such as merging, splitting, etc. However, it

is still a great challenge for traditional 2D RGB image approaches if objects are occluded the entire time they are present in the scene.

By utilizing the 3D information of pedestrian contours we acquired, the issues related to occlusions may be easily and efficiently dealt with. We assume that in nearly every instance of occlusion, two or more occluded objects must have a different depth relative to the camera. Based on this, we develop a pedestrian extraction algorithm which is similar to the region growth algorithm (20) in concept but quite different in implementation.

Our pedestrian extraction algorithm scans each contour regardless of whether it contains one pedestrian or more. It is assumed that the depth values of the adjacent pixels vary gradually across a region occupied a single pedestrian, so that only an abrupt change in depth within a pedestrian contour will be identified as the start of a new region. It should be noted that depth can vary considerably from one edge of a pedestrian region to the other, due to the installation angle of the Kinect device.

The algorithm runs by searching seeds of different regions in the current pedestrian contour and judging which region a pixel belongs to. To remove the interference of the noise in depth data, a new region is judged to be effective only if five subsequent pixels do not match the depth value of any existing region. The last of the five unmatched pixels is the seed for the newly established region. Each region is treated as a pedestrian, and its position is marked by the position of its seed.

To judge whether a pixel $p_i(x, y)$ belongs to a region R_j with a seed $S_j(x, y)$, we define a criteria function as follows:

$$R_j(p_i(x, y)) = \begin{cases} 1, & |R_j.depth - p_i(x, y).depth| \leq Th \\ 0, & otherwise \end{cases} \quad (4.4)$$

where $p_i(x, y).depth$ and Th are the depth value of the pixel $p_i(x, y)$ and the threshold respectively; $R_j.depth$ denotes the mean depth of all the pixels in region R_j , and defined as:

$$R_j.depth = \frac{1}{N} \sum_{p_i \in R_j} p_i(x, y).depth \quad (4.5)$$

where $R_j(p_i(x, y)) = 1$, pixel $p_i(x, y)$ belongs to region R_j , otherwise the opposite.

The details of the pedestrian extraction algorithm are described in **Error! Not a valid bookmark self-reference.**

4.4.5 Pedestrian Tracking and Counting

Table 4.1 Pseudo-codes of Pedestrian Tracking and Counting Algorithm

```
If (template.Count==0) /* create the template */
  For each (pedestrian in current frame)
    If (pedestrian.depth within  $\Delta d$ )
      Add the current pedestrian to template;
      Initiate its untracked times to 0;
      ++pedestrian Counts;
    End if
  End for
Else
  For each (pedestrian in current frame)
    If (pedestrian.depth within  $\Delta d$ )
      Calculate  $d_{\min}(P_{Ci})$ ;
      If ( $d_{\min}(P_{Ci}) < \Delta r$ ) /* Pedestrian tracked, update the template */
        Update the template using  $P_{Ci}$ ;
        Update its untracked times to 0;
      Else /* a new pedestrian appear, add it to the template */
        ++pedestrian Counts;
        Add the current pedestrian to template;
        Initiate its untracked times to 0;
      End if
    End if
  End for
End if
For each (pedestrian in template)
  If (++its untracked times==2) /* a tracked pedestrian disappeared for 2 times,
  remove it from template */
    Remove it from template;
  End if
End for
```

Once the pedestrians are extracted, they have to be tracked from frame to frame to get the pedestrians volumes. To accomplish this, several parameters must be kept for each pedestrian: 3D position, ID tag, and untracked time.

The key to tracking pedestrians is to identify their presence in the current frame and each successive frame. In our proposed method, this is done by establishing a tracking template and comparing the parameters of the pedestrians in the current frame with those in the tracking template at a small predefined tracking range Δd , then finding the minimum cost match. For each

pedestrian with the minimum cost match, we assume that there is a threshold radius confining the range in which it should appear in the next frame. Thus, if the minimum cost is within that threshold radius, the corresponding pedestrian continues to be tracked and the matching one in the template is updated. Otherwise, it will be treated as an untracked pedestrian and removed from template if the untracked time gets to a given value. With the depth information as the tracking range Δd , the 3D Euclidean distance is used as the cost. So the cost of two potentially matching pedestrians $P_{Ci}(x_i, y_i, z_i)$ and $P_{Tj}(x_j, y_j, z_j)$ is calculated as follows:

$$d(P_{Ci}, P_{Tj}) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \quad i = 1, \dots, m; j = 1, \dots, n \quad (4.6)$$

where $P_{Ci}(x_i, y_i, z_i)$, $P_{Tj}(x_j, y_j, z_j)$ are the pedestrian in current frame and one in tracking template respectively; m and n are the numbers of pedestrians in the current frame and the matching template respectively. The minimum cost $d_{min}(P_{Ci})$ for pedestrian $P_{Ci}(x_i, y_i, z_i)$ in current frame is defined as:

$$d_{min}(P_{Ci}) = \min(d(P_{Ci}, P_{T1}), \dots, d(P_{Ci}, P_{Tj}), \dots, d(P_{Ci}, P_{Tn})) \quad (4.7)$$

Once the minimum cost match for a current pedestrian is found to be within the threshold radius Δr , it is considered to be the same pedestrian as it progresses from frame to frame. The detail of the pedestrian tracking and counting algorithm is described in Table 4.2.

4.5 Experiment

For implementation of our proposed approach, we used visual studio 2010, C# 4.0, .NET framework 4.0, Kinect for windows SDK v1.6, as well as Emgu CV 2.4 which is the wrapper of OpenCV under the .NET platform. OpenCV (Open Source Computer Vision Library) was initially introduced by Intel and released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000. This library is implemented in C/C++.

The hardware used in our experiment is Microsoft Kinect and a desktop computer with an Intel(R) Core(TM) i5-2400 3.1GHz CPU and 4GB RAM. Both RGB and depth images by Kinect are 640x480 resolution.

4.5.1 Experiment results

Experiments were conducted for all the three scenarios, and examples of successfully detecting and tracking pedestrians in the crowded scenes based on consecutive frames are demonstrated in Figure 4., Figure 4. and Figure 4.4. From Figure 4., we can see that the pedestrians were still detected and tracked even though severe occlusions occurred frequently in this scenario. In addition, our approach can also successfully detect pedestrians in scenes with a cluttered background where other still objects such as tables, computer, shelves, pillars etc. are present, which is a great challenge for other depth-based detection approach such as (Hsieh et al. 2012). It should be noted that, in Figure 4., the pedestrian behind the pillar in (a) and the last pedestrian in (b) were not detected just because they were out of range and their depth information was not available.

Figure 4. and Figure 4.4 show the experiment results in outdoor environment. In Figure 4.3 (a), (b), and (c), the farthest pedestrian in the white shirt cannot be detected for the same

reason; i.e. his depth was not available at that distance. However, other pedestrians can be successfully detected and tracked once their depth information is available.

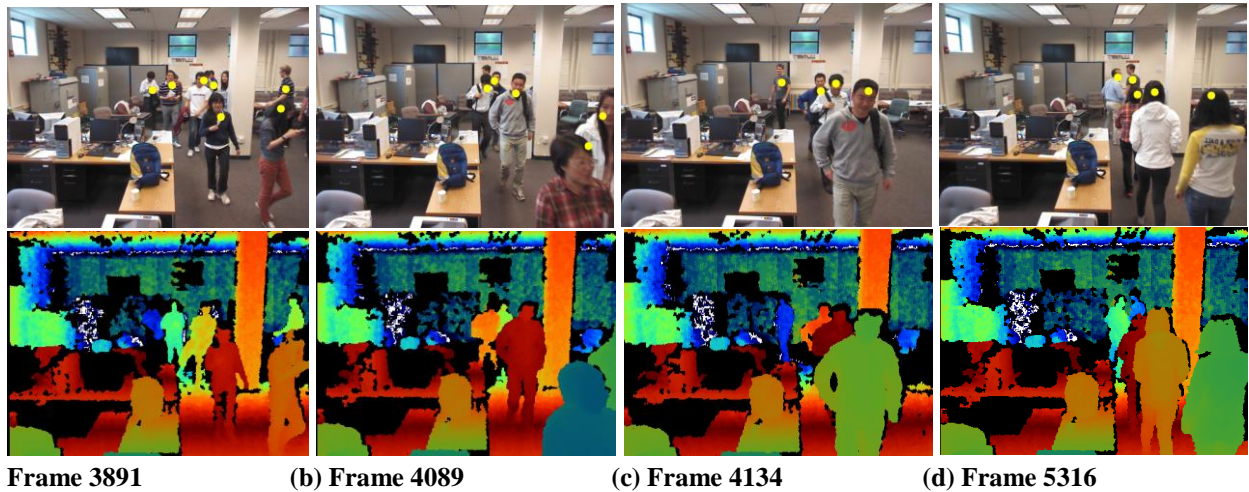


Figure 4.2 Examples of successful detecting and tracking pedestrians for scenario 1 (Yellow solid circle locating one pedestrian), RGB image on top while corresponding depth image on bottom

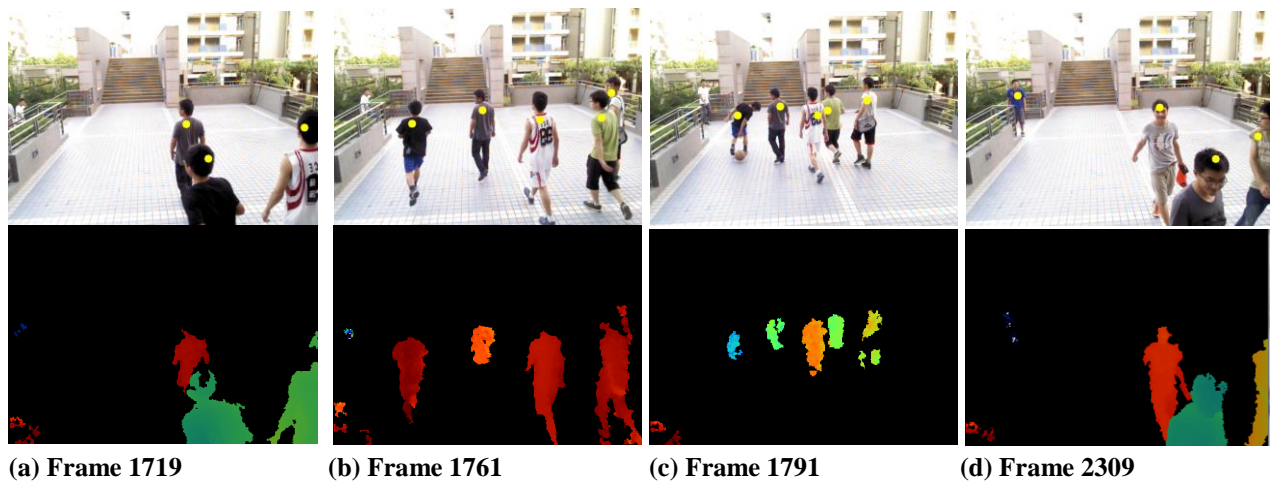


Figure 4.3 Examples of successful detecting and tracking pedestrians for scenario 2 (Yellow solid circle locating one pedestrian), RGB image on top while corresponding depth image on bottom

While in most cases the proposed approach in this paper can successfully detect and track pedestrians in crowded scene, missing detections occur in some situations where two or more pedestrians walk so closely that they have almost the same depth relative to the Kinect. As indicated in Figure 4.4, one of the four pedestrians was not detected in scenario 3. In fact, the undetected pedestrian and the adjacent detected one have nearly equivalent average depths of

8879mm relative to Kinect. According to our pedestrian extraction algorithm described in section 4.4, they were clustered to one region which resulted in the missing detection. Though missing detection exists in this case, the missing pedestrian could be tracked and counted once it is detected again. It should be noted that in Figure 4.5, others behind these four pedestrians are not detected just because they were out of the detection range at that moment which can be easily seen from the depth image.

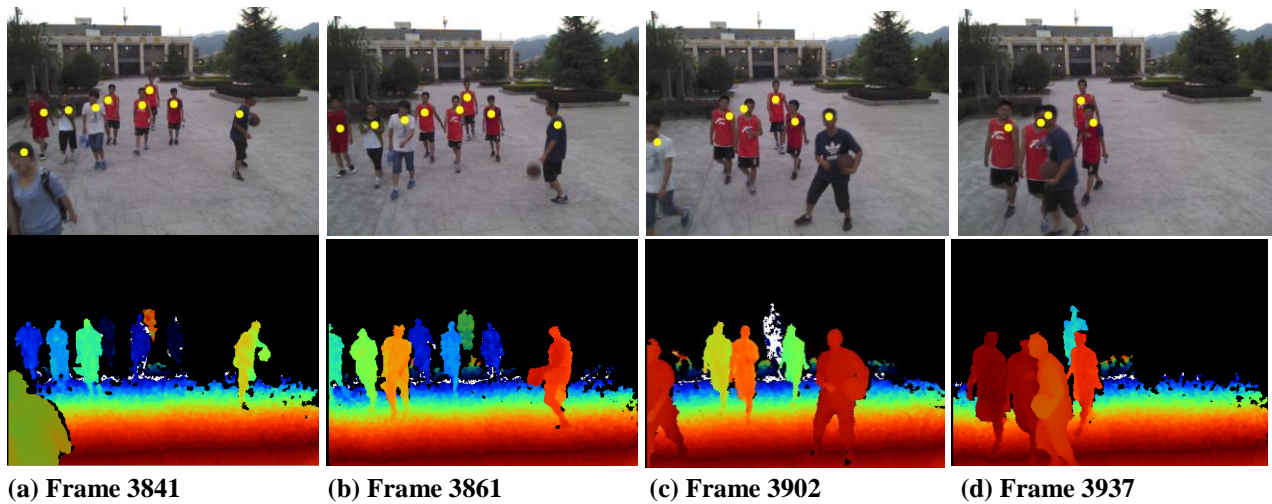


Figure 4.1 Examples of successful detecting and tracking pedestrians for scenario 3 (Yellow solid circle locating one pedestrian), RGB image on top while corresponding depth image on bottom

From the experimental results illustrated in Figure 4.-Figure 4.4, we can definitely conclude that in most cases, our proposed approach performs reasonably well for detecting and tracking pedestrians in crowded scenes with cluttering backgrounds, even though severe occlusions occur frequently.

It was also found that though Kinect was initially designed for indoor applications, it performs very well in outdoor environments. In our studied cases, scenario 2 is on a sunny day while scenario 3 is on a cloudy day. A pedestrian with a depth of at least 10m can be reliably



Figure 4.2 Example of missing detection (Yellow solid circle locating one pedestrian), RGB image on the left while corresponding depth image on right

detected in scenario 3, and at least 7m far away from Kinect in scenario 2. Both of these are beyond the reliable depth range of 4m reported by Microsoft.

4.5.2 Counting Accuracy and Real-time Performance

Table 4.1 summarizes the resulting counts and accuracies of the approach proposed in this paper under various scenarios. The automatic counts were generated by the tracking approach developed in this paper. Pedestrians were also manually counted as ground truth data. The results show that the indoor scenario has higher accuracy, but in general the results remain reasonably accurate.

Table 4.1 Summary of Counting Test Results

Scenario	Test length	Manual counts	Automatic counts	Under counting	Over counting	Accuracy (%)
1	5 min	56	59	0	3	94.7
2	5 min	60	56	4	0	93.3
3	5 min	58	54	4	0	93.1

Detection errors resulted in some over counting and under counting. The primary reason for over counting lies in the incomplete pedestrian contour formed due to vacancy and fractional false regions in places where the foreground and background are similar in color, which separates one pedestrian contour into two or more. Under counting results from a failure to detect one or more pedestrians when they have almost the same depth relative to Kinect, as indicated in Figure 4.5.

Real-time performance is essential for almost all the real world pedestrian detection system. Our proposed approach has an average running time of 50ms per frame in all the three tested scenarios, i.e. 20 fps for Kinect images with 640x480 resolution. This means it can be applied to scenarios where real time performance is required.

4.6 Conclusion

By utilizing Microsoft's Kinect device with 3D sensing capabilities, a low cost solution for occlusion robust pedestrian detection in crowded scenes was proposed and tested. The proposed approach fused the information extracted from RGB images with that from depth images to efficiently detect pedestrian in crowded scenes where occlusion occurs frequently, which is very challenging for traditional image based approaches. Three crowded pedestrian scenarios from indoor and outdoor environments were used to test the proposed approach, and the results are very promising both in terms of accuracy and real time performance. This demonstrates that the low cost Kinect is quite feasible in real-world pedestrian detection for crowded scenes.

The approach proposed in this paper could be effectively used in a broad range of applications including but not limited to:

Management of large pedestrian infrastructures such as public transport, railway stations, airports, stadiums, shop center etc. Pedestrian detection is required to quantify and monitor the demand for the infrastructure in order to correspondingly adjust supply.

- Onboard passenger volume estimation for public transit such as buses, light rail, ships, etc.
- Pedestrian data collection for academic research on pedestrian behavior and movement patterns, pedestrian traffic safety, etc.
- Pedestrian data collection for intersection geometric design, signal timings, etc.
- Other pedestrian surveillance applications related to security and safety.

While promising results were achieved, the approach proposed in this paper has a limitation in that missed detections occur especially when two or more pedestrians walk closely together such that they have almost the same depth relative to the Kinect device. What's more, pedestrian walking speeds and trajectories were not addressed in this paper, though they could be easily acquired from the proposed approach. Further enhancements in the detection and tracking algorithm will definitely help improve accuracy of the proposed approach.

CHAPTER 5.0 BLUETOOTH DATA COLLECTION SYSTEM FOR PLANNING AND ARTERIAL MANAGEMENT

5.1 Introduction

This chapter documents the research and development of an inexpensive portable wireless roadside data collection system. This system is comprised of the roadside data collection units (DCUs), and a web-based software application that is used to process the collected data.

The system developed utilizes Bluetooth technology as the wireless technology platform due to the widespread use of Bluetooth-enabled devices in vehicles, and builds off of results and knowledge gained from prior research projects (Porter et al. 2011; Kim et al. 2012). These projects focused on the development and implementation of a permanently installed Bluetooth-based wireless travel time data collection system for arterials. In this project a portable wireless roadside data collection system intended for short-term data collection purposes is designed, researched, and developed. The portable roadside DCUs can work as a system or in isolation depending on the needs of the application. Applications include:

- Intersection performance
- Origin-Destination data collection
- Travel time data collection.

The objectives of this part of the project are:

1. Utilize prior research experience to develop an inexpensive, easily deployed portable system for wireless Bluetooth-based automatic collection of vehicle movement data.
2. Use this prototype wireless data collection system to collect Origin-Destination data for traffic planning models.

3. Apply the system at an intersection to monitor intersection performance.
4. Evaluate the use of the proposed system to collect travel time data needed for project level analysis to monitor, evaluate, and maximize the performance of advanced traffic signal systems; and compare the performance of arterials.

The remainder of this chapter will begin with a literature review and background information on the data collected using Bluetooth wireless technology. This information is presented to facilitate better understanding of the data collection system operation and the research and development conducted. After the background information the project objectives will define report sections where the work and results related to that objective are presented.

5.2 Literature Review and Background Information

5.2.1 Performance Data with Bluetooth Sensors

Most of the applications of Bluetooth-based DCUs to transportation have been in the area of travel time data collection. In addition to Wasson et al. (2008), other examples of such research are Malinovski et al. (2010), Puckett and Vickich (2010), Quayle et al. (2010), Quayle and Koonce (2010), and Hagani et al. (2010). One exception is the study performed by Tsubota et al. (2011), where the “duration” (the length of time the same MAC address was seen by a sensor) and travel time data obtained from a system of Bluetooth DCUs installed on a main arterial in Brisbane, Australia, was analyzed. Scatter plots of duration (at the downstream detector) versus travel times between DCUs did not show a strong relationship between duration and travel times. The researchers stated that the lack of a strong relationship between duration and travel times was due to uncontrollable environmental factors. Finally, it was demonstrated through simulation that the relationship between duration and travel times is linear in

uncongested conditions, but becomes bimodal as congestion increases. However, no specific intersection performance measures are computed.

5.2.2 Data Description

This section presents background information on the data collected by Bluetooth-based wireless data collection units (DCUs) and how it is utilized. The objective of data collection by a set of DCUs is to estimate when a vehicle containing a discoverable Bluetooth device just passes each DCU. This is referred to as *point detection* and is depicted in Figure 5.1. If this can be done accurately for a series of DCUs on a specific road segment, then a number of performance measures can be estimated.

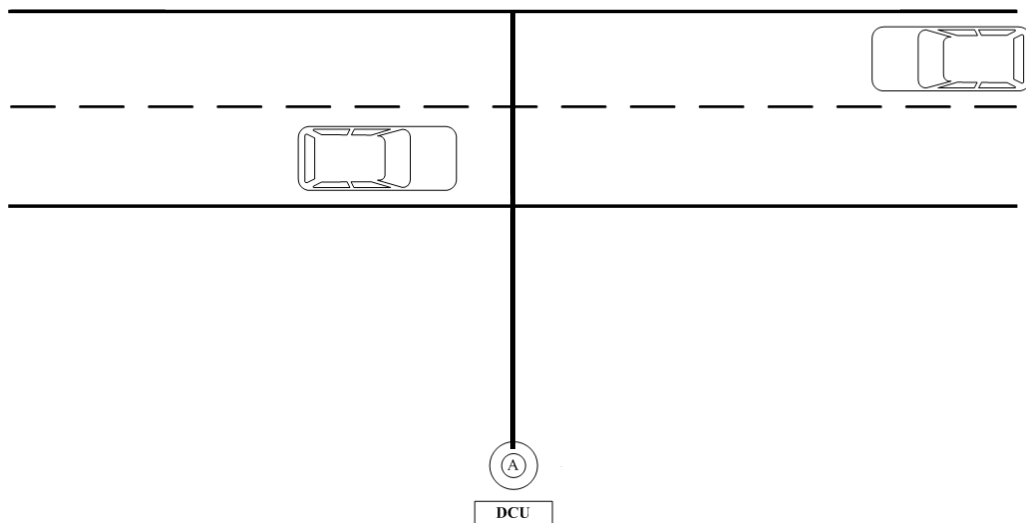


Figure 5.1 *Point detection* refers to estimating when a vehicle containing a discoverable Bluetooth device just passes the DCU marked by the vertical line across the road

Bluetooth-based DCUs repeatedly conduct an “inquiry” procedure to identify any Bluetooth devices that are within the unit’s antenna coverage area. Bluetooth devices within the

DCU coverage area will respond to this inquiry with a data packet that contains a media access control (MAC) address, which is used as an identifier for the vehicle containing the device.

Because the antenna coverage area of a DCU can cover several hundred feet (or more) of the road being monitored, and because multiple vehicles may travel on the road over the same time period, several features of the data collected by a single DCU are:

- There may be multiple MAC addresses detected over a fixed time period.
- A single MAC address may be detected multiple times as the vehicle travels past the DCU. The multiple detections are referred to as a *group*.
- The number of times a single MAC address is detected may vary for the different MAC addresses detected.
- Different MAC addresses may have the same time stamp (i.e., date and time).

For the portable data collection system developed, each DCU will have one or more data files that contain all detected MAC address records stored in a comma separated text file. No data filtering or data processing will be conducted on the DCU. Figure 5.2 shows a sample MAC address record.

Detected MAC Address	Date and Time	RSSI	DCU MAC Address
↓	↓	↓	↓
1F:16:76:C3,2013-04-29 09:57:32.265782,-66,0:1:95:17:C9:C5			

Figure 5.2 A sample MAC address record stored on a DCU

In Figure 5.2, RSSI is the received signal strength indicator, which is a measure of the detection strength. The RSSI has been used in past work to utilize a *single* DCU for point

detection. More information on the use of RSSI within a single DCU, and more detailed Bluetooth data collection background information can be found in Kim et al. (2012).

5.3 Portable Collection System Design

This section documents the final design of the portable data collection system design, beginning with a summary description of the system and its operation. This will be followed by descriptions of individual system components, new data processing web application developed for specific applications of the system, and system cost.

5.3.1 System Description

The portable Bluetooth-based data collection system provides a means to automatically collect vehicle movement data between points on a road segment defined by the location of individual DCUs. The system consists of multiple battery-powered DCUs and their packaging, and a web-based software application for data processing. A user of the system will deploy the DCUs on a temporary basis for a particular application of interest, and the total period of data collection will dictate the need for battery changes. After deployment, the user will collect the DCUs and download data from the DCUs to a USB jump drive. The raw data files can then be transferred to a personal computer. The user will access the web-based data processing application, upload the data files, and proceed with the appropriate data processing and analysis option. Results can be printed and/or copied to spreadsheets. Figure 5.1 shows a hypothetical deployment of four DCUs to collect data to estimate intersection performance. In Figure 5.1, the DCUs are packaged within traffic drums.

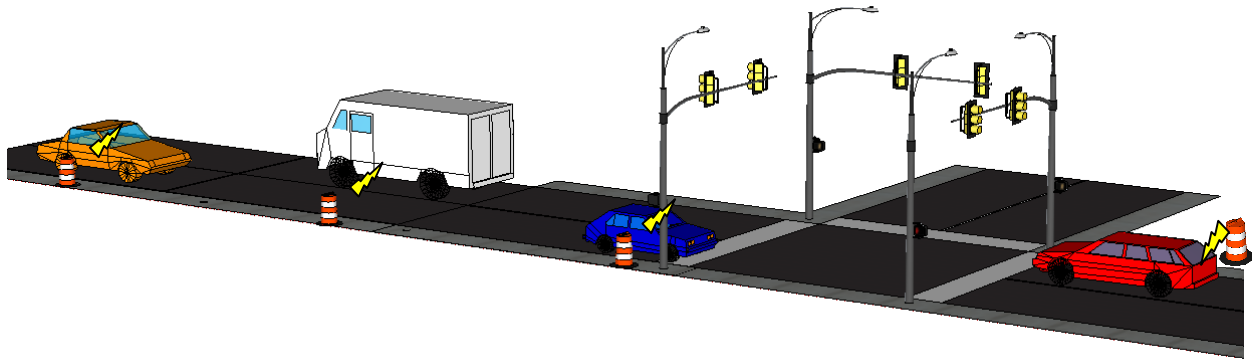


Figure 5.1 System deployment for intersection performance estimation

There are two types of DCUs: a *coordinator* DCU, and a *router* DCU. The use of a coordinator DCU depends on whether the specific application requires time synchronization between the DCUs. The application depicted in Figure 5.1 and other applications where vehicle movement data is of interest will require time synchronization. Periodically, all DCUs will update their time to match the coordinator DCU. For such applications, a single coordinator DCU is used, and the DCUs must be close enough to another DCU (within several hundred feet) to communicate with it. The DCUs communicate using a Zigbee radio and the system of deployed DCUs will constitute a “mesh network,” which means that a router DCU that cannot communicate directly with the coordinator DCU can do so through other router DCUs.

In other applications such as origin-destination data collection, time synchronization is not needed since the primary interest is in vehicle trip counts between various origins and destinations. Time synchronization is also not needed for temporary travel time data collection over road segments of approximately one half mile in length or greater.

The system was developed to be as user-friendly as possible and a “kit” of DCUs can be deployed by following a simple start-up procedure. The start-up procedure will vary depending

on the data collection application. Establishing DCU operation time settings and data download are accomplished by simply plugging a USB jump drive into a DCU.

5.3.2 Data Collection Unit Hardware and Software

The complete data collection unit consists of the following three main components:

1. Data collection microcontroller unit in an enclosure with an external Bluetooth adapter, micro-SD memory card, an integrated Zigbee radio, and an antenna for the Zigbee radio mounted to the enclosure.
2. An external directional antenna and antenna cable for the Bluetooth adapter.
3. An external rechargeable battery and battery cable.

In addition to the main DCU components, a GPS module that is only necessary at start-up to establish the time is also a part of the system. Since the GPS module is only used at start-up one module can be used with multiple DCUs. Figure 5.1 below shows the microcontroller unit with the Bluetooth adapter, Zigbee radio, and Zigbee antenna mounted to the opened enclosure. The components in Figure 5.1 normally stay together (except for the Zigbee antenna), and are matched with an external Bluetooth antenna, Zigbee antenna, and battery as needed.

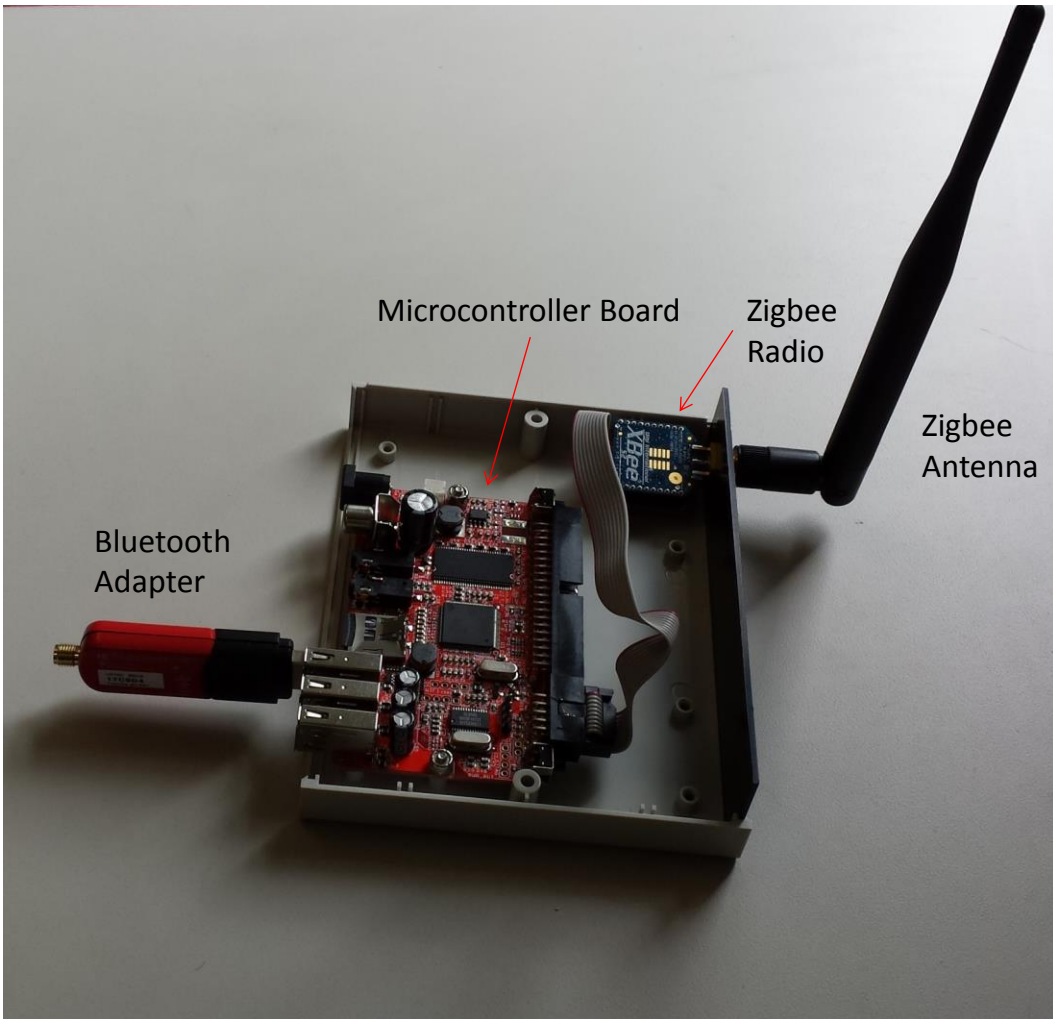


Figure 5.1 Microcontroller unit portion of the DCU



Figure 5.2 External Bluetooth antenna and cable battery



Figure 5.3 Battery and GPS module

The components in Figure 5.2 and Figure 5.3 are paired with a microcontroller unit as needed, and the GPS module is only needed when the system is started. Thus one GPS module can support multiple system deployments.

The specific component model numbers are shown in **Error! Reference source not found.**

Table 5.1 DCU Component List

Component	Manufacturer	Model Number
Microcontroller board	Olimex Ltd.	iMX233-OLINUXINO-MINI
Bluetooth adapter	Parani	Parani-UD100
Memory card	Sandisk	2GB MicroSD memory Card
Zigbee radio module	Digi	XBee XB24-Z7SIT-004
Antenna for Bluetooth adapter	HyperGain	HG2408P
Antenna for Zigbee radio	Taoglas	GW.71.5153
External DCU battery	Turcom	Ultra Capacity 33600mAh Portable Power Bank Charger

There are two types of DCUs:

1. *Coordinator* DCU. The coordinator sends the time to other DCUs for the purposes of time synchronization. If the DCUs deployed are to be used in a system with time synchronization, then the GPS module is only used with the start-up of the coordinator DCU.
2. *Router* DCUs synchronize time with the coordinator DCU, and also send data to the coordinator. They also serve as nodes in a mesh network if a router DCU cannot communicate directly with the coordinator DCU.

Both types of DCUs can be operated in two modes:

1. *System* mode where time synchronization to a single coordinator DCU is executed.
2. *Stand-alone* mode where the Zigbee radio is not operated and the DCU detects and stores MAC addresses in isolation (e.g., as is the case in for origin-destination data collection).

In this case each DCU must be started using a GPS module to establish the correct date and time.

The operational mode is selected by the start-up procedure completed. The software for operating the DCU which includes the operating system, start-up scripts, and python code source code is stored on a micro-SD card that is inserted into the DCU and accessed when the DCU is started. The main specifications of the software are:

- Operating system – Arch Linux.
- Python source code with the PyBluez and PySerial libraries for operating the Bluetooth adapter and Zigbee radio.

The main functions implemented in the start-up script, which are a series of Linux commands, are:

- Detection of the presence of a USB flash drive. If present, MAC address data files will be downloaded (moved) from the DCU to the USB drive, and text files (with DCU settings) will be copied from the USB drive to the DCU.
- Detection of the presence of a USB GPS receiver. If present the date and time on the DCU will be set from the date and time obtained through the GPS receiver.
- Running the DCU in the proper mode.
- Control the on/off running status of the Python code.

The main functions of the Python code are:

- Execution and control of the Bluetooth inquiry procedure and the storing of detected MAC addresses.
- Controlling communication between the DCUs using the Zigbee radios.

5.3.3 Web-Based Software Application for Data Processing

To process the data collected by the portable data collection system, a web-based—currently hosted on an OSU server—application has been developed. The URL for the application is <http://research.engr.oregonstate.edu/btdataanalysis/logIn.php>.

The application has been designed with three main data processing and analysis procedures for the following applications:

1. Origin-Destination data collection,
2. Travel time data collection,
3. Vehicle movement and road segment performance estimation (in progress).

The latter application includes various types of road segments, but was developed with intersection performance as a primary use.

The user of the application will normally execute the following steps after accessing the application.

- Choose the appropriate data processing and analysis procedure for the application,
- Upload raw data files obtained from the DCUs,
- Enter parameters for the analysis such as the location of the DCUs, the MAC address of the DCUs, the maximum allowed travel time between locations, etc.
- Conduct the analysis, view the results, and copy the results from the application directly into a spreadsheet.
- Close the application.

No data is saved on the server, so if the user closes the application raw data uploads will be required if the analysis is to be repeated. Figure 5.1 shows a screen shot of the origin-destination data processing screen after files have been uploaded and the analysis parameters entered.

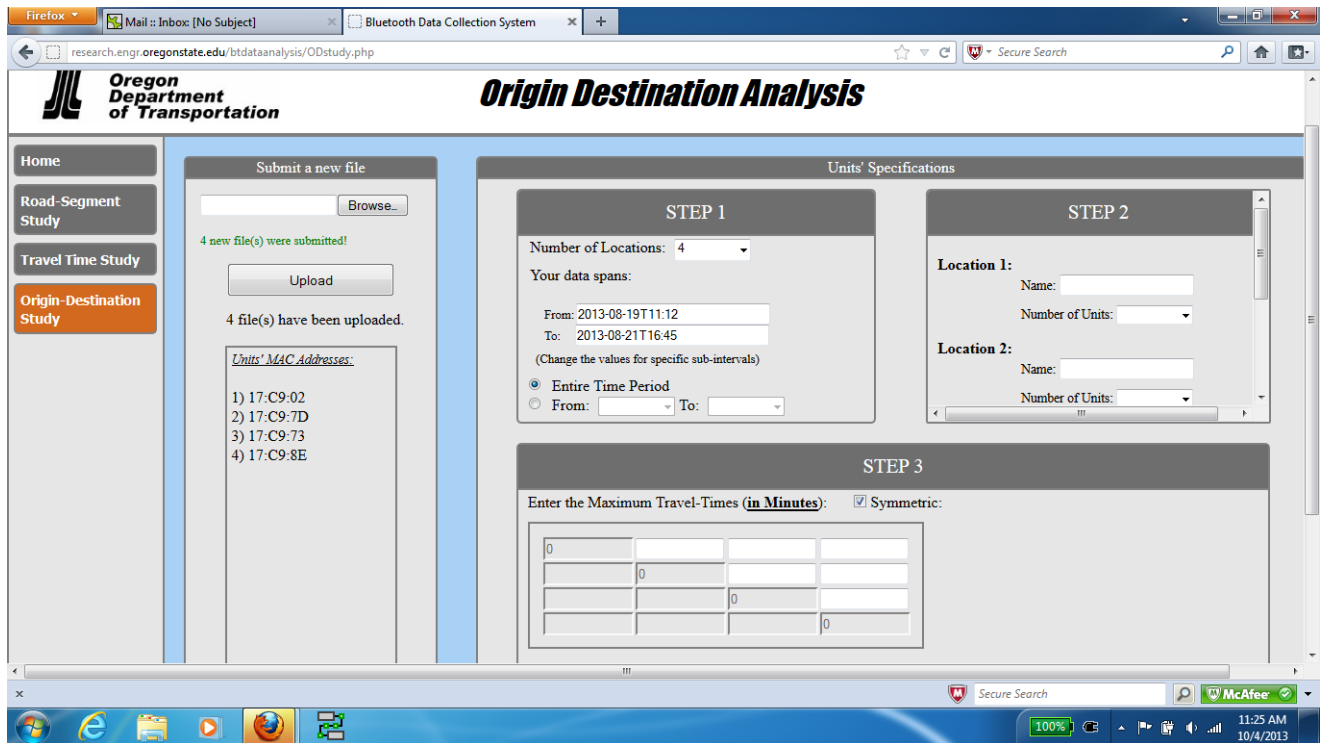


Figure 5.1 Web application screen shot showing origin-destination data entry

Figure 5.2 shows a screen shot of the origin-destination data analysis screen after the analysis has been completed.



Origin-Destination Analysis Summary

You can select, copy, and paste tables into an Excel datasheet

Selected interval:

From: 2013-04-29 09:57
To: 2013-05-03 15:48

Number of trips between locations:

	I-5 North	I-5 South	Sweet Home	Philomath
I-5 North	-	4932	78	104
I-5 South	5539	-	19	58
Sweet Home	132	8	-	17
Philomath	90	47	30	-

Average travel time between locations:

	I-5 North	I-5 South	Sweet Home	Philomath
I-5 North	-	12.32	34.09	39.16
I-5 South	12.24	-	28.98	31.29
Sweet Home	36.78	29.65	-	58.74
Philomath	38.96	30.52	60.48	-

Total number of vehicles detected at each location:

I-5 North	I-5 South	Sweet Home	Philomath
19896	16495	2626	1973

Total number of unique vehicles detected at each location:

(e.g., the same MAC address detected on different days is only counted once.)

I-5 North	I-5 South	Sweet Home	Philomath
11479	10028	1014	903

Figure 5.2 Web application screen shot showing origin-destination data analysis

5.3.4 System Cost and Packaging

The packaging developed for the system consists of a traffic drum with a painted external Bluetooth antenna. The microcontroller unit, battery, and cables are mounted inside the drum with zippered pouches.

Figure 5.1 shows a DCU deployed for origin-destination data collection with the Bluetooth antenna oriented perpendicular to the direction of traffic.



Figure 5.1 DCU deployed on I-5 for origin-destination data collection

Costs for individual components are shown in Table 5.1. The costs in Table 5.1 do not reflect volume discounts and are rounded up to the nearest dollar. For antennas volume discounts can be up to 40% less costly, but are less for other components.

Table 5.1 DCU Components – Approximate Pricing

Component	Manufacturer	Unit Cost
Microcontroller board	Olimex Ltd.	\$47
Bluetooth adapter	Parani	\$35
Memory card	Sandisk	\$6
Zigbee radio module	Digi	\$40
Realtime clock module	Olimex Ltd.	\$7
Microcontroller enclosure	NA	\$8
<u>Microcontroller Unit Total</u>		<u>\$143</u>
Bluetooth antenna and cable	HyperGain	\$32
Antenna for Zigbee radio	Taoglas	\$10
External DCU battery	Turcom	\$120

The Bluetooth and Zigbee antennas and the battery are intended to be purchased separately and to be paired with microcontroller units as needed. These particular components all have large volume discounts.

5.4 Origin-Destination Study Data Collection Test

Six DCUs were deployed to collect origin-destination data for a planning model of the Corvallis-Albany-Lebanon area. Table 5.1 is a list of the locations where vehicle counts and vehicle trip counts are to be recorded. The highlighted rows show the four locations where DCUs were deployed. Count ID locations 97 and 105 were on interstate highway I-5 and had two units deployed at each location. Due to the width of the median, one DCU was placed on the northbound side and another DCU was placed on the southbound side. Figure 5.1 is a map showing the numbered data collection locations.

Table 5.1 Data Collection Locations for Origin-Destination Data

Count ID	Intersection/Location	City
9	On Jefferson Hwy Hwy 164 (OR164) and west of Mill St. in Jefferson	Jefferson
95	Sodaville-Waterloo Dr & Hwy 016 (US20)	South of Lebanon and West of the City of Waterloo
96	Berlin Rd & Waterloo Rd	South of Lebanon and East of the City of Waterloo
97	I5 North of Dever Rd Interchange.png	Albany
98	Hwy 226 East of (Hwy20) Santiam Hwy SE	East City of Albany
99	Brewster Rd East of Berlin Rd.	Lebanon
100	I5-Exit 228 south of Corvallis Lebanon Hwy	Albany
101	Hwy 99E South of Bell Plain Dr.png	South of City of Tangent
102	99W Pacific Hwy South of Smith Loop-.png	South City of Corvallis
103	BT-CountID103-Hwy 34 and Decker Rd-.png	South west of City of Philomath
104	Corvallis Newport Hwy and Hwy 34	West City of Philomath
105	99W Pacific Hwy North of Camp Adair Rd	North of the City of Adair Village

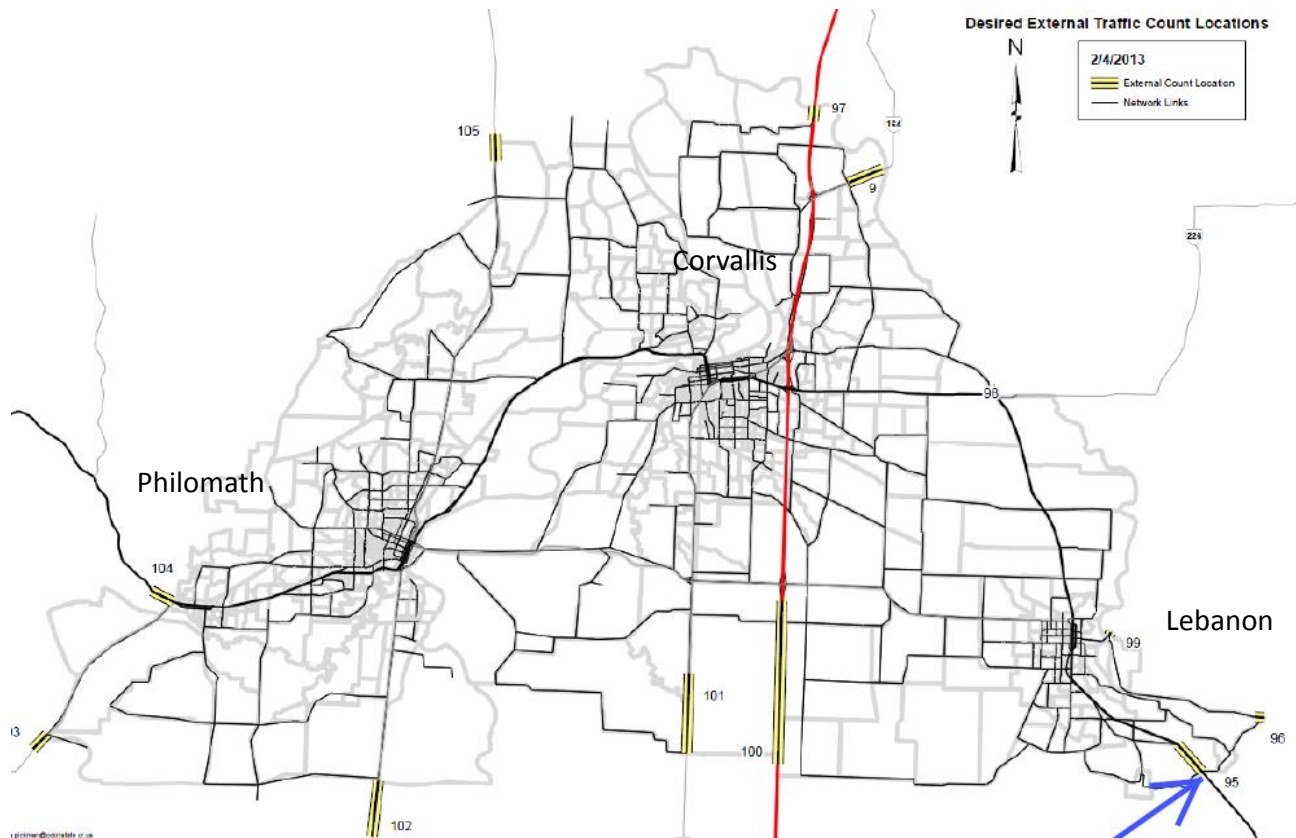


Figure 5.1 Data collection locations in the Corvallis, Albany, Lebanon area

The DCUs were continuously operated from approximately 11am, Monday, April 29, 2013 to 3pm, Friday, May 3, 2013. Battery units were changed with completely charged batteries every 24 hours. A single DCU at count ID location 97 (southbound side) was stolen after one day of deployment. Table 5.2 shows the exact deployment time periods.

Table 5.2 DCU Deployment Periods for Origin-Destination Data Collection

Count ID	Start Date and Time	End Date and Time
95	2013-04-29 11:30:51	2013-05-03 14:27:45
97	NB Unit: 2013-04-29 10:40:45 SB Unit: 2013-04-29 10:51:49	NB Unit: 2013-05-03 15:08:52 SB Unit: Missing
100	NB Unit: 2013-04-29 10:23:27 SB Unit: 2013-04-29 09:57:32	NB Unit: 2013-05-03 14:54:25 SB Unit: 2013-05-03 14:27:45
104	2013-04-29 12:18:13	2013-05-03 11:57:37

The web-based application described in section 5.3.3 was used to process the data collected by the DCUs. Table 5.3 shows the travel time maximum values used between the DCU locations when processing the data. Any MAC address “matches” between two locations with a time difference greater than the value in Table 5.3 were not counted. This prevents the counting of matches caused by vehicle detections on different days. The parameters in Table 5.3 are easy to change and are shown as symmetric, but can be entered as non-symmetric times.

Table 5.3 Maximum Travel Time used for Origin-Destination Data Processing

Origin- Dest	ID 97	ID100	ID 95	ID 104
ID 97	*	20 min	40 min	40 min
ID 100	20 min	*	60 min	60 min
ID 95	40 min	60 min	*	80 min
ID104	40 min	60 min	80 min	*

Table 5.4 through Table 5.7 are the four tables produced as output from the web-based data processing application.

Table 5.4 Trip Counts

Origin- Dest	ID 97	ID100	ID 95	ID 104
ID 97	*	5539	17	49
ID 100	4932	*	79	104
ID 95	7	131	*	15
ID104	41	90	26	*

Table 5.5 Average Travel Times for the Trips in **Table 5.2** (minutes)

Origin- Dest	ID 97	ID100	ID 95	ID 104
ID 97	*	12.24	26.65	28.29
ID 100	12.34	*	34.16	39.11
ID 95	27.15	36.71	*	51.24
ID104	27.17	38.96	53.53	*

Table 5.6 Total MAC Address Counts

ID 97	ID100	ID 95	ID 104
16499	19896	2627	1973

Table 5.7 Total Unique MAC Address Counts

ID 97	ID100	ID 95	ID 104
10027	11478	1013	902

In this test study, two DCUs were placed at each I-5 data collection location due to the width of the highway. This was done to examine if a DCU located on one side (e.g., southbound) was missing a large number of detections for vehicles traveling on the other side of the highway (e.g., northbound vehicles). At count ID location 100 (I-5 and Highway 34) there were 10,027 unique MAC addresses detected over the study period. The DCU on the southbound side detected 8,596 unique MAC addresses, and the DCU on the northbound side detected 9,190 unique MAC addresses. This indicates that a large number of detections were missed by both DCUs. The recommendation that follows is to use one DCU on each side of the road for deployments on wide roads such as I-5.

5.4.1 Estimation of Total Trip Counts

Because not all vehicles contain discoverable Bluetooth devices, the trip counts collected using the portable DCUs represent a fraction of the total trip count. Assuming that the percentage of vehicles containing discoverable Bluetooth devices is constant over time and different geographic areas, a “multiplier” can be used with the trip counts collected to estimate total trip counts. Tests were conducted to estimate the value of this multiplier. The first test estimated the fraction of traffic volume detected by the portable DCU, and the second test estimated the percentage of “trips” where the same discoverable Bluetooth device is detected by two different DCUs positioned at two locations along a trip route. Figure 5.1 shows the location of a single DCU positioned at two locations along a trip route. Figure 5.1 shows the location of a single DCU (adjacent to highway 34 in Corvallis, Oregon) that was utilized to estimate the fraction of vehicles detected by the DCU.



Figure 5.1 Data collection location used to estimate the vehicle fraction detected

The DCU was run for one hour, and the total number of vehicles travelling on highway 34 over this same hour (in both travel directions) was counted and recorded. 134 different MAC addresses were detected. These MAC addresses are assumed to be associated with vehicles since no address generated group sizes that indicated otherwise. 2045 vehicles were counted over this same period so that $134 \div 2045 = 6.55\%$ of the vehicles were detected.

The second test was conducted on an approximately one mile round trip route on the Oregon State University campus. On this route, two DCUs were located approximately 0.5 miles apart. A single test run consisted of vehicle with a discoverable Bluetooth being driven past both DCUs. After each test run, the vehicle was driven outside of the DCU coverage area and the Bluetooth device in the vehicle was turned off. After a two minute wait the Bluetooth device was restarted and the next test run was conducted. 20 test runs were completed, and the vehicle was detected at both DCUs in 19 of these test runs.

Combining the results of these tests, it is estimated that on average 6.2% of the vehicle trips are recorded by the portable DCUs, which implies that trip counts should be multiplied by 16.1 to obtain an estimate of total trip counts. This is consistent with the travel time data collection sampling rate reported in Kim et al. (2012), which was computed from a larger sample of data. It is also consistent with the results obtained from the intersection performance measurement test documented in section 5.5.

5.4.2 Conclusions

The data collection potential offered by the portable DCUs is supported by the test conducted. The Oregon Department of Transportation Traffic Planning and Analysis Unit (TPAU) also hired a consultant to collect count data at the locations shown in Figure 5.1. This

consultant was not able to collect counts at all locations due to problems with the pneumatic tube counters. Trip count data was unavailable due to both cost and feasibility. After the conclusion of this test TPAU requested a larger scale data collection study in the city of Newport, Oregon. Twenty-six DCUs were deployed in two different months to collect trip count and travel time data for three consecutive days (Wednesday to Friday). This study was completed successfully and provided a much less expensive data collection alternative.

Prior research (Kim et al., 2012) has shown that the travel time samples collected by similar DCUs are very accurate, and that a reasonable multiplier for trip counts is 16.1. This multiplier value was also verified in tests conducted as part of this project.

5.5 Intersection Performance Test

This section presents results from the application of the Bluetooth-based data collection system to collect vehicle movement data for vehicles traveling through an intersection. Vehicle movement data are the travel times (short in duration) between the DCUs, which are placed at known locations along the road with specific distances between each DCU. The travel time data samples collected at the intersection also include the time that it takes for a vehicle to interact with the control mechanism at the intersection. The test was conducted at an intersection with large amounts of pedestrian and cyclist traffic relative to vehicle traffic. The test location was at the intersection of NW Monroe Ave and NW Kings Blvd near the Oregon State University campus in Corvallis, Oregon (Figure 5.1). The test was completed on a Friday, during noon rush hour (from 11:00 am to 12:00 pm). This three-way stop-controlled intersection has several businesses in the vicinity and experiences high/medium levels of traffic in different modes, including passenger vehicles, buses, pedestrians and bicyclists. The frequent occurrence of

different travel modes introduces a very high level of complexity to the system since active Bluetooth devices are present in all travel modes. This environment represents one of the more complicated environments where such a system may be deployed.



Figure 5.1 Intersection utilized for the data collection test

The goal of this test was to compare the time duration that vehicles spend at the intersection obtained from the wireless data collection system to the same data obtained manually. For the purpose of this experiment, ground-truth intersection time duration data was obtained by video recording the intersection. The next section provides a summary of the test setup.

5.5.1 Intersection Test Setup

The overall setup configuration for this test is presented in Figure 5.1. Three DCUs were utilized in this test and they were located so that the beginning, stop, and the end points of the functional area of the intersection could be monitored for eastbound traffic. MAC address data

was collected for one hour. Two high definition (HD) and high-speed cameras were utilized to record traffic at the DCU locations. The high-speed feature allowed for the recording of up to 60 frames per second, which made it possible to track moving objects with very high accuracy. The video recorded by the cameras was used for validation purposes and made it possible to see exactly when a detected vehicle just passed a DCU.

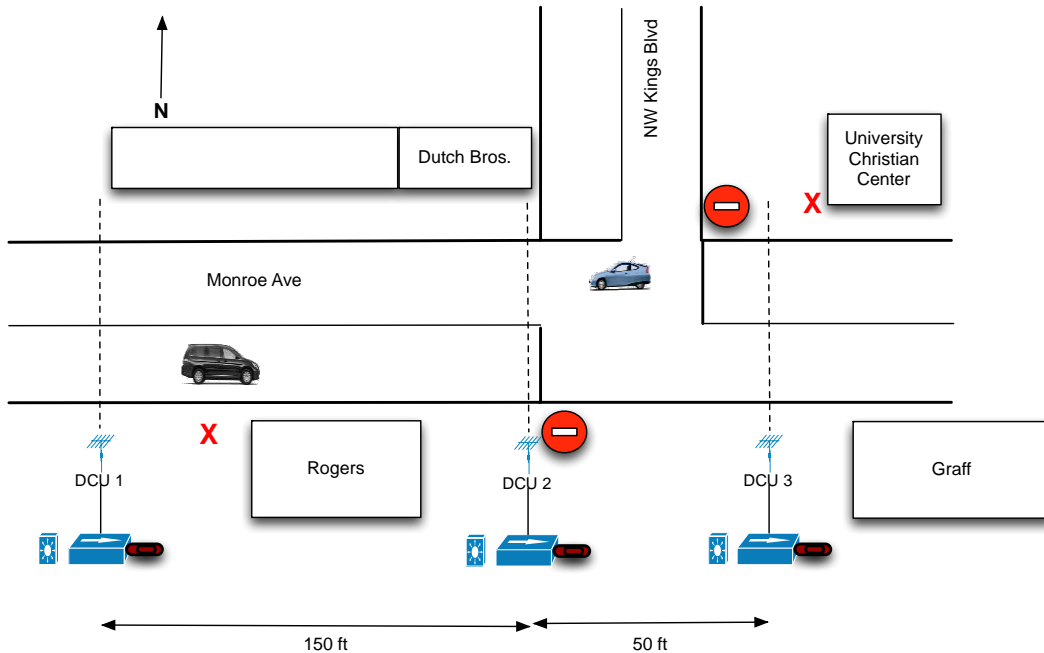


Figure 5.1 Test setup for intersection performance data collection test

5.5.2 Test Results

A total of 54 unique MAC addresses were detected by all three DCUs during the one-hour data collection period. By reviewing the video data from both cameras, a total of 25 unique MAC addresses were matched to specific vehicles (or a platoon of vehicles). In cases when a platoon of vehicles passed the DCU locations, it was not clear exactly which individual vehicle contained the active Bluetooth device. However, since the accuracy assessment is based on measured travel time between reference points (i.e., DCU locations) this did not affect the test results. Four hundred vehicles travelled past all three DCUs (from video review) during the one-

hour data collection period. This means that the installed DCU system was able to capture travel times for approximately 6.25% of the total traffic. Of the 25 vehicles detected, the travel times recorded by the DCUs were the same as those calculated from the video in 14 cases. In the other 11 cases, a maximum error of 6 seconds was recorded with an average error of 1.14 seconds. The summary of the results is presented in Table 5.1.

Table 5.1 DCU to DCU Travel Time Accuracy Results

	MAC	Travel Time From DCUs (sec)		Travel Time From Video (sec)		Travel Time Error (sec)	
	Address	DCU1 – DCU2	DCU2 – DCU3	DCU1 – DCU2	DCU2 – DCU3	DCU1 – DCU2	DCU2 – DCU3
1	1C:A1:2F:47	1	7	1	7	0	0
2	18:A3:6B:03	N/A	15	N/A	15		0
3	7E:AD:FB:B8	10	6	10	12	0	6
4	D1:68:B6:6C	5	13	5	13	0	0
5	43:7F:EA:02	16	9	16	15	0	6
6	6C:A7:3F:7A	10	5	12	5	2	0
7	6C:A7:3F:7A	N/A	5	N/A	9		4
8	7E:74:28:B0	5	4	5	4	0	0
9	9F:B7:14:E6	4	7	4	7	0	0
10	FE:73:4A:5C						
11	7E:25:51:47	N/A	13	N/A	13		0
12	AE:6D:FA:3B	5	7	5	7	0	0
13	58:0E:9E:36	5	9	10	4	5	5
14	4B:DE:10:B9	12	6	12	6	0	0
15	16:67:00:E0	11	5	11	8	0	3
16	1C:14:19:BF						
17	68:96:34:C0	6	10	6	10	0	0
18	7E:5D:3E:85	16	4	16	4	0	0
19	1C:A1:A7:36						
20	3D:1F:94:A4	9	5	9	5	0	0
21	3D:1F:94:A4	N/A	2	N/A	2		0
22	0C:5A:ED:DD	16	5	16	5	0	0
23	BE:6B:ED:CD	7	4	7	4	0	0
24	BE:46:1D:E4						
25	3E:EC:AF:75	14	7	14	7	0	0
					Average Travel Time	0.41	1.14
					Max (seconds)	5	6

In Table 5.1, the cells with “N/A” indicate that there was no record from that road segment for a particular vehicle (identified by the detected MAC address). The particular vehicle may not have passed all three DCUs, and/or was not detected by a DCU. For four MAC

addresses in Table 5.1 (10, 16, 19, and 24) all fields have been left blank. In these cases, the MAC address could not be paired to a vehicle from the video.

For this study, the functional area of the intersection can be defined as the length of the road between DCU 1 (150ft upstream of the stop bar) and DCU 3 (50ft downstream of the stop bar). Within this length of the road, vehicles approaching the intersection on eastbound NW Monroe Avenue interact with the traffic control device (a stop sign in this test). The average time it takes a vehicle to travel between DCU 1 and DCU 3 can be used as an intersection performance measure. It can be compared to the travel time at the posted speed limit. 17 of the 25 vehicles in Table 5.1 drove eastbound past all three DCUs. The average travel time from DCU 1 to DCU 3 obtained from the wireless data collection system is 16.5 seconds. The average time obtained from video recordings is 18.2 seconds, which is a 1.7 second difference. At the 25 MPH speed limit a vehicle traveling from DCU 1 to DCU 3 would take 5.45 seconds, thus the estimated delay due to the eastbound traffic control device is 11.05 seconds.

5.5.3 Conclusions

The results of the intersection test show that the portable DCUs have the potential to accurately monitor the performance of an intersection. Utilizing more than three DCUs would allow more detailed estimates of vehicle trajectories through the intersection. This is currently being researched as part of an Oregon Department of Transportation sponsored research project.

5.6 Traffic Signal Timing Evaluation

Three DCUs were deployed along Highway 99W in Sherwood, Oregon to test their use in travel time data collection to evaluate signal timing changes. The primary consideration was northbound travel along Highway between the DCU locations shown in Figure 5.1. DCU number

1 is the farthest south, and DCU number 3 is the farthest north. The DCU locations were determined by the Oregon Department of Transportation who are responsible for the signal operation along Highway 99W. Travel time data between the DCUs was collected for two separate time periods between which a signal timing change was to be implemented. The data collection periods were September 17, 2013 (00:00) to September 19, 2013 (23:59), and October 1, 2013 (00:00) to October 3, 2013 (23:59). Both data collection time periods were from Tuesday through Thursday.

The travel time data collection summary for both time weeks is shown in Table 5.1. Although the DCUs were only deployed on the east side of Highway 99W to collect northbound travel times, the DCUs were able to collect a number of southbound travel times, which are also shown in Table 5.1. Table 5.2 and Table 5.3 present the same summary information for the morning and evening rush hour periods.

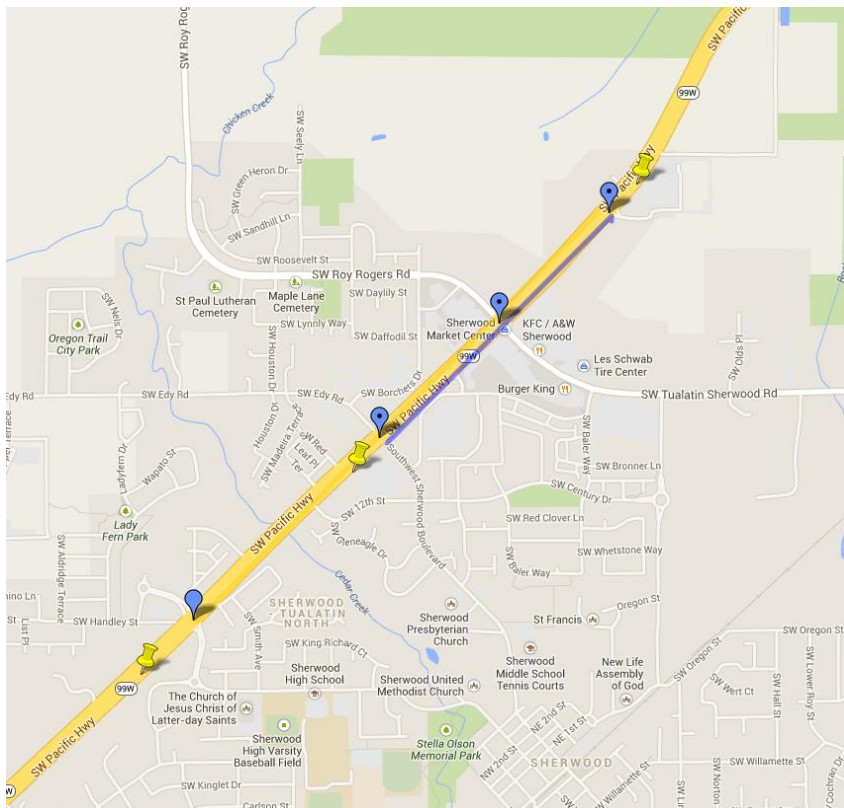


Figure 5.1 DCU deployment locations (yellow pins) along Highway 99W in Sherwood, Oregon

Table 5.1 Travel Time Data Collection Summary for Highway 99W Signal Timing Evaluation

September 17, 2013 (00:00) to September 19, 2013 (23:59)				October 1, 2013 (00:00) to October 3, 2013 (23:59)			
<u>Number of trips between DCUs</u>				<u>Number of trips between DCUs</u>			
	1	2	3		1	2	3
1	-	3097	1590	1	-	2623	1296
2	1922	-	1742	2	953	-	1483
3	1360	1145	-	3	605	990	-
<u>Average Travel Time Between DCUs (Minutes)</u>				<u>Average Travel Time Between DCUs (Minutes)</u>			
	1	2	3		1	2	3
1	-	1.09	3.33	1	-	1.1	3.41
2	0.99	-	2.21	2	0.93	-	2.17
3	4.08	2.91	-	3	3.9	2.81	-
<u>Total Number of Vehicles Detected at each DCU</u>				<u>Total Number of Vehicles Detected at each DCU</u>			
	1	2	3		1	2	3
	7193	6815	6143		4412	5945	5769
<u>Total number of unique vehicles detected at each DCU*</u>				<u>Total number of unique vehicles detected at each DCU</u>			
	1	2	5		1	2	3
	3340	3367	3068		2515	2987	2795

*The same MAC address detected on different days is only counted once.

Table 5.2 Travel Time Data Collection Summary for Highway 99W, 6AM - 8AM

Sept. 17, 2013 (06:00-08:00) to Sept. 19, 2013 (06:00-08:00)				Oct. 1, 2013 (06:00-08:00) to Oct. 3, 2013 (06:00-08:00)			
<u>Number of trips between DCUs</u>				<u>Number of trips between DCUs</u>			
	1	2	3		1	2	3
1	-	501	273	1	-	471	276
2	143	-	312	2	62	-	325
3	69	67	-	3	27	62	-
<u>Average Travel Time Between DCUs (Minutes)</u>				<u>Average Travel Time Between DCUs (Minutes)</u>			
	1	2	3		1	2	3
1	-	1.28	3.5	1	-	1.25	3.38
2	1.03	-	2.25	2	0.89	-	2.04
3	3.74	2.41	-	3	3.33	2.41	-
<u>Total Number of Vehicles Detected at each DCU</u>				<u>Total Number of Vehicles Detected at each DCU</u>			
	1	2	3		1	2	3
	872	882	694		633	814	716
<u>Total number of unique vehicles detected at each DCU*</u>				<u>Total number of unique vehicles detected at each DCU</u>			
	1	2	5		1	2	3
	527	562	442		419	510	446

*The same MAC address detected on different days is only counted once.

Table 5.3 Travel Time Data Collection Summary for Highway 99W, 4PM – 7PM

<u>Sept. 17, 2013 (16:00-19:00) to Sept.19, 2013 (16:00-19:00)</u>				<u>Oct. 1, 2013 (16:00-19:00) to Oct. 3, 2013 (16:00-19:00)</u>			
<u>Number of trips between DCUs</u>				<u>Number of trips between DCUs</u>			
	1	2	3		1	2	3
1	-	531	242	1	-	467	203
2	414	-	275	2	231	-	218
3	329	295	-	3	164	256	-
<u>Average Travel Time Between DCUs (Minutes)</u>				<u>Average Travel Time Between DCUs (Minutes)</u>			
	1	2	3		1	2	3
1	-	1.12	3.17	1	-	1.22	3.76
2	1.05	-	2.03	2	0.98	-	2.42
3	4.39	2.95	-	3	4.37	3.24	-
<u>Total Number of Vehicles Detected at each DCU</u>				<u>Total Number of Vehicles Detected at each DCU</u>			
	1	2	3		1	2	3
	1348	1419	1339		888	1179	1260
<u>Total number of unique vehicles detected at each DCU*</u>				<u>Total number of unique vehicles detected at each DCU</u>			
	1	2	5		1	2	3
	1016	1023	961		716	911	920
*The same MAC address detected on different days is only counted once.							

5.6.1 Conclusions

The results of the intersection test show that the portable DCUs have the potential to collect travel time data that can be used to evaluate different traffic system changes. In the test performed the travel time data is being used to evaluate signal timing changes. The data collection was automatic and thus required manual setup and takedown, and a single battery change to ensure continuous data collection over the desired 72 hour period.

5.7 Dedicated Short Range Communication

Dedicated short-range communication (DSRC) is a wireless technology which is designed to support a variety of applications based on vehicular communication (USDOT, 2006). The word “Dedicated” in DSRC refers to the fact that the U.S. Federal Communications

Commission has allocated 75 MHz of licensed spectrum in the 5.9 GHz band for DSRC communication (FCC 1998, 2003).

The primary motivation for deploying DSRC is to enable collision prevention applications. Collision prevention applications depend on frequent data exchanges among vehicles, and between vehicles and roadside infrastructure. However, DSRC can be used for many other applications beyond collision avoidance. Most of these involve communication to and from road side units (RSUs). For example, DSRC can be used to assist navigation, make electronic payments (e.g., tolls, parking, fuel), improve fuel efficiency, gather traffic probe data, and disseminate traffic updates (Kenney 2011).

The objective in this project was to investigate the potential of using DSRC technology to support the automatic collection of arterial operations performance data by serving as the main platform to support point detection, and vehicle re-identification data collection methods. For the specific purposes of this project, this potential was judged from two perspectives: *technology readiness* and *equipment availability and economic feasibility*. The conclusions reached are presented in the following sections.

5.7.1 Technology Readiness

Table 5.1 shows the most relevant specifications of DSRC in six categories: *standard(s)*, *operating frequency*, *application*, *range*, *data rates* and *latency*. Three common wireless technologies also used in vehicular applications (i.e., WiFi, Bluetooth, and ZigBee) are also shown to provide a reference for comparison.

Table 5.1 Most Relevant Specifications of DSRC and Other Wireless Technologies

	DSRC	WiFi	Bluetooth	ZigBee
Standard(s)	IEEE 802.11p (WAVE) – PHY & MAC IEEE 1609 (middle of protocol stack) SAE J2735 (message set dictionary)	IEEE 802.11	IEEE 802.15.1	IEEE 802.15.4
Operating Frequency	5.9 GHz	2.45 GHz	2.45 GHz	2.45 GHz (Global) 915 MHz (Americas) 868 MHz (Europe)
Application	Vehicular communications: Vehicle-to-vehicle (V@V) Vehicle-to-Infrastructure (V2I)	Wireless version of a common wired Ethernet network (WLAN)	Exchanging data over short distances (WPAN)	Low data rate, long battery life, and secure networking
Range	100 ~ 1000 m	100 m (outdoors) 20 m (indoors)	Class 1 ~ 100 m Class 2 ~ 10 m Class 3 ~ 5 m	10 – 75 m Up to 1500 m (ZigBee PRO)
Data Rates	6 to 27 Mbps	1 to 866.7 Mbps	0.7 – 2.1 Mbps	20 kbps 40 kbps 250 kbps
Latency	200 □sec	>20 msec	~ 100 msec	30 msec or less

DSRC has a maximum range of 1000 meters within the current standards. Under most operating conditions, DSRC will be limited to less than 200 meters. This limitation is well-suited to most vehicular applications, especially those involving vehicle re-identification, since the infrastructure where the roadside readers would be installed are generally within these kinds of ranges. DSRC offers the capability of broadcast messages. This is a significant advantage over point-to-point wireless communications (e.g., cellular) for vehicle re-identification applications since roadside readers can communicate with multiple vehicles simultaneously in an attempt to capture identification and signal strength data (USDOT 2005). One of the most significant potential advantages of DSRC technology is its capability for very low latency communications.

Latencies in the 200 μ sec range seem to be possible with DSRC, and many of the vehicle re-identification applications will certainly benefit from this feature.

DSRC is a technology that is definitely ready to support vehicle re-identification applications. If utilized in a similar fashion as Bluetooth, extracting vehicle identification and signal strength data would be accomplished with the use of the message set dictionary defined in the SAE J2735 standard.

5.7.2 Equipment Availability and Economic Feasibility

To assess equipment availability and economic feasibility, an Internet search was performed to search for DSRC-compliant hardware development platforms that could be used to manufacture data collection units. Also, two vendors that offer DSRC system solutions were contacted and provided an informal quote.

The Internet search revealed that single board computers (SBC) compatible with the DSRC technology standard are not currently available in the marketplace. Reasons for this unavailability are that widespread DSRC deployments are still to occur. Therefore, the market for these devices does not exist. Given the unavailability of commercial-off-the-shelf (COTS) components to build custom data collection units, the DSRC system vendors Denso and Savari Networks were contacted next.

Denso offers a DSRC solution referred to as V2X. Roadside units cost approximately \$2,000 and on-board units cost approximately \$1,000 each. No specifications about the equipment were available at the time of contact. The lead time for equipment delivery was estimated to be 6 months (Berg 2012).

Savari Networks offers a DSRC technology solution referred to as MobiWAVE™, which encompasses a family of products with several on board equipment (OBE), including the Vehicle Awareness Device (VAD), the Automotive Safety Device (ASD), and the Modular Communications Platform (MCP), as well as a Software Development Kit (SDK) (Savari 2013). As an example, the unit cost for the ASD platform is \$4,000 and the RSU platform is \$10,000. The SDK to develop applications for the ASD/RSU platform costs \$25,000 (Ravi 2103). The lead time for equipment delivery was estimated to be 4 months.

Based on the information gathered via the Internet and provided by Denso and Savari Networks, it is clear that the costs involved with procuring a DSRC platform are extremely prohibitive. As applications of this technology become more widespread and market penetration improves, it is anticipated that the hardware and software costs of this technology will become more reasonable.

CHAPTER 6.0 EFFECTIVE TURNING MOVEMENT VOLUME ESTIMATION FOR INTERSECTION ANALYSIS USING GAUSS-JORDAN ELIMINATION

6.1 Introduction

Engineers need intersection traffic supply and demand information to design and operate traffic systems, and turning movement volumes are a key aspect of traffic demand. Knowledge of turning movement volumes facilitates better decisions. As a result, engineers need access to more economical turning movement counts acquired at more locations and with greater frequency. This paper presents a method that solves for turning movement volumes using Gauss-Jordan elimination row operations (e.g., row swapping, multiplying rows by non-zero constants, and adding a factor of one row to another row). The input data are phase status, lane-by-lane detector counts, and limited exit detector counts. It evaluates existing intersection detector locations for their combined suitability to estimate turning movements and selects detection plans that minimize data requirements. The method accommodates varying lane configurations, varying detector locations, and includes or excludes phase status. Because the method is founded on direct implementation of basic matrix analysis row operations, the solution process is easy to implement.

The paper first gives a background and then in the methodology section it provides a detailed description of the turning movement calculation algorithm, involving an example intersection. Then the paper discusses how the methodology determines feasible intersection configurations with and without phase information and how to choose new detector placements. Then validation and error sensitivity test results are presented. Finally, the paper gives conclusions and recommendations for future research.

6.2 Background

Traffic detectors have a variety of applications. Local intersection control is the most predominant, where the intersection detectors communicate vehicle presence to the traffic controller, which then changes its state to accommodate newly arrived vehicles. System detection is another application, where mid-block detectors collect data that, together with intersection detectors, informs performance measures and/or system traffic control strategies. In recent years, technology advances expanded system detection capabilities to include vehicle tracking either in the form of magnetic vehicle reidentification, probe vehicle tracking, or video vehicle tracking. Unfortunately, these advances have not succeeded in economically acquiring turning movement volume data. One reason for this shortcoming is reidentification and probe vehicle technologies both can only sample a portion of vehicles. In addition, high capital costs limit the application of all of these technology advances, forcing the large majority of intersection analysis to rely on labor-intensive manually collected turning movement count data.

While turning movement count data collection using automated means is still not widely available, it is still useful to clarify the availability of detection that would prove useful for calculating turning movement counts. Many previous turning movement estimation efforts utilized lane-by-lane entry and exit counts and this paper depends on these data as well (Davis et. al. 1995; Nihan et. al. 1989; Dixon et. al. 2007). Stop bar entry counts can be lane-by-lane. However, it should be noted that detection standards tend to vary from agency to agency, but it is safe to say that stop bar detection is fairly common at signalized intersections. Lane-by-lane detection is less common, but desirable for more accurate detection and signal operations (Yuan,

Y. et. al. 2013). Exit detectors are much less common and are primarily used for system detection purposes, not for local intersection control.

The proposed method possesses several advantageous attributes, which are listed below, and the following literature review discusses previous methods relevant to each of them.

- Analysis scope,
- Detector location,
- Use of detector and phase status event data,
- Input data, and
- Estimate mathematical form.

6.2.1 Analysis Scope

Different methods for estimating turning movements require different analysis scopes and these can be categorized by generalized network approaches and intersection input-output approaches. Some methods use general network topological relationships between detector points and turning movements and these are defined by vehicle paths through a network. These vehicle paths help correlate changes in volume collected at a specific location to changes in turning movement volumes at other locations. As a result, detector locations can vary and are not limited by requiring complete instrumentation at all exit and entry lanes for a given intersection. However, these methods are complicated by their need to employ detection at key locations throughout the broader network in order to maximize information regarding each of the unknown turning movement volumes. In addition, these methods rely on mathematical solutions that are not generally used in engineering practice and require simplifying assumptions, such as possessing reliable vehicle paths, prior knowledge of origin-destination volumes, or extensive

detection requirements. Furthermore, although these methods provide valuable data for planning level applications, they lack the accuracy that operations and design decisions require (Chen et. al. 2012; Nakatsuji et. al. 2004; Lan et. al. 1999).

In comparison, intersection input – output methodologies are significantly simpler. Generally, these methods require detection data on all lanes entering and exiting an intersection and have a much more limited analysis scope. Because of the limited scope, these methods can be more reliable and executed with more common modeling and statistical techniques that rely on vehicle count multi-sampling or time-series analysis (Davis et. al. 1995; Nihan et. al. 1989; Cremer et. al. 1987; Martin 1997; Yi et. al. 2010; Tian et. al. 2004; Dixon et al. 2007).

6.2.2 Detector Location

Detector location is important for turning movement volume estimation and is key for determining which solution approach to take. Two basic strategies are taken for detector location. Entering – Exiting detection focuses on one intersection, where all entering and exiting traffic are observed by locating detectors on each entering and exiting lane. General network detection relaxes the detection requirement by enlarging its scope to include more than one intersection. Although the general network strategy does not require detection on all entering and exiting lanes, it does require detection at locations such that all path flows contributing to the turning movements are observed.

Entering – exiting detection strategies only require the analyst to specify the required turning movements to estimate (Nihan et al. 1989; Cremer et al. 1987). Others developed methods that worked with this same detection strategy, while allowing for incomplete counts (Davis et. al. 1995).

In contrast, the general network strategy requires the analyst to describe the surrounding network, the detector locations, network street characteristics (i.e., lengths, speeds, etc.), origins and destinations, and a means to relate traffic traveling between a given origin and destination to traffic counts at detector locations (Chen et al. 2012; Nakatsuji et al. 2004; Lan et al. 1999).

6.2.3 Use of detector and phase status event data

Research has shown that a vehicle's turn movement can be inferred if intersection detectors are situated at entrances and exits such that they allow accurate time correlation between vehicles arriving at an entrance detector and vehicles arriving at an exit detector. List et al. showed this was the case for roundabouts and was able to achieve very accurate results using travel time estimates between entrances and exits (List et al. 2006). Signalized intersections were analyzed in similar fashion, where phase status events were used to correlate entrance and exit events (Tian et al. 2004; Yi et al. 2010).

6.2.4 Input data

All turning movement volume estimation methodologies required different detector aggregations. Most methods require vehicle detection aggregated into volumes. Only those methods that base their turning movement estimation on the connection of individual detection events require disaggregate detection data (List et al. 2006; Tian et al. 2004; Yi et al. 2010).

Path-based methods require vehicle path information, defining the set of links vehicles will use to travel from an origin to a destination. For a small linear network, there is only one possible path for each origin-destination pair and the relationship between a detector's counts and path flows is clear. However, as the network size increases, two complexities arise. The

number of possible paths increases and the relationship of a detector's counts during one interval to path flows during another interval changes as a function of travel time.

6.2.5 Estimate process

Because of the range of characteristics defining methods proposed in the literature, this discussion refers to each of these proposed methods as estimation processes. Consequently, a process is defined by its mathematical approach and its underlying requirements regarding input data.

Previous processes use a variety of mathematical techniques to estimate turning movement counts for small areas for which most of the trip length is outside the area of interest. Cascetta et al. (1988) identified three categories: maximum likelihood estimators, Generalized least squares estimators, and Bayesian estimators. Two more categories exist: recursive estimators (Cremer 1987; Nihan 1987; Okutani 1987; Chang 1994; Ashok 1996; Dixon 2002) and heuristic estimators (Nakatsuji et. al. 2004; List et. al. 2006). All of the estimators in each of these categories utilize a procedure to optimize an objective function that includes minimizing the difference between the estimated link volumes and the observed link volumes. Regression is the simplest of these estimators (Robillard et. al. 1975), but none estimate volumes for conventional intersections by simply manipulating and solving simultaneous equations.

Processes using raw individual vehicle detections, tally individual turning movements, depending on the pair of entrance and exit detections that the process deems most likely (List et. al. 2006). However, all processes using aggregated input data estimate turning movement volume in an aggregate form.

Some processes assume fewer independent detectors than the number of unknown turning movements and require detector data from more than one interval to use regression (Nihan et. al. 1989; Gajewski et. al. 2002). These processes produce estimates that are averages for the input data intervals. Other process employ recursive estimation techniques, where a prior solution is updated with each time interval's data (Cremer 1987; Nihan 1989).

The regression and recursive methods have been used for cases as small as estimating turning movements for one intersection to estimating origin-destination volumes for general networks. In general, more sophisticated methods primarily address a general network analysis scope (Lan 1997; Nakatsuji 2004; Chen 2012).

6.2.6 Summary

Turning movement estimation methods usually have extensive input data requirements, employ a sophisticated estimation process, and have insufficient accuracy for design or operations. This paper proposes a method that assumes the possibility of lane-by-lane detection at intersection entrances and exits, but minimizes the detection requirements leading to an economical estimation process. In so doing, the method achieves accuracy that is directly controlled by the user by way of detection accuracy. If the detector data are very accurate and the detector locations yield independent equations equal in number to the number of unknowns then the turning movement volume estimation problem can be accurately solved.

6.3 Methodology

This method uses lane-by-lane vehicle counts and intersection geometry to create a matrix of detectors versus turning movements to solve for turning movement counts. If phase information is available then this method can separate the turning movements and counts into

concurrency groups. In concept, a concurrency group is a mechanism to group phases and the counts of their corresponding turning movements into portions of the cycle during which they can occur. Counts included in a given concurrency group occur from the time the first phase in the group is active until the end of the all-red interval for the last phase served in the group. For example, an eight phase intersection conveniently arranges into two concurrency groups defined by the barriers contained in a ring-barrier diagram (group 1: NBL, SBT, SBL, NBT; group 2: EBL, WBT, WBL, EBT). During a group's portion of the cycle, detectors will record volumes resulting from movements executed during its corresponding phases, including right-turn-on-red movements. This results in two sets of counts for the detectors, one for each concurrency group. More than two concurrency groups may improve solution feasibility and this method allows the user to define the number of concurrency groups.

The method relies on an important distinction between a turn movement and a Lane Specific Movement (LSM). A LSM specifically defines a turn by the entrance lane of an approach and the exit lane of an exit. For example, a through movement served by two lanes would include two LSMs, one from the inside lane at the approach to the inside lane at the exit and the other LSM from the outside lane at the approach to the outside lane at the exit. A turn movement is the aggregation of the corresponding LSMs (e.g., Left turn, Through, and Right).

Turning movement counts for each concurrency group are solved separately and the results summed across groups to produce a final result of overall estimated turning movement counts. The phase information can reduce the number of detectors needed to solve some intersections and may be necessary to solve others. There are some intersections that this method cannot solve even with phase information; however they are not the norm.

This method can work with unsignalized intersections as well. Because there is no signal, it is treated the same as a signalized intersection without phase information. Most unsignalized intersections have too few turn pockets to be solved. However, some will be feasible, especially three legged intersections.

This method requires detectors obtaining vehicle counts on many of the lanes, but usually not all. The section, Identifying Solvable Intersection Configurations and New Detector Placement, describes a method for determining the most strategic set of detector locations.

6.3.1 Overall Method Description

Figure 6.1 contains information for an example application of the proposed method. On the top right, a figure illustrates all possible lane-by-lane detector locations, accompanied by lane numbers. On the top left, two elements of information are given in a table. The first element establishes the concurrency groups into which phases are grouped. Second, below the concurrency grouping, are the counts for each lane that has a detector. In this example, lane four and six do not have detectors. On the bottom right, turning movements and LSMs are defined in terms of their exit and entrance lanes.

Note that the proposed method allows many different detection options to suit varying intersection configurations. This simplified intersection configuration and detection layout was intentionally chosen to allow a demonstration of key aspects of the proposed method.

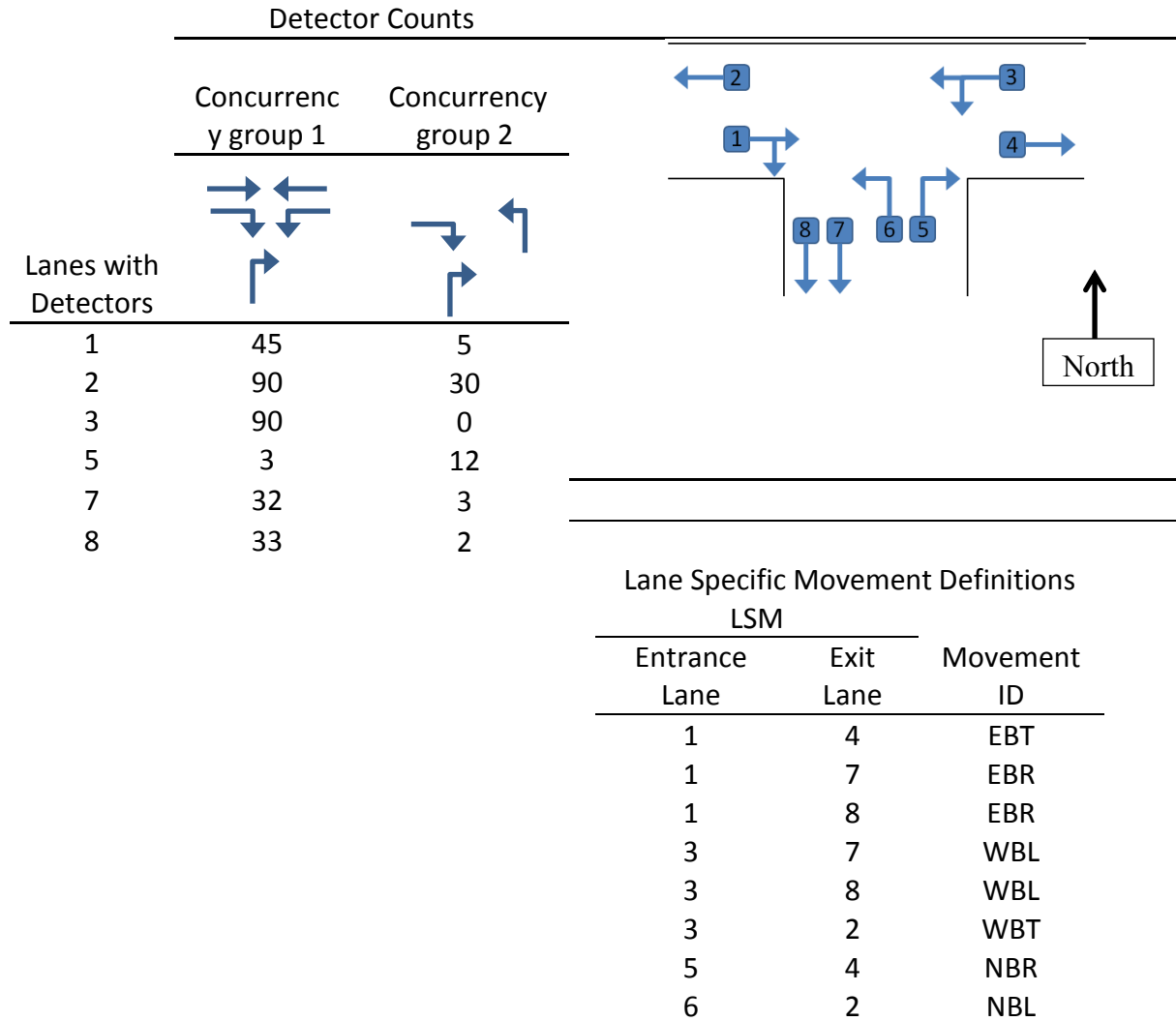


Figure 6.1 Intersection Concurrency Groups, Counts, Lane Numbering, and Lane Specific Movement Definitions

The following describes the 12-step process to estimate turn movements using the intersection shown in Figure 6.1 as an example to illustrate key points.

Step 1: Calculate the turning movement counts occurring during the green time of one concurrency group at a time. Each detector needs to have counts for each concurrency group (see Figure 6.1). If this phase information is not known and only total detector counts are known, treat the intersection as if it had only one concurrency group. This step results in detector counts aggregated by concurrency group.

Step 2: Associate detectors with LSMs for all concurrency groups. As defined previously, an LSM is defined by the entrance lane, the exit lane, and the movement ID. The LSM for the example problem are shown in Figure 6.1, bottom right. The example presented in this paper works under the assumption that right turns and left turns may cross the detectors in either lane 7 or 8. In addition, right turns are allowed on red.

Step 3: Create a list of all possible Exit Detector Combinations (EDC). An EDC is a group of two or more exit lanes with detectors on the same intersection leg. The EDC will be the same for each concurrency group. There is only one possible exit detector combination for the example intersection, which is 7/8. However, a three lane exit would have three possible combinations and result in three EDCs. Moreover, each intersection leg with multiple exit detectors will contribute more EDC to this list. Each EDC represents an equation, an alternative to its corresponding individual detector equations. Opting for an EDC alternative may be necessary for a feasible solution.

Step 4: Select a concurrency group. This step begins the loop through the concurrency groups, repeating Steps 4 through 12.

Step 5: Create the “a0” matrix for the concurrency group selected in Step 4 that has a row for each individual detector, a row for each EDC in the EDC list, and a column for each LSM contributing to the concurrency group’s counts. Even though the current execution of this step is for concurrency group 1, the “a0” matrices for both concurrency groups are given in Figure 6.2 for the sake of discussion.

For concurrency group 1, refer to the table on the left. Notice that the no NBT movements can occur during concurrency group 1. Consequently, it is not included in the concurrency group 1 “a0” matrix. The table on the right is the concurrency group 2 “a0” matrix and will be used in the next loop, when solving for the turning movement volumes occurring during the second concurrency group. For the second concurrency group, only the EBR movement is included from the eastbound and westbound approaches.

The table cells of 1’s and 0’s represent detector/movement combinations. If a movement crosses a detector then the corresponding cell contains a “1”, otherwise it contains a “0”. The EDC rows follow the detector rows. In this case, there is only one, the combination of detectors 7 and 8. In the case of the EDC rows, each cell represents an EDC/movement combination, which should contain a “1” if the movement crosses any of the detectors contained in the EDC. In this example, the row for the EDC element of 7/8 will be one if a movement crosses detector 7 or 8 and a “0” otherwise (see Figure 6.2). Only the coefficients and counts comprise an “a0” matrix, the rest of the rows and columns are for row and column labeling purposes.

		Concurrency Group 1 LSM							Concurrency Group 2 LSM					
		EBR	EBR	EBT	WBT	WBL	WBL	NBR			EBR	EBR	NBL	NBR
Lanes with Detectors	Group 1 Counts	1 to 8	1 to 7	1 to 4	3 to 2	3 to 7	3 to 8	5 to 4	Lanes with Detectors	Group 2 Counts	1 to 8	1 to 7	6 to 2	5 to 4
1	45	1	1	1	0	0	0	0	1	5	1	1	0	0
2	90	0	0	0	1	0	0	0	2	30	0	0	1	0
3	90	0	0	0	1	1	1	0	3	0	0	0	0	0
5	3	0	0	0	0	0	0	1	5	12	0	0	0	1
7	32	0	1	0	0	1	0	0	7	3	0	1	0	0
8	33	1	0	0	0	0	1	0	8	2	1	0	0	0
7/8	65	1	1	0	0	1	1	0	7/8	5	1	1	0	0

Figure 6.2 Example Matrix “a0” (first concurrency group and second concurrency group)

Step 6: Given the list of EDCs from Step 5, create all possible unique combinations of the EDCs and arrange them in order of increasing complexity. The discussion refers to each combination of EDC as an EDC set. Intersections with three lane approaches or several multi-lane approaches will have several EDCs in the EDC list (established in Step 3) and these can be arranged into unique EDC sets listed in order of increasing complexity.

Step 7: For the concurrency group selected in Step 4, this step manages looping through different EDC sets. The algorithm attempts to solve for the turning movement volumes beginning with the first EDC set and stops when a solution is found. It is important to list the smallest EDC sets first, so that the simplest solutions are attempted first. For the given example, two sets of EDCs would be possible: none and 7/8.

Lanes with Detectors		Group 1 Counts		Concurrency Group 1 LSM						
				EBR	EBR	EBT	WBT	WBL	WBL	NBR
		1 to 8	1 to 7	1 to 4	3 to 2	3 to 7	3 to 8	5 to 4		
1	45	1	1	1	0	0	0	0		
2	45	0	0	0	1	0	0	0		
3	90	0	0	0	1	1	1	0		
5	3	0	0	0	0	0	0	1		
7	32	0	1	0	0	1	0	0		
8	33	1	0	0	0	0	1	0		

Lanes with Detectors		Group 1 Counts		Concurrency Group 1 LSM						
				EBR	EBR	EBT	WBT	WBL	WBL	NBR
		1 to 8	1 to 7	1 to 4	3 to 2	3 to 7	3 to 8	5 to 4		
1	45	1	1	1	0	0	0	0		
2	45	0	0	0	1	0	0	0		
3	90	0	0	0	1	1	1	0		
5	3	0	0	0	0	0	0	1		
7/8	65	1	1	0	0	1	1	0		

a) “a” matrix for the EDC set of “none”

b) “a” matrix for the EDC set of “7/8”

Figure 6.3 Group 1 “a” matrices

Step 8: Create the “a” matrix, given the EDC set selected in Step 7, which will have the same columns as matrix “a0” but will only contain a subset of the rows. In Step 7, the algorithm started with the simplest set of EDCs, which is “none”. As shown in

Lanes with Detectors	Group 1 Counts	Concurrency Group 1 LSM						
		EBR 1 to 8	EBR 1 to 7	EBT 1 to 4	WBT 3 to 2	WBL 3 to 7	WBL 3 to 8	NBR 5 to 4
1	45	1	1	1	0	0	0	0
2	45	0	0	0	1	0	0	0
3	90	0	0	0	1	1	1	0
5	3	0	0	0	0	0	0	1
7	32	0	1	0	0	1	0	0
8	33	1	0	0	0	0	1	0

Lanes with Detectors	Group 1 Counts	Concurrency Group 1 LSM						
		EBR 1 to 8	EBR 1 to 7	EBT 1 to 4	WBT 3 to 2	WBL 3 to 7	WBL 3 to 8	NBR 5 to 4
1	45	1	1	1	0	0	0	0
2	45	0	0	0	1	0	0	0
3	90	0	0	0	1	1	1	0
5	3	0	0	0	0	0	0	1
7/8	65	1	1	0	0	1	1	0

c) “a” matrix for the EDC set of “none”

d) “a” matrix for the EDC set of “7/8”

Figure 6.3, the “a” matrix only includes rows corresponding to detectors 1, 2, 3, 5, 7, and 8, excluding any EDCs. The “a” matrix represents a system of equations. In this case, there are six equations, where the Group 1 counts constitute the equation solutions and the cells to the right of the count contain the equation coefficients for the unknown LSM counts. There are seven unknowns, so it is not possible to solve for all LSMs, because there are more unknowns than there are equations. It is important to proceed to Step 9 to address this issue.

Step 9: Delete all but one of any columns that have the same value in every row of the coefficients matrix and contribute to the same turning movement. Then check that the number of equations equals the number of unknowns. After this step, there should be no two columns that are identical, which is equivalent to eliminating redundant variables. For the “a” matrix using the EDC set of “none”, no columns can be deleted (see Figure 6.3a). Consequently, there are six equations and seven unknowns, so the current “a” matrix is not feasible and the algorithm returns to Step 7 to select the next EDC set of “7/8”.

The algorithm then proceeds to Step 8 to create the next “a” matrix with the “7/8” EDC set. Step 8 then excludes the rows for detectors 7 and 8 (see Figure 6.3b). Excluding these equations created two sets of columns that are identical. The two EBR columns are identical and so are the WBL columns. These duplicate columns correspond to variables that became redundant by adding the equation for EDC “7/8” and excluding the equations for detectors 7 and 8. This step (Step 9) removes one of each of the duplicate columns, creating a system of five equations and five unknowns (see Figure 6.4a). Step 10 will determine if these equations are independent and yield a feasible solution.

Step 10: Use Gauss-Jordan elimination row operations to simplify matrix “a” coefficients to achieve an identity matrix and the turning movement count solutions (see Figure 6.4b). The value in the “Group 1 Counts” column is the solution for the turning movement in the column that contains a 1 in the value’s same row. For example, the value “25” is the solution for the EBT corresponding to the LSM going from detector 1 to 4. The Gauss-Jordan process was augmented to delete all zero rows, ensure the fewest row operations, and minimize the variance in the turning movement count errors. The appendix “Simplifying the Matrix” explains this in more detail. If the matrix can be simplified to a feasible solution, then this step results in an identity matrix.

If the “a” matrix can’t be simplified then the current EDC set does not work. If another EDC set exists then the process loops back to Step 7 to select the next EDC set to evaluate its feasibility.

		Concurrency Group 1 LSM				
		EBR	EBT	WBT	WBL	NBR
Lanes with Detectors	Group 1 Counts	1 to 7/8	1 to 4	3 to 2	3 to 7/8	5 to 4
1	45	1	1	0	0	0
2	45	0	0	1	0	0
5	3	0	0	0	0	1
3	90	0	0	1	1	0
7/8	65	1	0	0	1	0

		Concurrency Group 1 LSM				
		EBR	EBT	WBT	WBL	NBR
Group 1 Counts		1 to 7/8	1 to 4	3 to 2	3 to 7/8	5 to 4
20		1	0	0	0	0
25		0	1	0	0	0
45		0	0	1	0	0
45		0	0	0	1	0
3		0	0	0	0	1

a) Reduced “a” matrix from Step 7

b) Simplified matrix and solution from Step 8

Figure 6.4 Matrix input to Step 8 from Step 7 and Matrix resulting from Step 8

Because it was possible to simplify matrix “a”, the given detector configuration can be solved for all of the turning movements and Step 11 should be next. However, if Step 10 failed (matrix “a” was unable to be simplified) then the process returns to Step 7 to select the next EDC set in the list.

Step 11: If none of the EDC sets from Step 3b result in feasible solutions then the given detector configuration cannot be used to calculate turning movement counts. See the following section for more on determining where to place detectors.

Step 12: If more concurrency groups remain then return to Step 4 to select the next one. This step maintains running sums for each turning movement. Results from the first concurrency group are the first to contribute to these sums. Because there is one more concurrency group in this example, this method loops back to step 4, to repeat the process for the next concurrency group.

After completing the steps for the second concurrency group in this example, the resulting turning movement count estimates are complete and they are given in Table 6.1.

Table 6.1 Turn Movement Estimates for Concurrency Groups 1 and 2

Movements	Concurrency Group Counts		Total
	Group 1	Group 2	
NBR	15	5	20
NBT	15	0	15
NBL	10	0	10
WBR	5	10	15
WBT	0	20	20
WBL	0	25	25
EBR	5	25	30
EBT	0	25	25
EBL	0	20	20
SBR	10	5	15
SBT	10	0	10
SBL	15	0	15

This method uses Gauss-Jordan elimination to solve a system of equations for the LSM counts. An equation is written for each detector saying that the number of vehicles that crossed

the detector equals the sum of the vehicles from each LSM that passes over the detector. The equations for each of the detectors are combined into a matrix built in this method (Matrix “a”). Then row operations are used to solve for the LSM counts. Maintaining the direct use of row operations to solve the problem, instead of matrix analysis functions, provides two benefits stemming from the interjection of different detection options. One benefit is being able to solve for the turning movement volumes relying on the fewest detectors. Another benefit is finding the detection layout with the fewest detectors that would result in a solution.

The two additional actions that need additional explanation are the deletion of one of two identical columns of the matrix and the use of the EDC list. By deleting an identical column, the process changes the solution space to solve for the sum of two LSM counts that are for the same turn movement, rather than individually. For example, refer to the WBL turn movement of the intersection shown in Figure 6.1. Aggregating exit detectors 7 and 8 removes the need to solve for the counts for LSMs 3 to 7 and 3 to 8 separately, because both LSMs complete the count of the westbound left turn. Instead, all that is needed to find the total westbound left turns is to know the sum 3 to 7 and 3 to 8. This is because if detectors 7 and 8 were aggregated then the two LSMs would have the same coefficients in every equation. So, they can be factored together, thus allowing solving for the sum of the two LSMs (e.g., $2a+2b=4 \Rightarrow 2(a+b)=4 \Rightarrow a+b=2$). To factor the two movements together, the algorithm would delete one of the two LSM columns (3 to 7 or 3 to 8), and the remaining column would represent the sum of the two LSMs (e.g., 3 to 7/8) (see Figure 6.4b).

The purpose of an EDC is to aggregate detectors on the same exit into one effective detector. Detector aggregation results in LSMs with identical columns in the matrix.

Subsequently, through column deletion, the method reduces unknowns, increasing turn movement solution feasibility. However, the following conditions must be observed for detector aggregation at a specific exit to increase solution feasibility.

- First, there needs to be more than one exit detector at the exit.
- Second, two or more turn movements must cross the exit detectors.
- Third, each turn movement crossing one of the exit detectors being aggregated must have two or more LSMs.
- Fourth, for two or more of these turn movements, the LSMs must share the same entrance detector or not have an entrance detector.

For situations satisfying these four conditions, aggregating the exit detectors makes the LSM columns identical for a given turn movement, allowing one column to be deleted. So, by aggregating the two exit detectors, the matrix loses one row for each detector that will be aggregated and gains a row for the aggregated detector. Overall, this reduces the number of equations by one. However, if the above four criteria are met then aggregating detectors will eliminate at least two columns, one for each turning movement entering the intersection from a single lane and that crosses the aggregated detector. In this way, the algorithm increases intersection feasibility through detector aggregation.

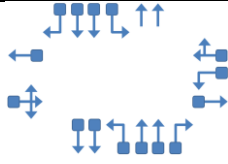
























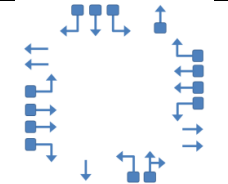
























In order to ensure that no opportunity is missed to make an otherwise unsolvable intersection solvable, every combination of multiple exit detectors for the same travel direction is compiled, creating the EDC list. Then every combination of elements of the EDC list (EDC sets) is tried until a solution is found, to see if any combination of exit detectors will allow the matrix to be solved.

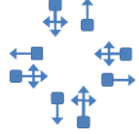
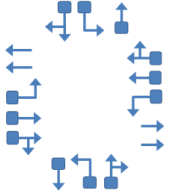
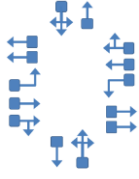
6.3.2 Identifying Solvable Intersection Configurations and New Detector Placements

Many different counting scenarios exist for existing intersections. It is not possible to enumerate all intersection configurations and their corresponding feasible counting scenarios.

Table 6. shows the detector locations for five different example intersections. These examples are based on existing intersection configurations in Moscow, Idaho. The first two intersections can be solved with no phase information (i.e. one concurrency group) or with phase information (two concurrency groups). Intersections C and D can only be solved if phase information is available. Intersection E cannot be solved even with phase information. Each intersection is shown with the minimum number of exit detectors needed to solve for the turning movement counts (assuming that all entrance lanes have detectors).

Table 6.2 Turn Movement Count Feasibility Examples – Minimum Detection

Intersection	Minimum Detection Diagram	Possible Phase Grouping	Solvable with phase data	Solvable without phase data												
A		<table border="1" style="width: 100%; text-align: center;"> <tr> <th colspan="2">Concurrency Group 1</th> <th colspan="2">Concurrency Group 2</th> </tr> <tr> <td>Phase 1 </td> <td>Phase 2 </td> <td>Phase 3 </td> <td>Phase 4 </td> </tr> <tr> <td>Phase 5 </td> <td>Phase 6 </td> <td>Phase 7 </td> <td>Phase 8 </td> </tr> </table>	Concurrency Group 1		Concurrency Group 2		Phase 1 	Phase 2 	Phase 3 	Phase 4 	Phase 5 	Phase 6 	Phase 7 	Phase 8 	Yes	Yes
Concurrency Group 1		Concurrency Group 2														
Phase 1 	Phase 2 	Phase 3 	Phase 4 													
Phase 5 	Phase 6 	Phase 7 	Phase 8 													
B		<table border="1" style="width: 100%; text-align: center;"> <tr> <th colspan="2">Concurrency Group 1</th> <th colspan="2">Concurrency Group 2</th> </tr> <tr> <td>Phase 1 </td> <td>Phase 2 </td> <td>Phase 3 </td> <td>Phase 4 </td> </tr> <tr> <td>Phase 5 </td> <td>Phase 6 </td> <td>Phase 7 </td> <td>Phase 8 </td> </tr> </table>	Concurrency Group 1		Concurrency Group 2		Phase 1 	Phase 2 	Phase 3 	Phase 4 	Phase 5 	Phase 6 	Phase 7 	Phase 8 	Yes	Yes
Concurrency Group 1		Concurrency Group 2														
Phase 1 	Phase 2 	Phase 3 	Phase 4 													
Phase 5 	Phase 6 	Phase 7 	Phase 8 													

C		<table border="1"> <tr> <th colspan="2">Concurrency Group 1</th> <th colspan="2">Concurrency Group 2</th> </tr> <tr> <td>Phase 1</td> <td>Phase 2</td> <td>Phase 3</td> <td>Phase 4</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> </table>	Concurrency Group 1		Concurrency Group 2		Phase 1	Phase 2	Phase 3	Phase 4					Yes	No								
Concurrency Group 1		Concurrency Group 2																						
Phase 1	Phase 2	Phase 3	Phase 4																					
D		<table border="1"> <tr> <th colspan="2">Concurrency Group 1</th> <th colspan="2">Concurrency Group 2</th> </tr> <tr> <td>Phase 1</td> <td>Phase 2</td> <td>Phase 3</td> <td>Phase 4</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Phase 5</td> <td>Phase 6</td> <td>Phase 7</td> <td>Phase 8</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> </table>	Concurrency Group 1		Concurrency Group 2		Phase 1	Phase 2	Phase 3	Phase 4					Phase 5	Phase 6	Phase 7	Phase 8					Yes	No
Concurrency Group 1		Concurrency Group 2																						
Phase 1	Phase 2	Phase 3	Phase 4																					
Phase 5	Phase 6	Phase 7	Phase 8																					
E		<table border="1"> <tr> <th colspan="2">Concurrency Group 1</th> <th colspan="2">Concurrency Group 2</th> </tr> <tr> <td>Phase 1</td> <td>Phase 2</td> <td colspan="2">Phase 3</td> </tr> <tr> <td></td> <td></td> <td colspan="2"></td> </tr> <tr> <td>Phase 4</td> <td>Phase 5</td> <td colspan="2">Phase 6</td> </tr> <tr> <td></td> <td></td> <td colspan="2"></td> </tr> </table>	Concurrency Group 1		Concurrency Group 2		Phase 1	Phase 2	Phase 3						Phase 4	Phase 5	Phase 6						No	No
Concurrency Group 1		Concurrency Group 2																						
Phase 1	Phase 2	Phase 3																						
Phase 4	Phase 5	Phase 6																						

Note: RTOR are also accommodated, but are not explicitly shown in the concurrency grouping diagrams.

As a rule of thumb, intersections without phase information that fit one of the following descriptions are feasible.

- two one-way roads.
- a two-way street and a one-way street with at least one turn pocket.
- two two-way streets are frequently feasible if they have at least 5 turn pockets.

Signalized intersections that don't fit the above should be checked to see if they can be solved with phase information.

The street network for Moscow, Idaho was examined to see the frequency with which the proposed method can yield a feasible solution. Of the 17 signalized intersections, 11 are feasible without phase information and 4 additional ones are feasible with phase information. On the other hand, of the 548 unsignalized intersections, only 27 are feasible.

The twelve steps described previously can be used to identify new detector placement in order to make an existing configuration solvable. This is done by starting the twelve steps with the existing configuration and iterating through different configurations until a solution is found. Based on the analysis of Moscow, Idaho intersections, typically only one additional detector is needed for signalized intersections and two for unsignalized intersections.

6.3.3 Method Validation

The proposed method's capability to produce accurate results was determined using field data with randomly generated error. The validation used data from two sources. One consisted of data manually extracted from video recorded at the signalized intersection of State Highway 8 and Farm in Moscow, Idaho. The second was taken from previous research (Smaglik 2007; MioVision 2013). For all three intersections, two types of data were collected, where one was spot counts at the stop bar and spot counts at the exit lanes. Another was turn movement counts (the ground truth data). Errors were randomly generated for the stop bar and exit count data to create detection errors according to ranges documented in previous research (Smaglik, 2007).

The proposed method was applied to each intersection in three ways: one without phase information, the second with phase information and no detection errors, and the third with phase information and with detection errors. If the intersection configuration and given phase information afforded a feasible solution, eleven trials were run. The first trial used the true detector counts and the remaining ten trials had random errors applied to the detector counts, with random errors varying within the ranges shown in the list below (Smaglik, 2007).

- Left turn lane detectors: 0.5% to 13.7%
- Through lane detectors: 4.4% to 23.3%
- Right turn lane detectors: 1.1% to 6.1%

The validation test results are given in Table 6.. As expected, for all three intersections, the trial run with no detection error, estimated the turn movement counts exactly, regardless of phase information. Naturally, Table 6. also shows that detection errors propagate through to the final estimates.

Table 6. summarizes the results for situations with detection error, with and without phase information. Overall, the proposed method estimates turn counts with Mean Absolute Percent Error (MAPE) in the approximate range of 9% to 13.6%, which is quite promising. Finally, the overall Mean Percent Error (MPE) is positive, in the range of 9% to 13%, indicating that the estimates are biased high. This bias occurred because the error ranges of the detectors were all positive, resulting in detectors that over counted.

In addition to increasing solution feasibility, including phase information also reduced the errors, as is evidenced by the validation data. As is shown Table 6., the MAPE changed by -

0.9%, 0.3%, and -1.4%, for the three intersections, respectively, indicating that, in addition to increasing feasibility, adding phase information possibly increases accuracy.

Table 6.3 Validation Test Results

Intersection	Error Measures	W/o Phase Information; No detection error	With Phase Information; No detection error	W/o phase Information; With detection error	With phase Information; With detection error
SH 8 and Farm	MPE	0	0	13.0%	12.7%
	MAPE	0	0	13.6%	12.7%
MioVision, 2013	MPE	0	0	11.1%	11.4%
	MAPE	0	0	11.1%	11.4%
Smaglik, 2007	MPE	0	0	9.8%	9.0%
	MAPE	0	0	10.4%	9.0%

6.3.4 Conclusions and Recommendations

The proposed method employs widely known Gaussian Elimination operations to search for detection arrangements and solve for turn movement volumes. Validation tests found that the method, while sensitive to detection errors, yields reasonable solutions that can be very accurate. In addition, the method can estimate turning movement volumes for most signalized intersections and some unsignalized intersections.

Several important distinctions characterize this method and they are as follows:

1. Solves quickly through Gauss-Jordan elimination,
2. Provides for lane-by-lane detection,
3. Integrates phase information,
4. Considers lane-specific movement as unknowns,
5. Leverages a straight forward procedure for aggregating exit detectors,
6. Simultaneously determines a minimum detection scenario, and

7. Selects detection to minimize estimate variance.

Future research should focus on two areas. One area is applying the methodology to freeway and urban street facilities. The second area is reducing error sensitivity by detection error filtering and integrating information from multiple time intervals by averaging or simple linear regression.

REFERENCES

- Alnowami, M., B. Alnwaimi; F. Tahavori; M. Copland and K. Well. A quantitative assessment of using the Kinect for Xbox360 for respiratory surface motion tracking. *Proc. SPIE Medical Imaging*, 2012, pp. 83161T-83161T
- Basso, F., M. Munaro, S. Michieletto, E. Pagello, and E. Menegatti. Fast and robust multi-people tracking from RGB-D data for a mobile robot. *Intelligent Autonomous Systems*, Springer Berlin Heidelberg, 2013, pp. 265-276.
- Berg, Roger (Personal communication), Vice President, Wireless Technologies, Denso. April 19, 2012.
- Boulos, M. N. K., B. J. Blanchard, C. Walker, J. Montero, A. Tripathy, and R. Gutierrez-Osuna. Web GIS in practice X: a Microsoft Kinect natural user interface for Google Earth navigation. *International Journal of Health Geographics* Vol. 10(45), 2011.
- Brandes, U. A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology*, Vol. 25, 2001, pp. 163–177.
- Brandes, U. On Variants of Shortest-path Betweenness Centrality and Their Generic Computation. *Social Network*, Vol. 30, 2008, pp. 136–145.
- Brennan, T.M., et al., (2011). “Visual Education Tools to Illustrate Coordinated System Operation” Transportation Research Board, National Research Council, Washington, DC.
- Broach, J., J. Dill, and J. Gliebe. Where Do Cyclists Ride? A Route Choice Model Developed with Revealed Preference GPS Data. *Transportation Research Part A*, 46, 2012, pp. 1730–1740.
- Buehler, R. and J. Pucher. Cycling to work in 90 large American cities: new evidence on the role of bike paths and lanes. *Transportation*, Vol. 39, 2012, pp. 409-432.
- Callister, D. and Lowry, M. Tools and Strategies for Wide-Scale Bicycle Level of Service Analysis. *Journal of Urban Planning and Development*. Vol. 139, No. 4. 2013.
- Charreyron, S., S. Jackson, L. F. Miranda-Moreno. Towards a Flexible System for Pedestrian Data Collection Using Microsoft Kinect Motion Sensing Device. 92nd Annual Meeting of the Transportation Research Board, Washington D.C. No. 13-3284, 2013.
- Chen, Anthony, Piya Chootinan, Seungkyu Ryu, Ming Lee, and Will Recker. “An Intersection

- Turning Movement Estimation Procedure Based on Path Flow Estimator.” *Journal of Advanced Transportation* 46, no. 2 (2012): 161–176. doi:10.1002/atr.151.
- Cremer, M., and H. Keller. “A New Class of Dynamic Methods for the Identification of Origin-destination Flows.” *Transportation Research Part B: Methodological* 21, no. 2 (1987): 117–132.
- Davis, G. A., and C. J. Lan. “Estimating Intersection Turning Movement Proportions from Less-than-complete Sets of Traffic Counts.” *Transportation Research Record* no. 1510 (1995): 53–59.
- Derrible, S. Using Network Centrality to Determine Key Transfer Stations in Public Transportation Systems. Presented at the 91st Annual Meeting of the Transportation Research Board, Washington D.C., 2012.
- Dill, J. Bicycling for Transportation and Health: The Role of Infrastructure. *Journal of Public Health Policy*, Vol. 30, 2009, pp. 95-110.
- Dixon, Michael P., Ahmed Abdel-Rahim, Michael Kyte, Phil Rust, Howard Cooley, and Lee Rodegerdts. “Field Evaluation of Roundabout Turning Movement Estimation Procedures.” *Journal of Transportation Engineering* 133, no. 2 (2007): 138–146.
- Eash, R. Destination and Mode Choice Models for Nonmotorized Travel. In *Transportation Research Record: Journal of the Transportation Research Board No. 1674*, TRB of the National Academies, Washington, D.C., 1999, pp. 1–8.
- Federal Communications Commission, Intelligent Transportation Services Report and Order, R&O FCC 99-305, Oct. 21, 1998.
- Federal Communications Commission, Dedicated Short Range Communications Report and Order, R&O FCC 03-324, Dec. 17, 2003.
- Gajewski, Byron J., Laurence R. Rilett, Michael P. Dixon, and Clifford H. Spiegelman. “Robust Estimation of Origin-destination Matrices.” *Journal of Transportation and Statistics* 5, no. 2/3 (2002): 37–56.
- Gandhi, T., and M. M. Trivedi. Pedestrian Protection Systems: Issues, Survey, and Challenges. Intelligent Transportation Systems. *IEEE Transactions*, Vol. 8.3, 2007, pp. 413-430.
- Griswold, J, A. Medury, and R. Schneider. Pilot Models for Estimating Bicycle Intersection Volumes. In *Transportation Research Record: Journal of the Transportation Research*

- Board No. 2247*, TRB of the National Academies, Washington, D.C., 2011, pp. 1–7.
- Hagani, A., H. Masoud, F.S. Kaveh, S. Young, and P. Tarnoff. Freeway Travel Time Ground Truth Data Collection Using Bluetooth Sensors. CD-ROM. Transportation Research Board of the National Academies, Washington, D.C., 2010.
- Harvey, F., Krizek, K. J., and Collins, R. (2008). “Using GPS Data to Assess Bicycle Commuter Route Choice.” Presented at the Transportation Research Board 87th Annual Meeting Washington D.C., 2008.
- Highway Capacity Manual 2010*. Transportation Research Board of the National Academies, Washington, D.C., 2010.
- Hillier, B., A. Penn, J. Hanson, T. Grajewski, and J. Xu. Natural movement: or, configuration and attraction in urban pedestrian movement. *Environment and Planning B*, Vol. 20, 1993, pp. 29–66.
- Hillier, B., and J. Hanson. *The Social Logic of Space*. Vol. 2. Cambridge University Press Cambridge, 1984.
- Hsieh, C.-T., H.-C. Wang, Y.-K. Wu, L.-C. Chang and T.-K. Kuo. A Kinect-Based People-flow Counting System. IEEE International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS), 2012.
- Hudson, J., T. Qu, and S. Turner. *Forecasting Bicycle and Pedestrian Usage and Research Data Collection Equipment*. Research Project Number P2009330, Report P2009330. Texas Transportation Institute; Capital Area Metropolitan Planning Organization; FHWA, U.S. Department of Transportation, 2010.
- Iacono, M., Krizek, K. J., and El-Geneidy, A. (2010). “Measuring non-motorized accessibility: issues, alternatives, and execution.” *Journal of Transport Geography*, 18(1), 133–140.
- Jones, M., S. Ryan, J. Donlon, L. Ledbetter, D. Ragland, and L. Arnold. *Seamless Travel: Measuring Bicycle and Pedestrian Activity in San Diego County and Its Relationship to Land Use, Transportation, Safety, and Facility Type*, PATH Report UCB-ITS-PRR-2010-12, 2010.
- Jiang, B., and C. Claramunt. “Topological Analysis of Urban Street Networks.” *Environment and Planning B: Planning and Design*, Vol. 31, 2004, pp. 151–162.

- Kenney, J.B. (2011). Dedicated Short-Range Communications (DSRC) Standards in the United States. *Proceedings of the IEEE*, Vol.99, No. 7, pp. 1162-1182.
- Kim, D.S., Porter, J.D., Magana, M.E., Park, S., and Saeedi, A. Wireless Data Collection System for Travel Time Estimation and Traffic Performance Evaluation. *Final Report ODOT SPR 737 OTREC-RR-012-06*, 2012.
http://www.oregon.gov/ODOT/TD/TP_RES/docs/Reports/2012/SPR737_Wireless.pdf.
- Lan, Chang-Jen, and Gary A. Davis. “Real-time Estimation of Turning Movement Proportions from Partial Counts on Urban Networks.” *Transportation Research Part C: Emerging Technologies* 7, no. 5 (October 1999): 305–327. doi:10.1
- Landis, B. Using the Latent Demand Score Model to Estimate Use. *ProBike/Pro Walk 96 Resource Book*. Presented at the Ninth International Conference on Bicycle and Pedestrian Programs, Portland, Maine, 1996.
- Larsen, J., and A. El-Geneidy. “A Travel Behavior Analysis of Urban Cycling Facilities in Montréal, Canada.” *Transportation Research Part D: Transport and Environment*, Vol. 16, 2011 pp. 172–177.
- Lee, B., Jennings, L., and El-Geneidy, A. “How Does Land Use Influence Cyclist Route Choice? Geospatial Analysis of Commuter Routes and Cycling Facilities.” Presented at the Transportation Research Board 90th Annual Meeting Washington D.C., 2011.
- List, G. F., and S. M. Eisenman. “Identifying Vehicle Trajectories and Turning Movements at Roundabouts.” In *5th International Symposium on Highway Capacity and Quality of Service*, 2006.
- Liu, H., X. Hu, S. Yang, K. Zhang, and E. Di Paolo. Application of Complex Network Theory and Genetic Algorithm in Airline Route Networks. In *Transportation Research Record: Journal of the Transportation Research Board No. 2214*, TRB of the National Academies, Washington, D.C., 2011, pp. 50–58.
- Liu, F., J. Evans, and T. Rossi. Recent Practices in Regional Modeling of Nonmotorized Travel. In *Transportation Research Record: Journal of the Transportation Research Board No. 2303*, TRB of the National Academies, Washington, D.C., 2012, pp. 1–8.
- Lowry, M., D. Callister, M. Gresham, and B. Moore. Assessment of Communitywide Bikeability

- with Bicycle Level of Service. *Transportation Research Record: Journal of the Transportation Research Board No. 2314*, TRB of the National Academies, Washington, D.C., 2012, pp. 41–48.
- Luber, M., L. Spinello, and K. O. Arras. People Tracking in RGB-D Data With On-line Boosted Target Models. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011.
- L. Xu and J. Landabaso, Segmentation and tracking of multiple moving objects for intelligent video analysis. *BT Technology Journal*, Vol. 22.3, 2004, pp. 140-150.
- Malinovskiy, Y., Y. Wu, Y. Wang, and U. Lee. Field Experiments on Bluetooth-based Travel Time Data Collection. CD-ROM. Transportation Research Board of the National Academies, Washington, D.C., 2010.
- Martin, P. T. “Turning Movement Estimation in Real Time.” *Journal of Transportation Engineering* 123, no. 4 (1997): 252–260.
- McCahil, C., and N. Garrick. The Applicability of Space Syntax to Bicycle Facility Planning. In *Transportation Research Record: Journal of the Transportation Research Board No. 2074*, TRB of the National Academies, Washington, D.C., 2008, pp. 46–51.
- Menghini, G., N. Carrasco, N. Schüssler, and K.W. Axhausen. Route Choice of Cyclists in Zurich. *Transportation Research Part A*, Vol. 44, 2010, pp. 754–765.
- MioVision, <http://miovision.com/wp-content/uploads/sample-reports/Intersection-Count-PDF-Report.pdf>. Accessed July 11, 2013.
- Nihan, Nancy L., and Gary A. Davis. “Application of Prediction-Error Minimization and Maximum Likelihood to Estimate Intersection O-D Matrices from Traffic Counts.” *Transportation Science* 23, no. 2 (May 1, 1989).
- Nakatsuji, Takashi, Kenji Nakano, Chumchoke Nanthawichit, and Hironori Suzuki. “Estimation of Turning Movements at Intersections: Joint Trip Distribution and Traffic Assignment Program Combined with a Genetic Algorithm.” *Transportation Research Record: Journal of the Transportation Research Board* 1882, no. -1 (January 1, 2004): 53–60.
- National Bicycle and Pedestrian Documentation Project (NBPDP), Institute of Transportations Engineers and Alta Planning, Accessed July 17, 2013 <http://bikepeddocumentation.org/>
- Penn, A., B. Hillier, D. Banister, and J. Xu. Configurational modelling of urban movement

- networks. *Environment and Planning B: Planning and Design*, 1998, pp. 59–84.
- Porter, J.D., Kim, D.S., Magaña, M.E. (2011). Wireless Data Collection System for Real-Time Arterial Travel Time Estimates. ODOT Research Report Number OR-RD-11-10. Oregon Department of Transportation, Salem, OR.
- Pucher, J., R. Buehler, D. R. Bassett, and A. L. Dannenberg. Walking and cycling to health: a comparative analysis of city, state, and international data. *American Journal of Public Health*, Vol. 100, 2010, pp. 1986-1992.
- Puckett, D.D., and M.J. Vickich. Bluetooth-Based Travel Time/Speed Measuring Systems Development, Final report of University Transportation Center for Mobility project report #09-00-17, Texas Transportation Institute, 2010.
- Quayle, S.M., P. Koonce, D. DePencier, and D. Bullock. Arterial Performance Measures Using MAC Readers: Portland Pilot Study. CD-ROM. Transportation Research Board of the National Academics, Washington, D.C., 2010.
- Quayle, S.M., and Koonce. Arterial Performance Measures Using MAC Readers: Portland's Experience. NATMEC Conference, June 23, 2010.
- Raford, N., A. Chiaradia, and J. Gil. Space Syntax: The Role of Urban Form in Cyclist Route Choice in Central London, *Recent Work, Safe Transportation Research & Education Center*, Institute of Transportation Studies, UC Berkeley, 2007.
- Raford, N., and D. Ragland. Space Syntax: Innovative Pedestrian Volume Modeling Tool for Pedestrian Safety, In *Transportation Research Record: Journal of the Transportation Research Board*, 1674, TRB of the National Academies, Washington, D.C., 1999, pp. 94–101.
- Research Board No. 1878, TRB of the National Academies, Washington, D.C., 2004, pp. 66–74.
- Ravi, Email communication, Saveri Networks Sales Representative. June 29, 2013.
- Replogle, M. “Integrating Pedestrian and Bicycle Factors into Regional Transportation Planning Models: Summary of the State-of-the-art and Suggested Steps Forward.” Presented at the Urban Design, Telecommuting and Travel Forecasting Conference, 1995.
- Robillard, P., 1975. “Estimating the O-D Matrix from Observed Link Volumes”. *Transportation*

Research 9, pp. 123-128.

- Saito, Forbush, (2011). “Automated Delay Estimation at Signalized Intersections: Phase I Concept and Algorithm Development” Utah Department of Transportation Research and Development Division.
- Salas, J., and C. Tomasi. People detection using color and depth images. *Pattern Recognition*, Springer Berlin Heidelberg, 2011, pp. 127-135.
- Savari Networks. <http://www.savarinetworks.com/files/MobiWAVE-DS-Family.pdf>. Accessed: October 8, 2013.
- Schwartz, W. L., Porter, C. D., Payne, G. C., Suhrbier, J. H., Moe, P. C., and Wilkinson III, W. L. (1999). *Guidebook on Methods to Estimate Non-Motorized Travel: Overview of Methods*.
- Seera, S., N. Brandlea, and C. Rattib. (2012). “Kinects and Human Kinetics: A New Approach for Studying Crowd Behavior”. arXiv preprint arXiv:1210.2838 (2012)(<http://arxiv.org/abs/1210.2838>).
- Shimbel, A. Structural Parameters of Communication Networks. *The Bulletin of Mathematical Biophysics*, Vol. 15, 1953, pp. 501–507.
- Smaglik, E.J., Vanjari, S., Totten, V., Rusli, E., Ndoeye, M., Jacobs, A., Bullock, D.M., and Krogmeier, J.V. “Performance of Modern Stop Bar Loop Count Detectors over Various Traffic Regimes.” In *Transportation Research Board 86th Annual Meeting*, 2007.
- Smaglik, E. J., et al. (2011). “Comparison of Alternative Real-Time Performance Measures for Measuring Signal Phase Utilization and Identifying Oversaturation” Transportation Research Board, National Research Council, Washington, DC.
- Spinello, L., K. O. Arras. People Detection in RGB-D Data. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011.
- Stowers, J., M. Hayes, A. Bainbridge-Smith. Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor. 2011 IEEE International Conference on Mechatronics (ICM), 2011, pp.358-362.
- Stinson, M., and Bhat, C. “Commuter Bicyclist Route Choice: Analysis Using a Stated Preference Survey.” *Transportation Research Record: Journal of the Transportation Research Board*, 1828, TRB of the National Academies, Washington, D.C., 2003, 107–115.

- Tian, J., M. R. Virkler, and C. Sun. "Field Testing for Automated Identification of Turning Movements at Signalized Intersections." *Transportation Research Record: Journal of the Transportation Research Board* 1867, no. -1 (2004): 210–216.
- Tremeau, Alain, and Nathalie Borel. A region growing and merging algorithm to color segmentation. *Pattern recognition*, Vol. 30.7, 1997, pp. 1191-1203.
- Tsubota, T., A. Bhaskar, E. Chung, and R. Billot. Arterial traffic congestion analysis using Bluetooth duration data. *Australasian Transport Research Forum 2011 Proceedings*, Adelaide, Australia, 28-30 September 2011.
- Trip Generation Manual, 9th Edition*. Institute of Transportation Engineers. 2012.
- Turner, S., A. Hottenstein, and G. Shunk. *Bicycle and Pedestrian Travel Demand Forecasting: Literature Review*. Research Study No. 0- 1723, Report 1723-1. Texas Transportation Institute; FHWA, U.S. Department of Transportation, 1997.
- UDOT, (2013). "<http://udottraffic.utah.gov/signalperformancemetrics/>"
- U.S. Dept. Trans., Vehicle Safety Communications Project Task 3 – Final Report, Nat. Highway Traffic Safety Admin., Rep. DOT HS 809 859, Mar. 2005
- U.S. Dept. Trans., Vehicle Safety Communications Project – Final Report, Nat. Highway Traffic Safety Admin., Rep. DOT HS 810 591, Apr. 2006
- Wang, L., P. Waddell, and M. Outwater. Incremental Integration of Land Use and Activity-Based Travel Modeling. In *Transportation Research Record: Journal of the Transportation Research Board*, No. 2255, TRB of the National Academies, Washington, D.C., 2011 pp. 1–10.
- Wang, Z., et al. (2014). "Estimating Queue Length at Signalized Intersections Using Multi-Source Data A Shockwave Theory Approach.pdf" Transportation Research Board, National Research Council, Washington, DC.
- Wasson, J.S., J.R Sturdevant, and D.M. Bullock. Real-Time Travel Time Estimates Using Media Access Control Address Matching. *ITE Journal*, Vol. 78, No.6, 2008, pp. 20--23.
- Wu, S., S. Yu, W. Chen. An attempt to pedestrian detection in depth images. 2011 Third Chinese Conference on Intelligent Visual Surveillance (IVS), 2011.
- Won, K. H., S. Gurmu, S. K. Jung. Pedestrian Detection using Labeled Depth Data. 2013 IEEE

19th Korea-Japan Joint Workshop on Frontiers of Computer Vision, 2013.

Xia, L., C.-C. Chen and J. K. Aggarwal. Human Detection Using Depth Information by Kinect. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2011

Yi, Ping, Chun Shao, and Jialei Mao. “Development and Testing of an Automated Turning Movement Identification System” (February 2010).
<http://trid.trb.org/view/2010/M/1127254>.

Yuan, Yufei, R. Eddie Wilson, Hans Van Lint, and Serge Hoogendoorn. “Estimation of Multiclass and Multilane Counts from Aggregate Loop Detector Data.” *Transportation Research Record: Journal of the Transportation Research Board* 2308, no. 1 (2012): 120–127.

Zhang, Y., X. Wang, P. Zeng, and X. Chen. Centrality Characteristics of Road Network Patterns of Traffic Analysis Zones. In *Transportation Research Record: Journal of the Transportation Research Board*, No. 2256, TRB of the National Academies, Washington, D.C., 2011, pp. 16–24.

Zivkovic, Zoran. Improved adaptive Gaussian mixture model for background subtraction. 2004. IEEE ICPR 2004. Proceedings of the 17th International Conference on Pattern Recognition, Vol. 2, 2004.

APPENDIX

Appendix 3.1 GIS Tools

1 Map Count Data Help File

Title: 1 Map Count Data

Summary: This tool is used to map observed count data into a polyline shapefile and a point shapefile. First, it maps the turning movement volumes by drawing turning movement polylines around each observed intersection and mapping the counts to the correct turning movement. Second, it maps the total intersection volumes as a point file, where it creates a new point for each observed intersection.

Syntax: MapCountData*i*Count (Input__Folder_of_excel_files, AM_PM, Year, Time_Period, Output__Turn_Movement_Shapefile, Output__Intersection_Points, Map_to_Augmented_Network, Input_Network, Output_Network)

Parameter	Explanation	Data Type
Input__Folder_of_excel_files	Dialog Reference This is a folder containing a series of files that represent count data at different observed locations. The count data need to be contained in an Excel file containing the iCount Data Entry Form in order for this tool to correctly parse the count data. There is no python reference for this parameter.	Folder
AM_PM	Dialog Reference This input is a filter that determines what time of day you want to map the observed count data for. AM refers to mapping the counts observed during morning time period. PM refers to mapping the counts observed during evening time period. Both refers to mapping the counts observed during the morning and evening	String

	<p>time periods. There is no python reference for this parameter.</p>	
Year	<p>Dialog Reference This input is a filter that determines what year you want to map the observed count data for. 2011 refers to mapping count data collected during 2011. 2012 refers to mapping count data collected during 2012. Both refers to mapping count data collected during 2011 and 2012. There is no python reference for this parameter.</p>	String
Time_Period	<p>Dialog Reference This input is a filter that determines what analysis period you want to map the observed count data for. 15 minute peak refers to mapping the peak 15 minute bicycle volume for each individual turning movement at each intersection. 1 hour peak refers to mapping the peak 1 hour bicycle volume for each individual turning movement at each intersection. 2 hour total refers to mapping the total bicycle volumes that were observed for either the AM or PM count periods. There is no python reference for this parameter.</p>	String
Output__Turn_Movement_Shapefile	<p>Dialog Reference The turning movement output feature class to be created. This feature class includes individual turning movement counts mapped to the individual turning movement polylines drawn around each observed intersection. There is no python reference for this parameter.</p>	Feature Class
Output__Intersection_Points	<p>Dialog Reference The intersection points output feature class</p>	Feature Class

	to be created. This feature class includes total intersection volumes mapped to a single point for each observed intersection. There is no python reference for this parameter.	
Map_to_Augmented_Network	<p>Dialog Reference</p> <p>This input determines if the count data will be mapped to the bikeways network or not.</p> <p>Yes means the count data will be mapped to the bikeways network.</p> <p>No means the count data will not be mapped to the bikeways network</p> <p>There is no python reference for this parameter.</p>	String
Input_Network (Optional)	<p>Dialog Reference</p> <p>If 'Map to Augmented Network' is set to yes then this input provides the bikeways network polyline shapefile for the count data to be mapped to.</p> <p>There is no python reference for this parameter.</p>	Feature Layer
Output_Network (Optional)	<p>Dialog Reference</p> <p>The output bikeways network feature class to be created. This feature class includes the filtered observed counts for each turning movement of each observed location.</p> <p>There is no python reference for this parameter.</p>	Feature Class

Appendix 3.2 Define Bicycle Impedance Help File

Title: 2 Define Bicycle Impedance

Summary: This tool is used to define the impedances of each element of a bikeways network.

The calculation is based on the input fields the user selects and is calibrated to represent impedances that effect a bicyclist's route choice.

Higher values make a link less attractive. For example, the impedance might be length. As another example, the impedance might be length * (slope factor), and this would increase the impedance for the link.

Syntax: DefineImpedance (Input__Network, Slope_Field, Friction_Field, Output_Impedance_Field_Name, Output_Network)

Parameter	Explanation	Data Type
Input__Network	Dialog Reference This is the input bikeways network feature class. This feature class is a polyline shapefile that represents the streets and shared-us paths that permit bicycle travel.	Feature Layer
Slope_Field (Optional)	Dialog Reference This field should provide the slope for every link. A link with a higher slope value will be more difficult for a bicyclist to traverse and thus increase the impedance of the link.	Field
Friction_Field (Optional)	Dialog Reference This field should provide the friction for every link. Friction could be anything from vehicle volumes to BLOS to the speed limit of the link. The friction field is representative of a link characteristic that might cause a bicyclist to choose a different route. For example, if BLOS is selected then a BLOS score of 1 (BLOS = A) would not increase the impedance of the link, but a BLOS score	Field

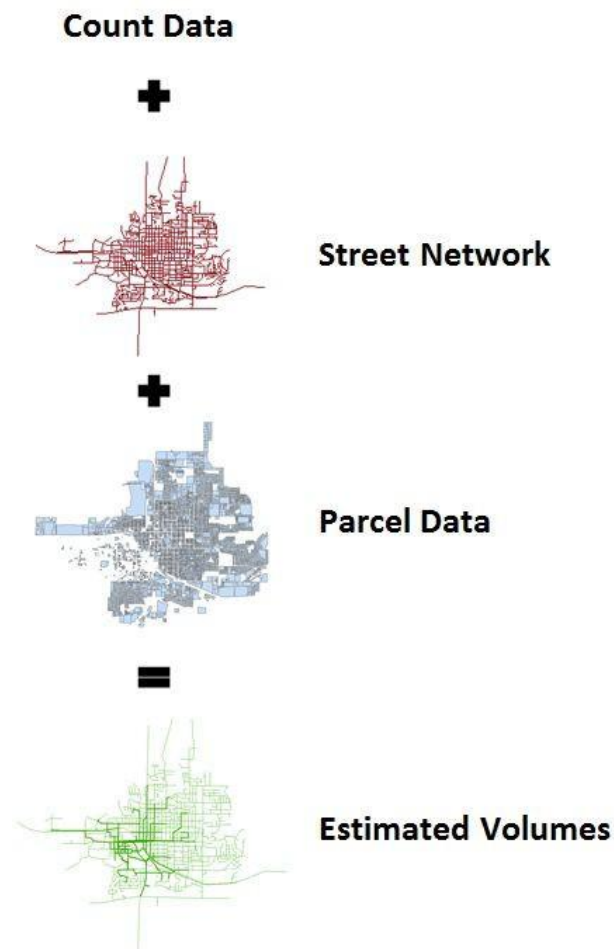
	of 4 (BLOS = D) would increase the impedance of the link.	
Output_Impedance_Field_Name	<p>Dialog Reference This is the name of the impedance field to be created. This name will be referenced in the Estimate Bicycle Volumes tool to assist in determining the shortest path between an origin and destination.</p>	String
Output_Network	<p>Dialog Reference The output feature class to be created. This feature class includes the impedance values for each street segment and shared-use path in the bikeways network.</p>	Feature Class

Appendix 3.3 Estimate Bicycle Volumes Help File

Title: 3 Estimate Bicycle Volumes

Summary: This tool is used to estimate bicycle volumes throughout a network from observed count data at sample locations. The calculation is based on the number of times a link in the network is used as the shortest path between user-supplied origins and destinations. The origins and destinations can be given "weight" multipliers as needed.

Illustration



Usage: There is no usage for this tool.

Syntax: EstimateBicycleVolumes (Input_1__Count_Data, Input_2__Origins, Origins_Multiplier_Field, Input_3__Destinations, Destinations_Multiplier_Field, Input_4__Bikeway_Network, Impedance_Field, Additional_Explanatory_Variables, Output__Estimated_Volumes)

Parameter	Explanation	Data Type
Input_1__Count_Data	<p>Dialog Reference This is a folder containing a series of files that represent count data at different observed locations. The count data need to be contained in an Excel file containing the iCount Data Entry Form in order for this tool to correctly parse the count data. There is no python reference for this parameter.</p>	Folder
Input_2__Origins	<p>Dialog Reference This is a feature layer represented by a polygon shapefile consisting of parcels to be used as trip origins. This shapefile must contain parcels as well as parcel area or the number dwelling units per parcel depending on if the origins are made up of non-residential or residential parcels, respectively. There is no python reference for this parameter.</p>	Feature Layer
Origins_Multiplier_Field (Optional)	<p>Dialog Reference The attribute field used to weight the origins. For example, number of dwelling units or square footage or trip production rate. There is no python reference for this parameter.</p>	Field
Input_3__Destinations	<p>Dialog Reference This is a feature layer represented by a polygon shapefile consisting of parcels to be used as trip destinations. This shapefile</p>	Feature Layer

	<p>must contain parcels as well as parcel area or the number dwelling units per parcel depending on if the origins are made up of non-residential or residential parcels, respectively.</p> <p>There is no python reference for this parameter.</p>	
Destinations_Multiplier_Field (Optional)	<p>Dialog Reference The attribute field used to weight the destinations. For example, number of employees or square footage or trip attraction rate.</p> <p>There is no python reference for this parameter.</p>	Field
Input_4__Bikeway_Network	<p>Dialog Reference This input provides the bikeways network polyline shapefile for the count data to be mapped to. This file must contain street center lines and shared-use paths.</p> <p>There is no python reference for this parameter.</p>	Feature Layer
Impedance_Field	<p>Dialog Reference This field should provide the impedance (also called travel cost) for every link. Higher values make a link less attractive. For example, the impedance might be link length. As another example, the impedance might be travel time = length/speed limit. The values must be greater than 0. (Note: give a huge value to a link that you want to keep in the network, but should not be used for travel, for example, a bike path restricted from car travel).</p> <p>There is no python reference for this parameter.</p>	Field
Additional_Explanatory_Variables (Optional)	<p>Dialog Reference All attribute fields to be used as explanatory variables during the regression process. For example, centrality and/or functional classification.</p> <p>There is no python reference for this parameter.</p>	Multiple Value

Output__Estimated_Volumes	Dialog Reference The output feature class to be created. This feature class includes estimated volumes for each street segment in the network. There is no python reference for this parameter.	Shapefile
---------------------------	---	-----------

Appendix 3.4 Pedestrian Model Fit Analysis

The NBPD data collected by the city for the iCount survey had both bicycle and pedestrian turning movement counts, and a model for pedestrians was developed in addition to the bicycle models. The pedestrian analysis included fitting regression models to the AM and PM time periods for the 2011, 2012, and a combination of 2011 and 2012 observed count data.

There were three primary differences between the bicycle and pedestrian models. First, the reachable target threshold distance was set to 0.9 because this provided the highest correlation between *OD centrality* and the observed counts, this analysis is shown in Figure 3.1 in Section 3.3.3. Second, the impedance used for the shortest path algorithm was distance; this is because a pedestrian route choice model was not formulated for this research. Finally, the third main difference was that the pedestrian model was analyzed for directional movements as well as gate counts. Each typical intersection consisted of eight gate counts including an entrance and exit gate count for each approach. To determine the gate counts, the observed movement counts corresponding to each gate location were summed, i.e. the south bound exit gate consisted of the SBT, EBR and WBL movement counts.

These models were fit using two different sets of trip multipliers. The first set of trip multipliers was the same as the trip multipliers used for the bicycle models. The second set of trip multipliers was based on the *ITE Trip Generation Manual*. The results of these analyses are shown below in Table A1 and Table A2.

TABLE A1 Pedestrian Model Fit Results: ITE Trip Generation Multipliers

		Directional Counts		Gate Counts	
		R ² (y)	R ² LN(y)	R ² (y)	R ² LN(y)
2011	AM	0.08	0.14	0.14	0.16
	PM	0.28	0.21	0.37	0.36
2012	AM	0.15	0.14	0.22	0.24
	PM	0.19	0.13	0.36	0.29
Both Years	AM	0.09	0.14	0.15	0.18
	PM	0.26	0.18	0.36	0.29

Note: R² (y) is the R-Squared value for the untransformed dependent variable. R² LN(y) is the R-Squared value of the natural log transformed dependent variable.

Note: Impedance = Distance, $\delta = 0.9$ miles, Multipliers: *ITE Trip Generation Manual*

TABLE A2 Pedestrian Model Fit Results: OD Multipliers from Section 2

		Directional Counts		Gate Counts	
		R ² (y)	R ² LN(y)	R ² (y)	R ² LN(y)
2011	AM	0.17	0.16	0.34	0.21
	PM	0.34	0.26	0.47	0.40
2012	AM	0.20	0.16	0.32	0.28
	PM	0.32	0.17	0.40	0.23
Both Years	AM	0.17	0.16	0.34	0.21
	PM	0.31	0.18	0.47	0.28

Note: R² (y) is the R-Squared value for the untransformed dependent variable. R² LN(y) is the R-Squared value of the natural log transformed dependent variable.

Note: Impedance = Distance, $\delta = 0.9$ miles, Multipliers: Residential = Dwelling Units and Non-Residential = Modified Area.

The results shown in Table A1 and Table A2 show poor model fits for the pedestrian models explored. Future work could explore why these models did not perform as well as the bicycle models. This work could look into formulating a shortest path algorithm that better

represents impedances affecting pedestrian route choice, or the future work presented in Section 3 could result in more acceptable model results.

Appendix 6.1: Simplifying the “a0” Matrix

Step 10 of the method requests that matrix “a” be simplified by Gauss-Jordan elimination row operations. There are various ways to do this. First, recognize that errors are present in detector counts. In addition, if the matrix has more rows than columns, then a solution may be attempted for different combinations of rows. As a result, because of count errors, each feasible combination will produce different estimates. The following is a method that will minimize the variance in the turn movement count errors. The method accomplishes this by minimizing the number of counts involved in estimating a turn movement count.

This method uses the three elementary row operations to simplify the matrix coefficients: (1) swap rows, (2) multiply a row by a non-zero constant, and (3) add a factor of one row to another row.

First, sort the rows in the “a0” matrix in ascending order by the number of nonzero coefficients they contain. This will allow the process to prioritize the use of rows that require the least number of alterations to simplify the matrix.

Next, use row operations to achieve an identity matrix of the coefficients, thus solving for the turn movement counts. When pursuing row echelon form, lower rows are shifted down to maintain as much of the initial sorting as possible. In addition, each row containing only zero coefficients is deleted, leaving an identity matrix.